Unsupervised Lexical Learning as Inductive Inference

Chun Yu Kit

Carnegie-Mellon University

M.Phil. Linguistics (1993)
City University of Hong Kong

Chinese Academy of Social Sciences

Tsinghua University

A dissertation
submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy

Department of Computer Science
and
Institute for Language, Speech and Hearing
University of Sheffield

September 2000

© Copyright 2000 by Chun Yu Kit. All rights reserved.
Unsupervised Lexical Learning as Inductive Inference

Chun Yu Kit

Submitted for the degree of Doctor of Philosophy
Department of Computer Science and
Institute for Language, Speech and Hearing
University of Sheffield

Abstract

To learn a language, the learners must first learn its words, the essential building blocks for utterances. The difficulty in learning words lies in the unavailability of explicit word boundaries in speech input. The learners have to infer lexical items with some innately endowed learning mechanism(s) for regularity detection – regularities in the speech normally indicate word patterns.

With respect to Zipf's least-effort principle and Chomsky's thoughts on the minimality of grammar for human language, we hypothesise a cognitive mechanism underlying language learning that seeks for the least-effort representation for input data. Accordingly, lexical learning is to infer the minimal-cost representation for the input under the constraint of permissible representation for lexical items. The main theme of this thesis is to examine how far this learning mechanism can go in unsupervised lexical learning from real language data without any pre-defined (e.g., prosodic and phonotactic) cues, but entirely resting on statistical induction of structural patterns for the most economic representation for the data.

We first review cognitive aspects of lexical acquisition, introduce basic concepts and principles concerned with learning as inductive inference, especially Solomonoff’s inductive inference theory and Rissanen’s MDL principle, and present representative computational models of lexical learning. Then we present the Virtual Corpus system as an efficient approach for counting n-grams in large-scale corpora – each n-gram, as a possible word, needs to be examined during the learning – and formulate the description length gain (DLG) measure to evaluate the goodness of a lexical candidate in terms of its compression effect in bits. An unsupervised lexical learning algorithm is formulated as an optimisation process to segment an utterance into candidates with the greatest sum of DLG. Learning experiments on child-directed English speech data illustrate that our learning approach reaches the state of the art of computational lexical learning.
Acknowledgements

First of all, I wish to thank Yorick Wilks, my supervisor, for his enthusiastic support, guidance and various kinds of help, without which I would have been unlikely to finish my graduate studies and thesis work. I also want to thank the other members of my PhD research panel: Robert Gaizauskas, my advisor, and Mick Perkins, my tutor and the chairperson of the panel.

Special thanks go to Randy LaPolla and Jonathan Webster, who helped proofread this thesis. In addition to proofreading the entire thesis, Randy also gave me valuable comments and advice on many issues related to language acquisition and English writing.

I wish to thank Ming Li and Paul Vitányi for helpful discussion on Kolmogorov complexity theory, learning with the MDL principle and other theoretical issues related to my thesis work. I also thank Ted Dunning and Steve Renals, for answering my questions about statistics, information theory and language modelling.

Thanks to all staff and researchers in the NLP group in the CS department at Sheffield who gave me help during my stay, especially Roberta Catizone, Hamish Cunningham, Mark Hepple, Mike Johnson, Kevin Humphreys, Paul McKeivitt, Takahiro Wakao, and Mark Steveson. Hamish and Kevin helped me out of many troubles in C/C++ programming and LaTeXing, respectively.

Many thanks to fellow students in the NLP group, especially Kalina Bontcheva, Rob Collier, Gael Dias, Hamid Khosrav, Alex Kroto, Mark Lee and Paul Woods, for making a very nice research atmosphere in the NLP lab, which I really enjoyed as a home during my stay at Sheffield.

Thanks to Malcolm Crawford, the ILASH research facilitator, for all his help when I was working in the ILASH suite. Thanks to Tony Simons, for help on C++ troubleshooting. Thanks to all people on the c-users mailing list in the Sheffield CS department, for ideas and help. Thanks to Phil Green, who welcomed me to the department the second day after I arrived in Sheffield for the first time.

Thanks to visiting researchers in the NLP lab, especially Toru Hisamitsu and Kyo Kageura, for helpful discussion on my work.

Thanks to my professors in CMU’s Computational Linguistics Program who gave me very good training in the field, which enabled me to move on to pursue my PhD, especially Bob Carpenter, David A. Evans, Ted Gibson, Lori Levin, Brad Pritchett and Teddy Seidenfeld. Thanks to Gregory T. Grefenstette, who taught me a lot while
sharing a cubicle with me in the Lab for Computational Linguistics at CMU. Also, many thanks to my fellow students Yeyi Wang and Xiang Tong for their help at CMU.

Thanks to Changning Huang for leading me to the field of NLP during my undergraduate final year at Tsinghua University. Without his influence, I would not have gotten into the field of NLP and would not have done this PhD.

Thanks to my wife Queenie Chau, and my daughter Nicole Kit, for bearing with me for so many years while I was entirely occupied by my thesis research. And special thanks to my son Jonathan Kit, for pushing me so hard to work on my thesis in part-time in the past two years – almost every evening after dinner, this two-year-old boy pointed to my nose and passed his mother’s order to me in Cantonese: “Cungloeng! Duksyu!” (Take a shower, and work on your book!). This thesis is dedicated to them, and to my parents, who have been pushing me to do a PhD since my childhood, to glorify the family, and are still pushing me to have a half dozen more sons to enlarge the family.

Finally, I gratefully acknowledge that my PhD research was supported by a University of Sheffield Research Scholarship at the Institute for Language, Speech and Hearing (ILASH).
# Contents

1 Introduction ........................................... 1
   1.1 Overview ........................................ 1
   1.2 Theoretical Background and Hypothesis .......... 1
   1.3 The Problem of Lexical Learning ................. 5
   1.4 Achievements of the Research .................... 8
   1.5 Outline of the Thesis ............................ 9

2 Lexical Acquisition – A Cognitive Perspective .... 13
   2.1 Overview ........................................ 13
   2.2 A Brief Overview of Language Development ..... 14
   2.3 Speech Input for Lexical Acquisition .......... 18
   2.4 Pre-linguistic Infants’ Speech Perception ..... 22
   2.5 Speech Segmentation and Word Discovery ...... 28
      2.5.1 Speech Segmentation ........................ 29
      2.5.2 Cues in Speech for Word Discovery ..... 32
   2.6 Summary and Discussion ........................ 50

3 Learning as Inductive Inference ...................... 54
   3.1 Overview ........................................ 54
   3.2 Learning .......................................... 56
      3.2.1 What is Learning? .......................... 56
      3.2.2 Learning and Regularity Discovery ...... 59
      3.2.3 A Scenario of Language Learning ......... 62
   3.3 Inductive Inference ................................ 68
      3.3.1 Occam’s Razor ................................ 69
      3.3.2 Kolmogorov Complexity ...................... 69
      3.3.3 The Minimum Description Length Principle . 70
3.3.4 Solomonoff’s Coding Method for Inductive Inference 74
3.4 Modelling, Search and Compression in Relation to Learning 76
  3.4.1 Learning as Modelling 76
  3.4.2 Learning as Searching 82
  3.4.3 Compression in Relation to Lexical Learning 85
3.5 Evaluation – How Well Does It Learn? 90
  3.5.1 The Theoretical Criterion 90
  3.5.2 The Empirical Criterion 92

4 Computational Models of Lexical Learning 93
  4.1 Overview 93
  4.2 Why Computational Models? 94
  4.3 Olivier’s Word Grammar Model 98
  4.4 Brent and Cartwright’s DR Model 101
  4.5 De Marcken’s Concatenative Model 110
  4.6 Brent’s Probabilistically Sound Model and MBDP-1 Algorithm 118
    4.6.1 Incremental Search 124
    4.6.2 Viterbi Search with Dynamic Programming 124
  4.7 Summary 128

5 N-grams and the Virtual Corpus 130
  5.1 Overview 130
  5.2 N-gram Language Models 131
    5.2.1 Fixed-order n-gram Models 132
    5.2.2 Mixed-order n-gram Models 133
  5.3 Number of N-grams 135
  5.4 The Virtual Corpus 139
    5.4.1 Position Indexing 140
    5.4.2 Sorting 142
    5.4.3 Counting 147
    5.4.4 Retrieval 149
    5.4.5 Other Utilities 150
  5.5 Summary 150
6 A Goodness Measure for Lexical Learning
  6.1 Overview ................................................................. 152
  6.2 Criteria for Language Learning ................................. 153
  6.3 One-part versus Two-part Code ................................. 155
  6.4 Description Length Gain ............................................ 158
    6.4.1 How good is a Model? .................................. 158
    6.4.2 How Good is a Rule? .................................. 159
    6.4.3 Calculation of Description Length ..................... 161
    6.4.4 Calculation of Description Length Gain .............. 162
  6.5 A Best-First Learning Algorithm ............................ 164
  6.6 Preliminary Experiments ......................................... 166
    6.6.1 Input and Representation .................................. 166
    6.6.2 Phrase Learning ............................................. 171
    6.6.3 Lexical Learning ........................................... 173
    6.6.4 Discussion .................................................. 173
  6.7 Summary and Concluding Remarks ............................. 175

7 Learning Algorithms and Evaluation ............................. 177
  7.1 Overview ................................................................. 177
  7.2 Representation for Lexical Learning ......................... 178
    7.2.1 Input Data ..................................................... 179
    7.2.2 Learning Result .............................................. 182
  7.3 Learning Algorithms ................................................. 190
    7.3.1 Optimal Segmentation with DLG .......................... 191
    7.3.2 Unsupervised Lexical Learning .......................... 195
    7.3.3 Induction of Lexical Candidates via Optimal Segmentation 196
    7.3.4 Lexical Refinement ........................................... 197
    7.3.5 Word Segmentation with a Refined Lexicon ............ 198
  7.4 Evaluation ................................................................. 200
    7.4.1 Input Data ..................................................... 200
    7.4.2 Evaluation Measures ...................................... 202
    7.4.3 Learning Performance ...................................... 206
  7.5 Discussion ................................................................. 214
    7.5.1 Negative DLG Segmentation ............................... 214
7.5.2 Word Type Precision and Recall ......................... 217
7.5.3 Other Problems ........................................... 220
7.6 Summary ..................................................... 221

8 Conclusions .................................................... 223
8.1 Directions of Future Work ................................. 225
List of Tables

4.1 Brent and Cartwright’s illustration of the calculation of the length, in characters, of representations for a tiny corpus with different lexicons. 103
4.2 Illustration of the calculation of the length, in characters, of representations for a more natural corpus. Italic letters are used to highlight the difference among the representations. 104
4.3 Performance of Brent and Cartwright’s baseline WC segmentation and DR optimisation segmentation integrated with various types of phonotactic constraint, averaged over nine mother-child dyads. Standard deviations are shown in parentheses. 109
4.4 The effectiveness of DR optimisation and phonotactic constraints, shown by how much increment of precision and recall they have caused in Brent and Cartwright’s experiments. 109
4.5 An examplar lexicon and the correspondent representation for an artificial utterance in de Marcken’s concatenative model. ◦ is the concatenation operator, and the terminals have no representation. 112
4.6 Segmentation precision/recall and lexical precision of MBDP-1 and the other six algorithms re-implemented by Brent and coworkers for comparison. All data in the “Start” and “End” rows in this table are read by eye from the performance charts in [32], with about ±5% accuracy. 126
5.1 Execution time of brsort() and qsort() for VC construction for the PTB-II WSJ corpus and the Brown corpus 145
5.2 Pseudo-C++ code for n-gram counting on a sorted VC with frequency accumulation, and an illustration of the frequency accumulation on an artificial corpus 148
òx= T o IJLEGHK;[.Q&CLfSK;[^I8 @ JLK @ JL5%l%8Ol%JB:*[.Q&CF3I:*GH3I:&fzZCFE :1I[CLG%:8OE&JLKQ*[\PFE[.5x:JB:8;CF5)}
8O5JLQ @ [.5%l%8O5IPeCLZ lI[\ZSCLf @ C`L[\Z JBPL[¡R+* @ CF3%5x:-,tK;[.5IPL:1Y })[.J @ 1qh8;:18;:Q @ CF3%5x:\}
K;[.5IPL:1)} @ C`L[\Z JBPL[JL5%lJ.`L[\Z JBPL[{¦M±­JB:*:J @ 1I[.lCF5t:1I[kK;[\fz: =9=9=S=9=9=S=9=9=9= T.mLr
òx=  o IJLEGHK;[.Q9CLflI[ @ CFEGCFQ8;:8;CF5CLfDCLZ lA] @ KO3%EGqK;[^I8 @ JLK @ JL5%l%8Ol%JB:*[.Qk8O5x:*Ctv5I[\Z*]
PLZ JL8O5I[.lK;[^I8 @ JLK)8;:*[.E&Ql%3IZ 8O5IP:1I[kK;[^I8 @ JLKZ[^v5I[.E[.5x:G%ZC @ [.QQ =S=9=9=S=9=9=9= T.m ì
òx= d o IJLEGHK;[.Q CLfSv5I[\Z*]ªPLZ JL8O5I[.l¿K;[^I8 @ JLK+8;:*[.E&Q CF3I:*GH3I: fzZCFEÛ:1I[K;[^I8 @ JLKhZ[^v5I[^]
E[.5x:2G%ZC @ [.QQ\}A8O5iJLQ @ [.5%l%8O5IPkCLZ lI[\ZDCLf @ C`L[\Z JBPL[L}H[.J @ 1eh8;:1i8;:Q @ CF3%5x:\}IK;[.5IPL:1)}
@ C`L[\Z JBPL[JL5%lJ.`L[\Z JBPL[{¦M±­JB:*:J @ 1I[.lCF5t:1I[kK;[\fz: =9=9=S=9=9=9=S=9=9=S=9=9=9= T.m ò
òx= W o IJLEGHK;[.Q&CLf9K;[^I8 @ JLK8;:*[.E&Qi8O5 :1I[v5%JLKK;[^I8 @ CF5ÂJBfz:*[\Z DCLZ lÂQ*[\PFE[.5x:JB:8;CF5
3%Q8O5IP:1I[eZ[^v5I[.l¥K;[^I8 @ CF5)}/8O5JLQ @ [.5%l%8O5IPCLZ lI[\ZCLf @ C`L[\Z JBPL[L}[.J @ 1 h8;:1¥8;:Q
@ CF3%5x:\}K;[.5IPL:1)} @ C`L[\Z JBPL[JL5%lJ.`L[\Z JBPL[{¦M±­JB:*:J @ 1I[.lCF5t:1I[kK;[\fz: =9=S=9=9=9= T.mLV
òx= r j Q3%E&E&JBZa CLfG%ZCLsHK;[.E&Q8O5e:1I[9CLZ:1ICLPLZ JBGH1%8 @ :*Z JL5%Q @ Z 8;G%:Q2CLf:1I[ n [\Z 5%Q*:*[.8O5
@ CLZGH3%QJL5%lt:1I[.8;ZJLlXÇ3%Q*:E[.5x:h8O5e:1I[SG%Z[^]ªG%ZC @ [.QQ8O5IP =9=S=9=9=9=S=9=9=S=9=9=9= BFd
òx= ì M[.JBZ 5%8O5IPG[\ZfzCLZ E&JL5 @ [SCLf:1I[93%5%Q3IG[\Z`A8OQ*[.l K;[^I8 @ JLKK;[.JBZ 5%8O5IPJLK;PLCLZ 8;:1%E&Q2CF5
DCLZ l%Q =9=S=9=9=9=S=9=9=S=9=9=9=S=9=9=S=9=9=S=9=9=9=S=9=9=S=9=9=9=S=9=9=S=9=9=9= B ò
òx= ò M[.JBZ 5%8O5IPG[\ZfzCLZ E&JL5 @ [SCLf:1I[93%5%Q3IG[\Z`A8OQ*[.l K;[^I8 @ JLKK;[.JBZ 5%8O5IPJLK;PLCLZ 8;:1%E&Q2CF5
DCLZ lJL5%ltsCF3%5%ltECLZGH1I[.E[.Q =S=9=9=S=9=9=S=9=9=9=S=9=9=S=9=9=9=S=9=9=S=9=9=9= B ò
òx= mâp1I[S[^«[ @ :8;`L[.5I[.QQhCLf(:1I[ o/< JLK;PLCLZ 8;:1%E JL5%lt:1I[S`LCD[.K @ CF5%Q*:*Z JL8O5x: =S=9=9=9= ATLT
òx= Vã0DCFEGHJBZ 8OQ*CF5CLf/K;[.JBZ 5%8O5IPiG[\ZfzCLZ E&JL5 @ [CF5DCLZ l%Q+JL5%lCF5DCLZ l%Q¦JL5%lsCF3%5%l
ECLZGH1I[.E[.Q\»¦3%5%Q3IG[\Z`A8OQ*[.lK;[^I8 @ JLKK;[.JBZ 5%8O5IPeJLK;PLCLZ 8;:1%E&Q+`L[\Z Q3%Q¦:1I[ o/< JLKw]
PLCLZ 8;:1%E =S=9=9=9=S=9=9=S=9=9=9=S=9=9=S=9=9=S=9=9=9=S=9=9=S=9=9=9=S=9=9=S=9=9=9= AT.
òx= T\Á0DCFEGHJBZ 8OQ*CF59CLfHK;[.JBZ 5%8O5IPG[\ZfzCLZ E&JL5 @ [/CF5kDCLZ l%Q(JL5%l9CF5kDCLZ l%Q(JL5%lECLZGH1I[.E[.QAT.
òx= TLT¢0DCFEGHJBZ 8OQ*CF5¦CLfACF3IZK;[^I8 @ JLKFK;[.JBZ 5%8O5IP2JLK;PLCLZ 8;:1%E&Q\©\G[\ZfzCLZ E&JL5 @ [h8;:1¦:1I[/Q*:JB:*[^]
CLf]ª:1I[^]bJBZ:hG[\ZfzCLZ E&JL5 @ [ =9=9=9=S=9=9=S=9=9=S=9=9=9=S=9=9=S=9=9=9=S=9=9=S=9=9=9= AT.d
òx= T. o IJLEGHK;[.QCLfQ*[\PFE[.5x:JB:8;CF5%Qhh8;:1t5I[\PFJB:8;`L[k{¦M± =9=9=S=9=9=9=S=9=9=S=9=9=9= AT.r
òx= T.d § CLZ l:ba4G[9G%Z[ @ 8OQ8;CF5eJL5%ltZ[ @ JLKOK)CLf(:1I[k3%5%Q3IG[\Z`A8OQ*[.l K;[^I8 @ JLK)JLK;PLCLZ 8;:1%E&Q =9= AT ò

`A8O8O8


List of Figures

3.1  An example of LZ78 coding  ........................................ 89
4.1  De Marcken's learning algorithm for the concatenative model  ...... 114
6.1  Examples of natural language text corpora  ........................... 167
6.2  A simple example process of best-first learning  ..................... 170
6.3  A simple example of DCFG language model  .......................... 171
6.4  An example of hierarchical structure  .................................. 171
6.5  Phrase patterns output from phrase learning on the Brown corpus  .. 172
6.6  The PTB-II POS tags in addition to nouns involved in the phrase learning 173
6.7  Lexical items output from the best-first learning on a WSJ article  .. 174
7.1  A fragment of the original transcription of a mother's speech from the
     Bernstein corpus of the CHILDES collection  ........................ 180
7.2  A fragment of the input corpus after pre-processing to filter out non-
     speech content  .................................................................. 181
7.3  A fragment of the input corpus with spaces deleted  .................... 183
7.4  A fragment of the output from the optimal segmentation of the Bernstein
     corpus  ............................................................................. 184
7.5  A fragment of the output from the word segmentation using the refined
     lexicon  ............................................................................. 188
7.6  The Viterbi algorithm for optimal segmentation with an illustration  .. 192
7.7  The model of lexical learning behind our lexical learning algorithms  .. 195
7.8  Performance of the 0S and 0S*V' algorithms in EM iterations  ....... 200
7.9  Word coverage rate versus frequency and coverage rank of word  ...... 218
7.10 Word type precision and recall in terms of frequency rank of word  .. 219
Chapter 1

Introduction

1.1 Overview

This thesis documents the author’s research work on unsupervised lexical learning from naturally-occurring language data. In general, the research falls within the field of machine learning of natural language. Specifically, it develops an information-theoretical approach to the unsupervised learning of a lexicon from infant-directed speech data. The learning is formulated as a process of inductive inference to squeeze out statistical regularities from the input data so as to achieve a least-effort (i.e. most economic) representation for the given data set. The learning result is a list of strings most of which match the real words in the native speaker’s lexicon.

This introductory chapter aims to give a brief bird’s-eye introduction to the thesis research work on computational lexical learning that has been carried out, several parts of which have been published in individual research papers co-authored by the author and his supervisor. This chapter will first present, briefly, the theoretical background and basic assumptions for the research, and then introduces the problem of lexical learning, by which the scope of research is defined. Then, we give a summary of the achievements of this research. In the final section, we give an outline of the remaining chapters of the thesis.

1.2 Theoretical Background and Hypothesis

The theoretical inspiration for the research comes from algorithmic information theory or Kolmogorov complexity theory [322, 198, 55, 219], in particular, Solomonoff’s inductive inference theory [322] which is the first of the three origins of algorithmic information theory, Rissanen’s Minimum Description Length (MDL) principle [294, 297, 299], Wal-
lace and colleagues' Minimum Message Length (MML) principle [344, 345], Vitányi and Li's formulation of the ideal MDL in terms of Kolmogorov complexity [338, 337] and their idea (or "intuition", in their own terms) on how to conduct inductive inference via compression on given data by squeezing out the embedded regularities piece by piece [219] (p.351).

Chomsky’s linguistic theory about language acquisition [62, 63, 64, 65, 66, 67], in particular the concept of language acquisition device (LAD) 2, which is hypothesised (or, more precisely, believed) to be an innate ability of human infants specific for language learning, also has had a substantial theoretical influence on our research reported in this thesis.

The key point of Chomsky’s innateness hypothesis of relevance to our research can be summarised as follows: human learners are born with certain innate structures (or schemata) for perceiving, representing and storing language data and with certain abilities to operate on such structures, but what are to be represented with such structures, i.e., the content, is to be learned from real data. Of course, this learning is under certain constraints imposed by the innate human endowment for language learning. This assumption of innately guided learning of natural language has been used for a long time as a commonplace explanation for why human infants can learn a language so fast and so easily: they learn particular things in particular way.

Chomsky’s thoughts on the minimality of grammar for natural language and his arguments for the minimally necessary innate structure (and ability) for language learning faculty are of special impact on our work, although he did not propose any measure for the minimality. The minimality of grammar (i.e., a lexicon, in our case) is reflected in our learning process, which seeks for the minimal-cost representation for both the input data and the lexicon itself. This learning process is suitable, almost perfectly, to be formulated within the MDL framework. Also, the learner in our studies is assumed to

---

1 We are not going to make any subtle differentiation between MDL and MML. Theoretically they are both rooted in Kolmogorov complexity; philosophically they are similar inductive principles for both data modelling and, especially, unsupervised learning as inductive inference. To our point of view, they subtly differ only in some technical details. A lengthy discussion on the difference of MDL and MML can be found in a recent special issue of The Computer Journal (Vol.42, No.4, 1999) on Kolmogorov complexity. More important to our research is their philosophy. Thus, in the rest of the thesis, the abbreviation MDL is assumed to subsume both.

2 Notice that this is the term in Chomsky’s early theory referring to the innate endowment of human species for learning natural language. The concept was elaborated and changed in Chomsky’s later work and, accordingly, referred to by various terms, e.g., language faculty with universal grammar (UG) as its initial state – the state independent of any language experience and before any language learning takes place.
have a minimal initial ability for learning, namely, the ability to count strings in the input data. What underlies the counting includes the ability to differentiate between strings (and symbols) and the ability to perform the arithmetic addition operation — other arithmetic operations that are used in the learning, i.e., multiplication, division and the use of logarithms, are regarded as extensions of addition. In this sense, our research can be thought of as an illustration of the possibility of learning words from natural speech data based on string counting: a learner able to count strings can learn words.

A theme in learning as inductive inference is to trace the underlying machinery that has generated the data as evidence for learning, via detecting the regularities embedded in the data. The regularity detection can be accomplished via compression with the aid of a universal coding scheme, e.g., Shannon-Fano code. Compression is argued to be an effective approach for an optimal approximation to the generally non-computable Kolmogorov complexity over a given data set [338, 337], where the data set is usually presented as a long sequence (i.e., string) of some elemental symbols. In the context of lexical learning, the data set consists of utterances, each of which is a sequence of atomic symbols in the language, i.e., either phonemes or letters. Conceptually, the Kolmogorov complexity of an object (e.g., a set of language data) is the length of the minimal effective representation for the object. Ideally, learning from a data set is to retrieve this minimal representation for the data, which means that all regularities from the data are extracted and the final result is a random sequence that cannot be further compressed. Since this minimal representation is not always reachable in general, what we can do is to compress the data as much as possible under the constraint(s) imposed by the representation format allowable in the learning. In this sense, the unsupervised learning (e.g., our unsupervised lexical learning, to be formulated in coming chapters) can be viewed as a formulation of Li and Vitányi’s “intuition”, mentioned above, about inductive inference under the MDL framework.

Following Solomonoff’s insight into the duality of compression and regularities [322], i.e., anything that can compress the data is a piece of regularity and any regularity must be able to be used to compress the data, we may state that, in general, a model that can compress the data to a greater extent can capture more regularities in the data, and is therefore a better model, in the sense that it is closer to the true machinery that has generated the data. In this sense, unsupervised learning as inductive inference means to derive the optimal set of regularities from the data that can compress the
CHAPTER 1. INTRODUCTION

data the most – once we have got the set of regularities via learning, it is another issue whether we really move on to use these regularities to conduct the compression. In the learning-via-compression approach, the compression is more a way of thinking about the computation involved in the learning process than a real procedure for compressing the input data.

The basic hypothesis for this thesis research in relation to human language acquisition is that there is a mechanism underlying human’s language learning faculty to seek for the least-effort representation for the input data. The rationale for this hypothesis includes, mainly, the least-effort principle in language first proposed by Zipf [362] and the thoughts on the minimality of grammar for human language in Chomsky’s linguistic theory. Wilks incorporates the least-effort principle as one of the main principles of his preference semantics [349, 350, 351], a computational theory for the semantic structure of natural language. Since natural language as a human behaviour is observed to be governed by the least-effort principle, it is reasonable to assume that the human language faculty has a least-effort learning mechanism (or strategy) to accommodate the input data, all of which are produced by native speakers observing the least-effort principle. The effort, or cost, in this context of unsupervised language learning is to be measured by the number of bits that are used to represent the input data and the language model itself (i.e., a lexicon in our case). Following this hypothesis, lexical learning observing the least-effort principle becomes the inference of the minimal-cost representation for the input under the constraint of permissible representation format for the lexicon. Computationally we assume that a lexicon is a list of words each of which is simply a string of atomic symbols in the language (e.g., phonemes in speech and characters in text).

The main theme of the thesis is to test how far this hypothesised language learning mechanism can go by itself in unsupervised lexical learning from real language data, without any help from a teacher or any pre-defined cues (e.g., prosodic cues and phonotactics), but merely relying on statistical induction to derive the structural patterns for the most economic representation for the input data. It is intended in the research that the learning follows the MDL principle restrictedly not only at the philosophical level but also at the operational level – the learner is intentionally implemented to calculate the description length for the input data in bits, instead of computing probability parameters to optimise a language model to fit the data, as many other researchers did in language modelling. Calculation of bits and probability are theoretically equivalent,
following Shannon's classic information theory [312, 78], but they lead to different operations within the learning process, particularly, when an optimisation process is involved. The operations for the optimisation on the number of bits in the representation for a string and on the probability of the string are technically different. It appears that the former approach is also worth serious exploration by some researchers.

1.3 The Problem of Lexical Learning

How do human infants learn a language? Many researchers have been seeking for a better understanding for this matter for many years, but the picture still remains unclear. Psycholinguists and cognitive scientists are amazed by the fact that children can learn a language so rapidly and almost effortlessly, although any language in the world looks so complicated in terms of its structures at various linguistic levels. How children can achieve this difficult learning task remains a central issue in psycholinguistics and cognitive science.

What has been clear, however, is the fact that language learners must first learn the sounds of their language, then the lexical items, in particular, words, and finally, they learn the grammar governing the production and comprehension of utterances together with the pragmatic constraints on the use of their language. Words are the essential building blocks that the grammar has to operate on while producing or comprehending an utterance. Without a lexicon of words, there is no grammar and thus no language. In this sense, lexical acquisition is at the centre of language learning.

However, the difficulty in learning words lies in the fact that adult speakers do not speak in isolated words either to each other or to language-learning infants and there is no explicit way to tell a pre-linguistic learner where word boundaries are in fluent speech. All language learners have to infer lexical items from continuous speech stream by themselves with the aid of some innately endowed learning mechanism(s) that can facilitate lexical learning via detecting the regularities and/or cues of various kinds embedded in the input – such regularities give indications of word boundaries.

Rationalists believe that children are born with some innate mechanism for language acquisition\(^3\), which is termed \textit{universal grammar} (UG) in Chomskyan linguistics. Clearly enough, it is unlikely that all language knowledge can be inborn in a child's mind, 

\(^3\)The terms “acquisition” and “learning” are used interchangeably within this thesis in general. But it is an explicit bias that the former is used more in relation to human learning (e.g., lexical acquisition) and the latter to machine learning (e.g., computational models for lexical learning).
CHAPTER 1. INTRODUCTION

otherwise all children would speak the same language.

It has been agreed that human infants are born with some basic mechanisms to detect, receive, store and represent language signals and primitive entities (e.g., sound units such as syllables and speech utterances of multiple syllables, between significant pauses of silence, just as they are born with some innate capability to see certain things (such as faces). On the basis of this essential inborn capacity for learning a language, human children still need to learn something (especially the structures and relationships among language signals and other primitive units) by themselves from their mother tongue environment, i.e., from the large volume of utterances spoken by their parents and other elders to each other or to the children. The critical question remains, however, as to which parts of language knowledge are inborn in a young infant’s mind and which parts are learned after birth. Although the mechanisms (or ability) for representing and memorising lexical forms, and other linguistic structures and properties of a lexical entry must be inborn, it is not feasible to argue against the fact that the lexicon of a particular language, especially, the individual lexical forms and their meanings, must be acquired by the infants during a certain period of exposure to the language evidence after birth.

What language-learning infants are exposed to in the process of language acquisition is fluent spoken utterances, each of which is a continuous stream of sound waves. How can they – infants who have a certain innate language learning capacity, as is assumed in some linguistic theories, but with little prior knowledge about any lexical forms and the correspondence between sound signals and lexical items – attain or build up the capacity to identify speech fragments in the sound signals as individual lexical items, detect their structural interrelationships with each other, and further associate particular meanings with the signal strings that have be identified as individual words?

In this thesis, we are not concerned with the question of how much of such a capacity must be innate in order to achieve the lexical learning task, nor how much linguistic knowledge can be learned based on this innate capacity. Rather, we are particularly interested in a bootstrapping problem at the onset of lexical learning – the time when a language-learning infant does not know any individual words: how can the infants, who initially have no prior knowledge about individual words (i.e., have an empty lexicon), detect individual words in the speech stream? This appears to be a chicken-and-egg problem: without knowing any words, how can an infant detect and segment individual words from the continuous speech stream? Without being able to detect individual
CHAPTER 1. INTRODUCTION

words in the speech stream, how can the infant expand its lexicon day after day by adding new words to it? To answer this question, we must find out what helps language-learning infants out of this chicken-and-egg circle. It appears that there must be some regularities, for example, distributional regularities and other sorts of cues, embedded in the language data that play a critical role in helping the infants out of the trap. But what regularities can the infants rely on, and how do they make use of such regularities to derive individual words?

The research reported in this thesis is an attempt to answer these questions. The purpose of the research on lexical learning covered in this thesis is threefold. First, it aims to test the hypothesis that there is a mechanism underlying language acquisition which seeks for the least-effort representation for the input data. Secondly, it explores machine intelligence with a focus on examining how much a computer can learn a natural language given that it has only a minimum innate capacity, that is, the capacity to differentiate between signals or, equivalently, characters in texts. Other related capacities, such as counting distinct signals and strings in a given corpus, are derived from this basic capacity. It would be an amazing achievement to demonstrate that a counting machine can learn words from natural language data with little supervision or guidance from a teacher. However, it is more important to notice that such success would not be due to the power of the learning mechanism involved. Rather, it would be because the strong regularities in empirical linguistic signals can be captured statistically and information-theoretically by a counting machine. It would show that the distributional regularities in language data play a critical role in the language learning by both human and machine learners.

Thirdly, it is hoped that the studies can help us to achieve a better understanding of the nature of the human language acquisition device that is assumed in Chomskyan linguistics to have a minimally necessary innate structure and ability. It is not intended in this thesis to draw too many conclusions about human language acquisition from the result of machine learning of natural language. It is merely an attempt to demonstrate the plausibility of the hypothesis that the mechanism of seeking for a least-effort representation can be an effective learning strategy for the human language faculty. In this sense, there is no doubt that the research on machine learning of natural language, especially when a minimum innate capacity is assumed, can shed light on the mechanism of human language acquisition: machine learning indicates that the strong regularities embedded in language data play a critical role in facilitating language acquisition in
addition to the innate mechanisms. This may help to explain why human infants learn a language so easily.

1.4 Achievements of the Research

The goal of the research has been to test the effectiveness of the learning strategy to infer the least-effort representation for input data in computational lexical learning. This strategy was hypothesised to be one of the learning mechanisms that underlie human infants’ language learning capacity.

The main achievements we have accomplished in the research can be summarised as follows. In the theoretical aspect, we first put forward the hypothesis that there is a least-effort learning mechanism underlying the human language faculty’s ability to learn linguistic structures, including lexical items, from the speech data that are produced observing the least-effort principle, and then we formulated this unsupervised learning strategy as an inductive inference within the MDL framework to derive the minimal-cost representation for the input data and the model (i.e., the lexicon) itself. We took a learning-via-compression approach to derive this minimal representation. For the purpose of this formulation, we defined the goodness measure description length gain, following information theory, to evaluate how much compression effect in bits we can gain from putting a word candidate into the lexicon. This goodness measure plays a critical role in guiding the learning towards the minimal representation.

In the technological aspect, we first implemented an efficient Virtual Corpus system using the suffix array technique, which is also known as PAT tree, for deriving n-grams of any length and their counts from large-scale corpora – this is the operational basis for the lexical learning, because it supports the learner to examine all possible candidates for words in the input data. Then, we formulated and implemented a Viterbi algorithm as an optimisation process to segment an utterance into chunks, each being a lexical candidate or a collocation of a number of frequently co-occurring words, with the greatest sum of description length gain over all chunks – this optimisation is again exploited to further segment the candidate chunks, many of which are collocations of words, into finer-grained lexical units. Having these intermediate chunks in the middle of the learning process appears to have reduplicated the progress of lexical acquisition by a human infants, who were also observed to possess multiple-word chunks as individual lexical units in their early stage of language acquisition [278]. In this sense, our lexical learner demonstrates that it works in a way highly consistent with human infants’ approach to learning a
lexicon from the language data to which they are exposed.

The evaluation of lexical learning performance in our work is also more comprehensive than in previous studies in computational lexical learning. In previous studies only precision and recall of words were used for evaluation. Such evaluation gives the same credit to the identification of a long word and that of a short word. In our evaluation, we not only used precision and recall on words but also used the precision and recall on word boundary and the correct ratio, namely, the proportion of the input data in the number of characters that are segmented into correct words. In terms of these comprehensive evaluation measures, our learning algorithm has demonstrated not only the state of the art of computational lexical learning with its learning performance but also the plausibility of the unsupervised learning mechanism as an effective learning strategy in human lexical development.

1.5 Outline of the Thesis

The remaining chapters of the thesis are organised as follows. Chapter 2 serves to give a cognitive, in particular, psycholinguistic, background for the research work of unsupervised lexical learning reported in this thesis. It reviews human lexical acquisition as a critical stage of language development. It first gives an overview of human language development in general and then moves on to specific cognitive aspects of lexical acquisition by infants, with special attention given to the characteristics of speech input for the lexical learning and to the pre-linguistic infant learners’ learning ability based on their speech perception, which is undergoing a rapid development during their first year of life. The focus of the chapter is on the learning strategies, especially the cue-based strategies that the pre-linguistic infants would exploit to locate lexical units in fluent speech input. One of the cue-based strategies is the metrical segmentation strategy that utilises the prosodic (in particular, rhythmic) characteristics of the speech input to infer possible word boundaries. Other cues that the infant learners are showed to take advantage of for speech segmentation and word discovery include allophonic cues, phonotactics and statistical distribution regularities in the speech input. We are specifically interested in the role that the distributional regularities might play in lexical acquisition.

Chapter 3 presents the theoretical background and foundation for the research covered in this thesis, with an emphasis on the basic concepts and principles of machine learning and information theory that are highly relevant to the unsupervised lexical learning. It first examines the problem of what learning is and gives a scenario of ma-
machine learning of natural language by examining the parties involved and the roles they
play in the learning, and then defines unsupervised learning from natural language as
inductive inference to extract regularities from the given data. Of special interest is
the initial ability of the machine learner for lexical learning, namely, the ability to do
little more than string counting – this is at an extreme along the line of Chomsky’s
thought on minimally necessary innate ability for human’s language learning faculty.
Next, we present the theoretical concepts and principles related to this kind of inductive
learning, namely, the basic concept of Kolmogorov complexity, Solomonoff’s inductive
inference and the formulation of the MDL principle, etc. There are also discussions
on the relation of text compression to the lexical learning and on learning as searching
through hypothesis space for the optimal hypothesis consistent with the given data.
During the discussion we formulate the basic idea of language learning via compression,
which is theoretically rooted in algorithmic information theory, in that we cannot, in
general, compute the shortest representation for a given input but can approach to an
approximation for it through compression. The underlying principles of text compression
related to lexical learning are then presented, with a focus on the dictionary-based
approaches.

Chapter 4 reviews previous studies on computational lexical learning, focusing on
a number of representative learning models and learning algorithms. We first have a
general discussion of computer simulation of language learning and its role in cognitive
studies of the learning mechanisms that lie beneath the human language faculty, and then
we have a detailed review on the individual computational models. These representative
models include Olivier’s word grammar, Brent and Cartwright’s distributional regularity
(DR) model, de Marcoen’s concatenative model and Brent’s probabilistically sound model
and the MBDP-1 algorithm. In addition to presenting the underlying theories and
techniques in these models and algorithms, we also analyse their pros and cons and,
especially, their influence on our studies.

Chapter 5 presents the operational basis for unsupervised language learning from
large-scale corpora. It starts with a brief look at n-gram language models and then
moves on to analyse the computational complexity of deriving and counting n-grams
in terms of the number of possible n-gram items in a given corpus. The result of the
analysis is that the number of all possible n-grams in a corpus is at most linear to
the length of the corpus – this shows the feasibility of lexical learning that relies on
n-gram counting. Afterward, this chapter focuses on reporting the technical details of
the implementation of the Virtual Corpus (VC) system based on the suffix array (i.e., PAT tree) structure for fast access to all n-grams of any length in a given corpus. The system provides a practical approach for searching through a huge number of n-gram items, each of which is a possible lexical item to be learned by the unsupervised learner. Computational lexical learning requires such a supporting tool, because all n-grams in the language input need to be examined in order to figure out which ones are good candidates for words and, finally, to determine the best set of candidates as the learning results that has the optimal compression effect on the input. This compression effect is to be calculated in bits by a theoretically sound and practically effective goodness measure, namely, the description length gain to be formulated in the next chapter.

Chapter 6 formulates a goodness measure to guide the lexical learning via compression within the MDL framework. It assumes a universal coding for the calculation of description length, for which either Shannon-Fano coding, Huffman coding or arithmetic coding would suffice for practical purposes. The average description length of a character in a given corpus is defined as the empirical entropy $- \sum \hat{p}(x) \log_2 \hat{p}(x)$ in bits, following the classic information theory. Accordingly, the goodness measure, termed as description length gain (DLG), is formulated in terms of the description length change after a piece of regularity (i.e., a string as a word candidate) is extracted from the given corpus and added to the lexicon. A simple best-first (or hill-climbing) learning algorithm within the learning-via-compression paradigm was implemented for the purpose of examining the reliability and effectiveness of this goodness measure. Experiments on both lexical and phrase learning showed promising results.

Chapter 7 first presents a number of unsupervised lexical learning algorithms guided by the DLG measure. We first introduce the representation for both the input and the learning results – input utterances and learned words are represented as plain strings, the simplest representation. Then we formulate a Viterbi algorithm for the segmentation of each individual utterance into an optimal sequence of chunks, mostly being words or collocations of words, that has a greater compression effect than any other segmentation. The lexical learning algorithm comprises two steps, both of which involve an application of the optimisation algorithm: the first step applies the algorithm to segment utterances into intermediate chunks and the second step applies it to the chunks to derive words as final results.

In the second half of Chapter 7 we report the experimental results from the lexical learning algorithms and the evaluation of their performance on child-directed speech
data from the CHILDES corpus [222, 220]. The results are evaluated by a number of evaluation measures including precision and recall of word and of word boundary and also by correct character ratio – the proportion of the input data segmented into correct words. The learning results indicate that the learning algorithm, based on the learning theory formulated in the preceding chapters, can capture very well the linguistic regularities in a naturally-occurring child-directed speech corpus at the lexical level. Our DLG based lexical learning approach demonstrates a learning performance at the top of the state of the art in the field of unsupervised lexical learning.

The last chapter, Chapter 8, concludes the research presented in the previous chapters and discusses possible directions of future work, in particular, the work of applying the theoretical framework to other language learning tasks, e.g., phrase learning from a part-of-speech (POS) tagged corpus and induction of context-free grammar for natural language. Furthermore, exploring the reality of DLG as a goodness measure for human lexical learning is also an interesting direction.
Chapter 2

Lexical Acquisition – A Cognitive Perspective

2.1 Overview

This chapter discusses cognitive aspects of lexical acquisition by human language learners. It gives the psycholinguistic and cognitive background for the research covered in the coming chapters of this thesis, by reviewing young children’s language development, with an emphasis on lexical acquisition. We will review the characteristics of the speech input that language-learning infants receive and will analyse the perceptual ability they have that enables them to learn to identify lexical items from the input, starting from an empty lexicon. The input and the perceptual ability together are the basis for an infant to acquire a lexicon for understanding and producing utterances. Without adequate speech input, an infant cannot learn a language. Without proper speech perception, an infant cannot receive adequate speech input, and therefore cannot acquire any language properly.

It is known that lexical acquisition is an important stage of language development. Words are the basic building blocks for utterances. Without words, there would be no phrases, no utterances, and therefore no syntax, no semantics, and, finally, no language. Therefore, lexical acquisition is thought of as a critical initial step, if not the very first step, towards a proper development of language competence.

In addition to reviewing existing studies on human infants’ lexical acquisition, we also ask some questions. For example, it is commonly believed by many psycholinguistic researchers that a number of prosodic cues in the speech input (e.g., child-directed speech) to the language-learning infants are critical factors that facilitate, and perhaps even enable, the lexical acquisition. A few of our questions concerning such prosodic
cues are the following. From where, and when, does an infant learn such cues? Does an infant know any such cues before knowing any words? Are these cues the starting point for lexical learning, or the result, or more precisely, the by-product, of an early period of lexical learning? If having knowledge of such cues is an indispensable starting point for lexical acquisition, a very interesting logical question would arise: without knowing words, at least some words, how could a learner know these cues are the cues for words?

Following these questions without any satisfactory answers, an even more interesting and more important question, which is one of the major motivations for this thesis study, that we would like to ask is: without any cues, even the most salient word boundary markers – the pauses in between phrases and utterances – can a learner, be it an infant or a machine, learn a lexic for a language from an adequate volume of speech input of the language? If we could get an affirmative answer for this question, even though the learner might not be able to get all lexical items completely correct, we would arrive at a more fundamental cognitive mechanism for lexical learning at the centre of language development than prosodic cues and other constraints (e.g., phonotactics). It is possible that all cues for words may be derived from this learning mechanism.

This chapter is organised as follows. In Section 2.2, we give a brief overview of child language development, serving to give a broad background to situate the cognitive, in particular, psycholinguistic, studies of lexical acquisition, to be discussed in later sections. Next, in Section 2.3, we discuss a number of idiosyncratic characteristics of the speech input to lexical acquisition that might play some critical role in facilitating, triggering and/or even bootstrapping the very initial stage of lexical acquisition. Section 2.4 reviews language-learning infants perceptual capacity for lexical learning. We are particularly interested in what abilities an infant is born with and what are later developed. Section 2.5 discusses some recent psycholinguistic studies on lexical acquisition about how a infant learns words with the aid of some special cues and constraints as well as inborn capabilities. Finally, in Section 2.6, we give a summary of this chapter, together with some discussions and criticisms.

2.2 A Brief Overview of Language Development

The fact that new-born babies only a few days old have a certain awareness of the difference between their own languages and other ones [244] and that infants a few days to a few months old are found to prefer to listen to natural language speech than other auditory inputs [74, 138] suggests that a child may begin language perception
and, therefore, language acquisition, before birth. It is argued that it is the human babies' inborn sensitivity to some specific prosodic properties in natural language speech that enables them to discriminate speech in their mother tongues from speech in other languages [244]. It is also noted that at the beginning infants are sensitive to all sound contrasts in all natural languages, but later this ability fades and the infants' speech perception gradually adapts to the phonology of their mother tongue [134].

Infants are born with a nascent structure-seeking mechanism to discover particular units with particular distributional patterns in natural language input, guided by innately specified structural constraints [284, 279, 169, 280, 281, 285, 282]. This mechanism is sensitive to the patterned organisation (e.g., rhythmic, temporal and hierarchical organisation) of natural language phonology common to all languages [123], be they spoken or signed [282], and the linguistic units and patterns learned by such mechanism correspond in size and organisation to phonetic and syllabic units common to all languages [282]. This direction of exploration is seen to follow the innateness hypothesis in Chomskyan linguistics that human species have an innate genetic endowment, known as universal grammar (UG), for acquiring natural languages [62, 63, 64, 65, 66].

A normal child starts to produce reduplicative babbling, composed of repeated syllables (bababa, dadada, etc.), within 6 to 10 months after birth. Some intonation patterns and some imitation of adults' speech are observed to appear during the late babbling stage, from 9 to 12 months. Interestingly, deaf children also begin to babble with their sign-language at a similar age, producing sequences of syllabic units\footnote{As in spoken languages, signed languages are constructed from a finite set of meaningless units (phonetic units); ... ASL’s phonetic inventory is drawn from the four parameters of a sign – handshape, movement, location, and palm orientation – each of which contains a restricted set of phonetic units (for example, a set of handshapes, a set of movements). Phonetic units are further organized into structured units called syllables.” “A well-formed syllable has a handshape, a location and a path movement (change in location) or secondary movement (change in handshape or orientation)” [283] (pp.1495, Note 20). ASL is American Signed Language, and “a sign [in signed languages] has identical linguistic properties to a word in spoken language” (Ibid, pp.1494).} that are observed to be fundamentally identical to vocal babbling produced by normal children who have an ordinary exposure to a spoken language [283]. Based on the observation of no significant difference between the acquisition of spoken and signed languages, it is argued, or suggested, that human infants' capacity for language acquisition is not specific for speech, rather, it is part of the children's cognitive capacity for acquiring abstract structures, including linguistic structure, from the surrounding world [267, 281, 283].

Children can understand many words long before they produce the first word [160]. Usually, a child produces one-word utterances roughly by the age of 10 to 11 months.
CHAPTER 2. LEXICAL ACQUISITION - A COGNITIVE PERSPECTIVE 16

The gap between comprehension and production is lexically great at this stage: a child may be able to understand about one hundred words when it starts to produce the first word [15]. The majority of the children’s early lexical items are names of individuals and objects in their environment, such as “Mama” and “car”. Some action verbs, which refer to actions that frequently take place around the children or related to them (e.g., “give”, “hit”, “drink” and “eat”), and a few adjectives (e.g., “big” and “good”) are also in the lexicon. Abstract words come into the lexicon much later. At the age of 16-18 months, the single word utterances seem to show some semantic categories (e.g., agent, action and object) [161], but have a very vague mapping to adult meaning, for example, a short utterance “cup” may mean “a cup is there”, “I see a cup” or “I want the cup”.

By the age of about 18 months, children start to produce two-word utterances, or more precisely, two-word phrases. The children also undergo a “word-spurt” or “naming explosion” at this stage. The number of words in a child’s lexicon increases rapidly in this period of language development. There is also evidence that the emergence of phrases in the child speech correlates to the word-spurt [263, 8]. A correlation between word learning and initial syntactic development is observed. For example, young children are sensitive to the syntactic information (e.g., part of speech) when inferring a new word’s meaning. They tend to guess that a verb-like new word refers to an action, a countable noun to a physical object or an individual, and a mass noun to a kind of substance or a piece of non-individual entity [42, 205, 318].

Next, children start to produce telegraphic speech [45]. The term “telegraphic speech” is specifically used in the study of child language development to refer to children’s short utterances that lack grammatical inflections and functional (or closed-class) words and/or morphemes, like determiners (e.g., “the”), prepositions (e.g., “of”) and suffixes (e.g., “-s” and “-ed”) [46]. Such utterances sound like telegrams using as few words as possible to convey essential meanings. Children tend to use telegraphic form even when they are trying to imitate adults’ full sentences. Omission of subject is a frequent phenomenon in children’s telegraphic speech [23]. Closed-class morphemes are observed to have a relatively fixed order to show up in children’s speech, e.g., in English, the morpheme “-ing” (as in “(is) talking”) for present progressive shows up earlier than third-person singular “-s” (as in “talks”) [44, 99]. The order (or tendency) of acquisition of a few frequent grammatical morphemes in English is: the present progressive “-ing” appears first, then the regular plural “-s”, possessive “-s”, irregular past tense forms and regular forms [360]. It is noted that semantic complexity and phonological salience
have a certain influence on the emergence order of these morphemes [100].

A very important phenomenon observed at this initial stage of grammatical competence is that children rarely make mistakes about word order [29, 28, 22, 44, 284, 24], indicating that children start to understand and master some fundamental syntactic properties in adults’ speech. There is evidence to support the idea that children have a certain grasp of word order knowledge even prior to their producing of telegraphic speech [151]. More interestingly, children even attempt to utilise word order to express grammatical relations at the initial phase of acquiring “free word order” languages, where grammatical relations are marked by case markers [267, 284].

Children start to produce multiple word utterances by the age of about two and a half years old. The average utterance length goes up gradually in the following few years. Accordingly, more and more functional words are used, and the utterances produced by children become more and more complex, in terms of grammatical structure. Yes-no questions, relative clauses and control structures (e.g., “I want him to come”) appear in the children’s speech. Children of five and six years old are able to add appropriate grammatical suffixes to words according to their grammatical classes, which can be induced from the context. This is known as the famous wug procedure, where wug is a word invented for experimental purposes [16]. Children also attempt to use, and invent, rules to infer morphological forms for a verb according to the time and tense. Interestingly, however, they make morphological mistakes, known as overgeneralization. Some frequently quoted examples are “goed” (vs. “went”), “comed” (vs. “came”) and “mans” (vs. “men”). Inappropriate usage of words also occurs in child speech, e.g., “I giggled the baby” instead of “I made the baby giggle”. Many parents try to correct their children’s mistakes like these. However, there is a consensus in child language development that parents’ correction of children’s speech mistakes, known as negative evidence, does not play an essential role in influencing the children’s language acquisition [145, 257, 26, 144, 231]. It is very interesting that in some cultures, adults do not talk to children before they have a full competence to speak, let alone correct their speech mistakes.

After the word spurt, children’s vocabulary grows steadily. It has been estimated that a child acquires about nine words per day from the age of 1.5 to 6 years [49]. It is also observed that there is a critical period for human language development. If a child has no chance to be exposed to any language by the onset of adolescence, the child seems to have a very slim chance to retain the ability to learn to speak in a language
with full-fledged syntax — there is empirical evidence for this from a number of cases of feral or isolated children, e.g., the Wild Boy of Aveyron [206] and Genie [81, 302]. If exposed to a language (including sign language) after the age of 7, children have less and less chance to become totally fluent in native accent. It is reported that few Chinese and Korean children who immigrated to USA after the age of 7 can become totally competent in American English [166]. A similar result is found with people acquiring ASL as their first language [265].

2.3 Speech Input for Lexical Acquisition

Infants who are born with the ability to learn natural language will not really learn a language if they are not exposed adequately to natural language speech. Two factors are critical in this exposure that enables the learning, one is the speech input to the learners, the other is what the learners really receive from the input based on their speech perception capabilities. In this section we will analyse the characteristics of the speech input to language-learning infants at the early stage of language development, in particular, the input for lexical acquisition, and discuss on what influence such input has upon the infants’ learning. In the next section we will review the pre-linguistic infants’ speech perception. The speech perception determines what an infant really receives from the surrounding speech environment.

From the time of their birth, infants are surrounded by adults’ speech, directed to other adults or to the preverbal infants. Young infants are reported to prefer to listen to infant-oriented “motherese” than to normal adult-to-adult speech [119]. The term motherese is used to refer to a kind of slow high-pitched speech with smooth, exaggerated intonation contours that adults tend to use to communicate with infants in many cultures [266]. It is argued in [118] that there exist certain universal speech patterns across languages and cultures in adults’ speech to infants, to soothe or express praise or disapproval with different prosodic contours — this kind of speech was even considered as a universal signal system that was believed to be based in human biology, a system of calls independent of the meaning carried by the speech but having a certain effect on children (including directing their attention or calming or arousing them).

Motherese, also known as baby talk and infant- or child-oriented speech, is characterised by many special features (as discussed in [132, 188, 327, 122, 123] and many others), e.g.,
CHAPTER 2. LEXICAL ACQUISITION - A COGNITIVE PERSPECTIVE

- Slower speaking rate
- Higher pitch
- A wider range of fundamental frequency
- Highly varied intonations, with significant exaggeration
- More frequent and longer pauses
- Simpler and shorter utterances
- More repetitive
- Special baby-words (e.g., *doggy* and *pussy*)
- More frequent onomatopoeia and interjections
- Restriction of topics to those relevant to a child’s world

The role of infant-directed speech in language acquisition, in particular lexical acquisition, is highly arguable. There is experimental evidence that young infants have a preference for listening to infant-directed speech over adult-directed speech. An experiment is reported in [119] that the 4-month-old infants in the experiment chose, by turning their heads, to listen to an infant-directed speech tape more frequently than to listen to an adult-directed speech tape. A further study in [120] found that this preference is strongly associated with the melody in the infant-directed speech, because the infants’ preference was observed to persist even when all but the melody was filtered out of the speech signal. It was also found later that even in the first month after birth, the new-born babies also had the preference for infant-directed speech [76], but to these younger infants, prosody alone appeared insufficient to maintain the preference – it was observed to be only associated with the full speech signal [77]. The findings in [120] and [77] suggest that new born babies receive the infant-directed speech signal as a whole at the very beginning, but by the age of about 4 months, they have learned, somehow, to isolate the pitch contours from the whole speech signal and have built up some kind of correlation between these contours and the positive interactions with their mothers – it is this correlation that leads to the preference for the prosodic contours in the infant-directed speech.

There are many conjectures about the role of infant-directed speech (in particular, its prosodic features) in the early stage of language acquisition, based on the young infants’
CHAPTER 2. LEXICAL ACQUISITION – A COGNITIVE PERSPECTIVE

preference for, and sensitivity to, the prosodic features in infant-directed speech. It is said that the correlation between various types of intonation and their affects may provide a first basis for children to understand the sound-meaning correspondence. It is also proposed that the significantly exaggerated intonation and stress patterns may provide important clues to help, or even trigger, the infants’ identification of lexical items and linguistic structures such as phrases and sentences in speech. The hypothesis that language learning heavily relies on the prosodic characteristics of the speech input to the learners is known as the prosodic bootstrapping hypothesis [137, 278, 253, 152, 136, 121] and has received considerable attention in recent years.

Many psycholinguistic experimental results were interpreted in a way to support this popular hypothesis. For example, the experimental results presented in [152], that 7- to 10-month-old infants prefer to listen to motherese utterances not interrupted by a pause over the other utterances interrupted by a pause, were interpreted as an indication that clauses are salient perceptual units for young infants. A subsequent study [191], using the same procedure as in [152], further found that preverbal infants only preferred the uninterrupted utterances over the interrupted ones in motherese, but not in adult-directed speech. This result is interpreted as suggesting that pre-linguistic infants are able to identify the clauses in motherese but not in adult-directed speech [153]. If we accept these two interpretations, we might conclude that without motherese as input, infants would not be able to detect clause boundaries and therefore could not learn a language.

However, there is strong evidence against such critical role that motherese would play in language acquisition: it is reported that in a number of cultures, for example, in Papua New Guinea [307, 308], Samoa [272, 309] and even among some African Americans [150], infant-directed speech is not available to pre-linguistic infants because adults simply do not talk to the infants before they have learned to speak. The over-estimation of the importance of infant-directed speech excludes the possibility that human babies in a culture with little motherese available still can learn a language purely from the adult-directed speech taking place around them.

What we can conclude, in consideration of the positive and negative evidence discussed above, about the role that infant-directed speech may play in language acquisition is as follows. First of all, there is evidence that infant-directed speech, if available, does facilitate language acquisition at an early stage. The prosodic packaging of some salient speech units like utterances and phrases appears to have a beneficial effect on leading
the learning infants to realise the existence of such linguistic structures in fluent speech. However, no matter how important the infant-directed speech may be to some stage of language development, we still do not have any grounds to state that language acquisition has to rest on the availability of speech input of this special style. Although speech of this style does take place in many cultures and does appear to have beneficial effect on the early stage of language acquisition, the fact that infants in some other cultures with little infant-directed speech available do succeed in learning their language from adult-directed speech indicates that their language faculties have enough capability to deal with natural language speech input of any style or characteristics for the purpose of acquiring a language.

Moreover, we have not had any clear idea about what causes an infant to prefer to listen to infant-directed speech. Is it an innate bias, or a preference due to prenatal exposure to human speech? If it is an innate bias, a new-born baby should equally prefer baby talk in any language. Why does it prefer only baby talk in its own mother tongue? If this preference is not an innate bias, it must be something that has to do with prenatal speech experience, that is, it is a result of learning (or memorising). Also, we still do not know what effect on later language acquisition would result from lacking such a listening preference. Does an infant have any acquisition barriers or special difficulties in any later stage of language acquisition if it is given no chance to build up such a preference? There seems to be little evidence supporting an affirmative answer to this question.

Furthermore, even in the speech environment with infant-directed speech available, it is still unlikely that the language-learning infants hear only infant-directed speech and no adult-directed speech – notice that there is so much adult-directed speech surrounding them spoken by their parents and other elders. We still need to obtain a clear idea of how much an infant learns from infant-directed speech and how much from adult-directed speech. What is very likely is that the infants hear a greater volume of adult-directed speech than infant-directed speech, because adults speak faster to each other and the time they speak in the baby talk style is relatively short. More importantly, when the language-learning infants can talk, they do not talk first in the infant-directed style and then move to the adult-directed style. Rather, they seem to straightforwardly go into the adult-directed style: even when they are producing baby-words, they tend to use a normal prosody rather than an exaggerated one, with almost no onomatopoeia and interjections (e.g., oh and uh) – although they always hear speech that has exaggerated
CHAPTER 2. LEXICAL ACQUISITION - A COGNITIVE PERSPECTIVE

prosodic characteristics and is full of onomatopoeia and interjections. Also, the infants learn many lexical items and syntactic patterns that seldom or even never appear in infant-directed speech. These observations indicate that the infants are quite clear in their minds that the target language for them to learn is the adults’ speech, not the baby talk.

It is argued in [83], based on the observation that many of those characteristics of child-directed speech, including the high frequency of phonological elisions and assimilations, are exactly the characteristics of adult-directed spontaneous speech in contrast to rehearsed speech heard on the radio and television, read speech in news broadcast and in lectures, that the infant-directed spontaneous speech should be considered to lie on a general continuum at the same end with the adult-directed spontaneous speech, with the former in a more extreme position. Although the phonological elisions and assimilations in spontaneous speech make the segmentation problem even harder for children, it is far from true that the infants can learn better from rehearsed or read speech than from spontaneous speech.

In short, infant-directed speech is no doubt a beneficial input to language-learning infants in many cultures. In particular, its prosodic characteristics such as the exaggerated pitch contours (e.g., pitch declinations) and lengthy duration of vowels at utterance ends and (prosodic) phrase ends appear to play the role of signaling the boundaries of some salient linguistic structures to the learning infants. However, the infant-directed speech is not the only input to language acquisition. We should not neglect the role that adult-directed speech may play in language acquisition: Even without infant-directed speech, human infants have no problem acquiring a language from the adult-directed speech around them. Thus, a better understanding is that both infant-directed and adult-directed speech are at the same end of spontaneous speech on a general continuum and that language acquisition, including lexical learning, does not entirely rely on the availability of infant-directed speech. It is reasonable that, ultimately, children learn more from normal adult-directed speech than from the special infant-directed speech, although the latter seems to play a more critical role in bootstrapping the initial stage of language acquisition.

2.4 Pre-linguistic Infants’ Speech Perception

Human infants are born with many amazing perceptual abilities that enable and facilitate language learning. It is argued that infants are born with a nascent structure-seeking
mechanism to discover particular sized units with particular distributional patterns in the input, be it spoken or signed, and this seeking procedure is believed to be guided by some innately specified structural constraints [282] (e.g., children’s early lexical use is already constrained along the bounds of word types, like object names, property words, events words). When an infant is born, this nascent mechanism is sensitive to phonological patterns (rhythmic, temporal and hierarchical) that are common in all languages [123], and is particularly sensitive to the syllable-like structures (in terms of their size and distributional patterns) in the input (spoken or signed) [283].

However, this nascent structure-seeking mechanism for language learning would be unable to work if the atomic elements, i.e., speech sounds, in the speech input could not be correctly detected by the pre-linguistic infants. While thinking about how infants start to acquire lexical items from the speech input with out any supervision, the first question we have to ask is how strong is the pre-linguistic infants’ ability to discriminate among speech sounds in terms of their phonological characteristics (or features) when they start to learn a language.

Just as humans are born to see, they are born to hear. An infant’s inner ear is physiologically fully-grown at birth. It is reasonable to assume that infants can hear even in the womb. There is evidence that even before birth the foetus’ auditory system is functioning. It is observed that the foetus in the womb responds to external sounds [201], and also that newborns show preference in their first day for their mother’s voice over an unfamiliar female’s voice [101]. This preference cannot be explained if the newborns have not had any auditory experience. Two stronger pieces of evidence supporting the idea that infants have started speech perception, and therefore language learning, before their birth are the demonstrations that the newborns can discriminate a passage loudly read by their mothers during the last six weeks of their pregnancy from an unfamiliar passage, even when it is read by a woman other than their mothers [103], and that they can distinguish utterances in their mothers’ language (e.g., French) from utterances in a foreign language (e.g., Russian) [244]. The prosodic contours in their mothers’ speech are observed to play an important role in enabling the infants to recognise utterances in their mother tongues. It is reported in [244] that even when the speech samples were filtered in a way to only keep the prosodic (in particular, rhythmic) information, the results turned out to be similar.

This listening preference must be shaped by prenatal speech exposure in the uterus. There are intra-uterine recordings to reveal the fact that the low-frequency components
CHAPTER 2. LEXICAL ACQUISITION - A COGNITIVE PERSPECTIVE

of sounds from the outside world, in particular the maternal speech with its distinctive rhythmic features, are audible in the uterus. It is found that late-term foetuses, in the last three months of gestation, respond to external speech stimuli in a rather consistent way [210, 209, 214]. It is reported in [213] that repeatedly presenting a pair of French syllables ([ba] and [bi], or [bi] and [ba]), uttered by a female, to foetuses of 36-40 weeks every 3.5 seconds elicited a deceleration of heart rate, and when the order of the two syllables was reversed, the same deceleration reliably showed up after 16 presentations. This fact was interpreted as an indication that the foetuses could discriminate between the two types of stimulus. Near-term foetus’ heart rate response to acoustic stimulation were further studied using the short sentence “Dick a du bon thé” (“Dick has some good tea”) in a male and a female voice [214]. It was observed that 77% and 66% of the subjects reacted, respectively, to the male and female voices with a decelerative heart rate change within the first 10 seconds of the stimulation. When the initial voice changed, 69% of the subjects showed a heart rate deceleration, whereas 43% of the control subjects, who kept hearing the initial voice, showed a slight acceleration [215]. These findings indicate that the near-term foetuses can differentiate between the two voices and that the foetal auditory system has matured to such a degree that it can detect an acoustic change in human speech based on a rather small speech sample. It is also reported in [102] that mothers’ loud recitation of one of two selected rhymes to their 37-week old foetuses once a day for a period of four weeks could result in the familiar rhyme causing the foetuses’ heart rate to decrease consistently when they were listening to the familiar rhyme, whereas the unfamiliar rhyme did not have such an effect.

Infants’ perceptual ability develops rapidly after birth. Within a few weeks of birth, they are able to make distinctions between human voice and other sounds, and in about two months, they can detect the difference between angry and friendly voice qualities [225]. Experiments using the nonnutritive sucking technique (also known as the high-amplitude sucking (HAS) technique, as in [153]), and the head-turning technique in the past three decades have greatly deepened our understanding of pre-linguistic infants’, in particular, newborns’, perceptual abilities related to speech. These techniques show that in the first month infants can distinguish voiced from unvoiced sounds [115], and can discriminate vowel contrasts (e.g., /i/ vs. /a/) [331] and consonant contrasts (e.g., /p/ vs. /b/) [200]. In two months they are able to tell apart different intonation contours and places of articulation [258, 72].

Initially, infants’ speech discrimination abilities are language-general, rather than
CHAPTER 2. LEXICAL ACQUISITION – A COGNITIVE PERSPECTIVE

specific to any language. The infants can discriminate contrasts outside the ambient language. It is reported that English babies can discriminate consonant contrasts that exist in Hindi [347] and vowel contrasts that exist in French [332], but not in English. A comprehensive review of pre-linguistic infants’ perceptual abilities for sound contrasts from many languages is given in [141] and in [172].

However, although the pre-linguistic infants’ hearing is so sharp that speech sound contrasts in any language can be detected, they do not perceive speech sounds as distinct individual sounds. Rather, they perform categorial perceptions while perceiving speech sounds: speech sounds with acoustic differences within a certain range are recognised as a phoneme. From the acoustic perspective, human speech signals are sound waves transferred through air pressure changes from a speaker’s vocal organs to a listener’s ears. Human ears – more precisely, human brains, which receive speech signals from the ears – do not interpret sound waves of human speech as waves, but as segments or units that bear phonological significance in a language. These basic units, or sounds, are known as phones or allophones. In human speech perception, phones are grouped into phonemes that are phonologically distinct from each other. In this sense, humans as natural language speakers only “hear”, or perceive, phonemes, rather than phones, in general. Every phone must be heard as a phonemic category. People cannot distinguish sounds with a 20-msec difference of voice onset time (VOT) within the same phonemic category, e.g., variant instances of /p/, but can detect a 20-msec difference across a phoneme boundary – for example, as demonstrated in [357], a sound in the /b/ and /p/ categories (whose phoneme boundary is at about 25 msec VOT) will be heard as a /b/ if its VOT is shorter than the boundary point (i.e., somewhere near 25 msec), otherwise, it will be heard as a /p/.

It is showed that infants’ categorial speech perception is similar to that of adults [115], although their phonology may not be identical to that of the adults. The more an infant’s speech perception has been attuned to the adults’ phonology, the less sensitive is its hearing to the sound contrasts not existing in its mother tongue. It is showed in [348] that English-learning infants 6 to 8 months old could discriminate consonant contrasts that exist in Hindi and Inslekepmx (a language spoken by the native Salish of British Columbia) but not in English, but very few English-learning infants of 10 to 12 months old could do so.

This categorial perception is not unique to speech. Some non-speech sounds, such as noise-buzz sequences, are also found to be perceived categorically [248]. And, the cat-
CHAPTER 2. LEXICAL ACQUISITION – A COGNITIVE PERSPECTIVE

...egorical perception of sounds is not specific to humans; it is reported that chinchillas, a kind of rodent, can also sense the phoneme boundary effect between /p/ and /b/ [202]. However, although the chinchilla’s aural system is observed to function in a similar way as human’s one does, the chinchillas (and many other mammal species which have a similar mechanism for aural perception) do not develop any spoken language relying on this categorical perception mechanism. Thus, it is concluded that the categorical perception of sounds, as reflected in the phoneme boundary effect, is a property of the mammalian aural system that can be utilised to develop languages as signal systems for communication, rather than a unique linguistic property of human’s auditory perception [201, 249, 153]. Nevertheless, it reflects the extent to which the human species has prepared, through evolution, for language acquisition. Without this particular endowment, humans would not have had any spoken language like those in use today, let alone language acquisition.

Although humans are born to have, in general, the ability to discriminate phonetic contrasts in any language (no matter how subtle these contrasts are) and the ability to perform categorical perception of speech sounds (in particular, the consonants), they do not seem to perceive speech phone by phone, but syllable by syllable. There is evidence that syllables are the basic units of mental representation of speech sounds and are, therefore, the effective units in the young infants’ speech perception. It is showed in [177] that while presenting randomly ordered sequences of syllables to 2-month-olds, the infants increase their response to new syllables, no matter how subtle the change of from the old syllables to the new ones, but the change of the number of the phonemes does not increase the response. A perhaps even stronger piece of evidence supporting the view that syllables are young infants’ effective mental representations for speech sounds is the study [19] on how 4-day-old infants categorise multi-syllabic utterances: the babies had no difficulty distinguishing sound sequences of difference numbers of syllables, e.g., two-syllable ones (e.g., rifo) and three-syllable ones (e.g., kesopa), but could not detect the difference between two-syllable sequences with four phonemes (e.g., rifo) and two-syllable sequences with six phonemes (e.g., treklu). These studies suggest that although young infants are highly sensitive to phonetic contrasts and to phoneme boundaries, they tend to, interestingly, nevertheless, perceive a speech stream syllable by syllable, instead of phoneme by phoneme.

Since syllables are found to be the effective units in speech representation, a speech stream can be understood as a sequence of individual syllables. Naturally-occurring ut-
CHAPTER 2. LEXICAL ACQUISITION - A COGNITIVE PERSPECTIVE

ences produced by adults are sequences of continuing syllables, rather than discrete sequences of isolated syllables. Therefore, pre-linguistic infants’ ability to identify individual syllables in a stream of on-going speech is critical to their language development, in particular, in the initial stage. An experiment in [143] using the head turning technique showed that pre-linguistic infants of 6.5 months old can discriminate individual syllables like [ha] and [du] not only in isolation but also in combination with other syllables. The experiment also revealed that when the target syllables were embedded in more complex contexts, the discrimination of them appeared more difficult. For example, the correct rate of the infants’ discrimination of [kokoD] from [kokoA], where the target syllables are embedded in a redundant sequence, is 75%, but the correct rate for the recognition of target syllables in mixed sequences such as [kotiba] and [kotidu] goes down to 67%. Another study [185] shows that prosodic features, such as stress on target syllables, e.g., [kotiba] and [kotidu], can help young infants to do the discrimination easier than on the same multi-syllabic sequences with an even intonation.

However, the recognition of syllables as the basic units in infants’ speech representation does not indicate that there is absolutely no segmental representation for any speech sounds. It seems necessary to have some segmental representation in many languages. For example, in English, we have a few morphemes of certain syntactic importance whose sounds do not make a syllable due to the lack of a vowel, e.g., ’s as the abbreviated form for the third person singular copula is (as in it’s here and that’s it), -s as the suffix for plural nouns (as in trees) and -ed as the suffix for past tense verbs in the cases of not preceded by a vowel (as in trained). It is reasonable that morphemes of this kind in a language may need to have a segmental rather than a syllabic representation. From this perspective, a speech stream can be represented as a continuous sequence of phonemes, and a syllable in the speech can, accordingly, thought of as a number of phonemes in a structure, e.g., in a CVC structure.

In summary, human infants are born with a remarkable sensitivity to sound contrasts existing in natural language speech and with a special ability to perform categorical perception of speech sounds, although they are born with very little speech experience, if not completely none. A language-learning infant’s speech perception is gradually attuned to the phonology in its ambient language as it is exposed to more and more speech data. In addition, there is evidence that syllables, instead of phones, are the representational and perceptual units perceived in the speech stream, i.e., syllables are the basic chunks of sound that the infants perceive – of course this observation does not
deny the fact that the young infants can distinguish phonemes within syllables and a syllable is actually a structure of phonemes. This also provides a basis to explain why so many prosodic features over syllables, such as stress and intonation contours, play a critical role to initiate and facilitate the pre-linguistic infants’ lexical acquisition. In next section we will focus on the prosodic and other cues that facilitate speech segmentation by the infants for the purpose of lexical acquisition.

2.5 Speech Segmentation and Word Discovery

Lexical acquisition involves at least two aspects; one is the identification (or determination) of word forms in spoken utterances and the other is the appropriate association of lexical properties (e.g., meaning or conceptual information, syntactic properties like part-of-speech and agreement) to each word form.

There is a bootstrapping problem in infants’ acquisition of lexical forms. It is reasonable to assume that they have an empty lexicon at the beginning of language acquisition. However, how do they get the learning started? In order to acquire words one after another to build up a lexicon, the infants must have the ability, without knowing any particular words at the very beginning, to recognise word forms in the speech stream. To our knowledge, however, one must know the word forms before one can recognise the words in the speech. So the problem is: how can infants, who know no words, extract words from the speech to which they are exposed, and put them in the lexicon one by one so as to enlarge lexicon day after day? There seems to be a chicken-and-egg problem for preverbal infants to resolve: they must be able to segment fluent speech into words in order to develop a lexicon from scratch, but they must know a certain number of words before they can perform the segmentation! How can an infant get around this problem and develop its lexicon? That is why lexical acquisition is so interesting and perplexing. It is not as trivial a problem as it seems to be – just pick out words from the speech input and put them into the lexicon. It is not as simple as this. It involves a deadlock problem: if you know no words, you can’t pick any words from the speech input; if you can’t pick any words, you remain knowing no words.

It seems unlikely that we can find out whether the chicken or the egg appears first. A possible way out of this dilemma is to assume that the infants have some means or strategies (e.g., the utterance-as-word strategy) to segment speech into words or word-like units for the purpose of developing their own lexicon, with the aid of various kinds of cues in the speech stream, e.g., pauses, stresses, that they are born sensitive
to or that they can somehow learn to detect after birth. This early lexicon may be
different from the adults' one at the infants' early age; for example, some collocations
of frequently co-occurring words may be taken as individual words in a child’s lexicon.
But this lexicon will gradually converge to the adults' lexicon along with the growth of
the infants’ language experience and competence. This convergence is a very interesting
process of learning that is worth more exploration in the field of language acquisition.

In this section, we will take a close look into what cues in adults’ speech can facilitate
infants’ lexical learning, and what strategies the infants can exploit to develop their
lexicon into one as close to the adults’ lexicon as possible. We first start with the speech
segmentation problem in the next subsection.

2.5.1 Speech Segmentation
Understanding spoken utterances involves a process of identifying discrete words from
the speech stream. Only after individual words in an utterance are properly recognised
can the structure of the utterance be analysed and its meaning be interpreted. Although
it seems effortless for adult listeners to carry out the word recognition task during
speech comprehension, it is by no means a trivial task. In addition to having to cope
with troubles caused by some undesirable characteristics of the speaker’s voice (such as
dynamically variant speaking rate, accent, co-articulation of adjacent words, etc.) and
background (speech) noise while listening to the speech signals (which can be understood
as a sequence of phonemes, or syllables), the listener also has to map this sequence of
continuous speech signals onto a sequence of lexical items from the listener’s own lexicon,
which is usually of tens of thousands of words. That is, the listener has to segment the
speech input into fragments such that each fragment matches an existing word in the
lexicon. The situation could be more complicated if the speech input involves any
unknown new word(s).

If formulated as a hypothesis selection problem in terms of some objective function
\( f(\cdots) \), the speech segmentation problem can be thought of as selecting a sequence of
words from the lexicon to cover exactly the input speech signal \( S \) such that the objective
function on the words can be maximised. It can be expressed by the equation below:

\[
Ws(S) = \arg \max_{w_1 \circ w_2 \cdots \circ w_n = S} f(w_1, \cdots, w_n)
\]

(2.1)

where \( Ws(S) \) denotes the resulting sequence of words from the input \( S \) and \( \circ \) is a
concatenation operation. If the object function is to evaluate the probability of the
word sequence, it can be rewritten as (2.2), following Bayes rule.

\[
f(w_1, \ldots, w_n) = \prod_{i=1}^{n} p(w_i|w_1 \cdots w_{i-1})
\] (2.2)

To compute this objective function, a probability distribution over words given a preceding context in a language must be given or estimated somehow, for example, based on individual word sequences’ relative frequencies.

Notice, however, that a constraint on on-line lexical processing for speech comprehension that is not taken into consideration in (2.1) is that the determination of individual words is to be done one by one in order: once a word is determined, the listener moves on to work on the next word. In general, backtracking to any previous word is not permissible in real time speech processing by human listeners.

The difficulties in speech segmentation and word recognition lie in the fact that word boundaries are not explicitly marked in the speech signal: not only are there no explicit markers (e.g., pauses) about where a word begins and ends, there are also no cues that are fully reliable, as noted in [218, 261] – although there exist many types of cues in continuous speech to facilitate the location of word boundaries, there are always many exceptional cases where the cues do not work right. Useful cues include lengthening of word-initial and -final syllables, allophonic cues (e.g., aspiration of word-initial stop consonants), phonotactic cues – disallowable sequences (esp., pairs) of consonants in words (or syllables) (e.g., [mr] in English), and many others. Importantly, although languages differ from each other also in terms of their rhythm and prosody, the particular metrical structure of a language can be made use of to facilitate the segmentation. For example, the syllable is the basic metrical unit in French and Spanish, and the mora is the basic metrical unit in Japanese [274, 90]. There is evidence that the native speakers of these languages make use of the syllabic and moraic information, respectively, to perform speech segmentation [87, 88, 275, 91].

In English and Dutch, the distinctive rhythmic characteristics of the strong and weak syllables are utilised by native speakers to do speech segmentation [89, 84, 83, 243, 269, 342, 343]. There are observations that more than 90% of content words in English start with a strong syllable, about 75% of the strong syllables are at word onsets in English speech [85] and that about 85% of Dutch words have a strong syllable at the word onset [310, 342]. It is reasonable to infer, based on these findings, that the native speakers of these languages tend to speculate a word onset at a strong syllable. Accordingly, the metrical segmentation strategy (MSS) is formulated to characterise the native speakers’
bias in pre-lexical processing of speech input [89]. There is evidence from a number of investigations by Cutler and co-workers that English adults do apply the MSS to predict word onsets with the occurrence of strong (or stressed) syllables in speech processing [82, 84, 83].

In addition to the tendency of using the MSS, adults’ segmentation of fluent speech are also full of other activities such as activation of candidate words at all points along the speech input and competition among activated candidates. For example, when can is heard, many words beginning with the syllable can, such as can, cancel, candle, canteen, etc., are activated. When more speech signals are received, some candidates will be ruled out if they are inconsistent with the new signals, and some continue to survive until the end of the utterance. The competition takes place not only among individual candidate hypotheses activated at the same point, but, more importantly, also among difference parses, each being a sequence of activated words, over the same speech fragment (e.g., a phrase) or the entire utterance. For example, when a speech fragment like met a fourth time is heard with some background noise (because of which th can be confused with f), theoretically, another possible parse over this fragment could be metaphor f time (borrowed from [270]). If word embedding is also considered, the case can be more complicated. How do human listeners resolve this kind of ambiguity in pre-lexical processing? A constraint called the possible word constraint (PWC) is proposed in [270] to model human subjects’ decision making in such cases: impossible words are disfavoured. That is, a parse with all chunks being possible words in the listener’s lexicon is preferred over a parse with some chunks being impossible (or unknown) words. As far as the preceding example is concerned, human subjects will tend to follow the PWC to choose met a fourth time, because in the other choice the chunk f is an impossible word in English. Also, evidence is given in [270] that it is easier, as shown by response time and correct rate in experiments, for human subjects to segment vaffapple into vaff apple than fapp into f apple, because vaff, which contains a vowel, is possibly a word in English whereas f is known for sure not to be a word – it is common sense that every word in English has a vowel. Even worse, an f standing alone does not even make a syllable, let alone a word.

There are a few computational models to simulate adult speech segmentation, for example, TRACE [239] and the Shortlist [271], implemented in neural networks with an emphasis on modelling the competition between candidate words. The Shortlist was extended later in [270] to incorporate the PWC. Here, we are not going into the details
of these models beyond the scope of our research on lexical learning.

As shown in the brief review above, speech segmentation and lexical recognition performed on continuous speech by adult listeners, who can be thought of as equipped with a huge (if not almost complete) lexicon, is by no means simple or trivial. Many useful cues, none of which are entirely reliable though, are utilised, and a number of complicated cognitive processes, e.g., candidate word activation and competition, are involved. The listeners have some language-specific strategies, e.g., the MSS for English and Dutch, to facilitate the processing. Their cognitive behaviours in pre-lexical processing also appear to observe certain constraints, e.g., the PWC.

With regard to the complexity in adult speech segmentation, there is reason to believe that speech segmentation and word discovery by pre-linguistic infants, who have an empty lexicon, is even more difficult and complicated. How do they come to know there are words in their language? What cues can they make use of to discover words in fluent spontaneous speech? Are there any specific strategies they can exploit?

### 2.5.2 Cues in Speech for Word Discovery

Although how young infants acquire word forms from a continuous speech stream is recognised as a central task in lexical acquisition, it is not yet clear so far in our psycholinguistic studies how and when the infants start to be aware of the existence of words in their language and then attempt to segment fluent speech into individual words. What we are clear about are, at least, the following facts: first, all language have words, of which individual utterances are composed - thus, to understand an utterance one has to decompose the utterance into words and then retrieve each word’s meaning from a lexicon in the mind; second, adult speakers do not, and also appear unable to, explicitly tell the infant language learners which sound sequences are words, in particular, where a word starts and ends, even in the situation of teaching them new words.

There seems to be two possible ways to inform lexical-learning infants of word boundaries: the first is to tell them, implicitly, by speaking in isolated words, isolated by significantly long pauses of silence; the second is to tell them explicitly by speech. Across all languages in the world, adult speakers do not speak in isolated words, even in the situation of speaking to infants in the infant-directed speech style. If they were to speak in isolated words, whether their offspring could learn their language would be seriously in doubt, because such speech loses many prosodic, rhythmic and other characteristics of frequency-and-volume-change (e.g., pitch contours and intonations) in the continuous
CHAPTER 2. LEXICAL ACQUISITION - A COGNITIVE PERSPECTIVE

speech of their language that are known to be very useful in bootstrapping the infants’
sensitivity to many speech units such as clauses and phrases in the language – these
units bear special prosodic demarcations to which the very young infants are sensitive
[172]. If one attempts to directly tell the infants about word boundaries, one would
have to speak to them in speech and they would have to segment the speech into words
for understanding – in order to achieve this, however, they have to learn to do speech
segmentation first. Thus, a deadlock problem arises. It appears that there is no effec-
tive way to teach preverbal infants to do speech segmentation for the purpose of lexical
learning. They have to get around the deadlock problem somehow by themselves.

The greatest difference in speech segmentation between adults and the very young
language-learning infants lies in the resources they can use: in addition to the various
types of cues to word boundaries, the most important resource available for adults is an
existing lexicon that can be thought of as containing almost all words in the language
– speech segmentation thus becomes an issue of decomposing a received utterance into
a number of existing words in the lexicon that best match the input; whereas infant
language learners start with an empty lexicon and also have to segment the speech
input into words or word-like units. The infants seem to face a much more difficult task:
they have to attempt a similar segmentation task with no comparable resources at all
at the early stage of lexical learning – they may not even know there exist words in
their mother tongue – and also they have to infer some cues to “word” boundaries for
later use. What leads them to become aware of the existence of words in general and to
extract individual words from fluent speech in particular?

A possible answer is that there may not be any particular thing(s) in speech signals
that lead to this kind of awareness, but only that the human infants have an innate
mechanism to derive a least-effort representation for the input data they encounter,
and words happen to be the pieces of building blocks in such a representation. When
the infants have more and more such pieces to a certain level, we call these entities
words. The evolution of a lexicon can be rather dynamic, in that some old pieces
(e.g., the multiple word collocations that were once recognised as individual words)
may be dropped and some new pieces must be added, in order to achieve a least-effort
representation for more and more new data. This dynamic lexicon finally comes to
stabilise, relatively though, at a state in which it would not need any radical change
to reach the least-effort representation while more and more upcoming input has either
already been seen before or can be decomposed into fragments of the seen data. This
stabilisation process seems to be a plausible process for an infant’s lexicon to converge to the adult lexicon.

The hypothesis that human infants have an innate mechanism for language learning to derive the least-effort representation for language data underlies the main theme of exploration in this thesis. In later chapters we will move on to define the least-effort representation for a given set of data, following the minimal description length (MDL) principle [294, 297] – a popular approach to utilising an approximation of Kolmogorov complexity [322, 198, 219] to do inductive inference – and also formulate computer learning algorithms to realise this lexical learning strategy for the purpose of examining what performance this strategy can achieve on naturally-occurring language data. In this section, we will focus on the cognitive aspects of young infants’ lexical learning. In particular, we review psycholinguistic studies on how infants exploit various cues to facilitate their inference and determination of possible word boundaries. As summarised in a recent review paper [174] by Jusczyk, the major cues used by infants in speech segmentation for extracting words from fluent speech to develop their lexicons include prosodic cues, allophonic cues, phonotactic cues and statistical cues (or distributional regularities). A large volume of background information on pre-linguistic infants’ perceptual sensitivity to these cues can be found in Jusczyk’s remarkable monograph [172] with thorough discussions. In the sections below, we will discuss the utility of these cues in young infants’ lexical acquisition.

Prosodic Cues

Human infants are known to have special capacities for speech perception from a very young age – some are innately endowed and some are acquired after birth. In particular, they are remarkably sensitive to the prosodic information, for example, the rhythm, in the fluent speech of their mother tongue. No doubt this perceptual sensitivity has to do with the prenatal exposure to speech sounds in the intra-uterine environment – this environment functions to protect the foetus from exogenous sounds by blocking sounds of high frequency (above 250 Hz) but letting the low frequency sounds be transmitted to the foetus’ inner ear with little reduction of sound pressure [1]. Based on their prenatal experience of hearing, new-born babies are able to discriminate their mother tongue from other languages, based on their detection of the rhythmic distinction across languages [244, 262].

There appear to be two trends in infants’ development of their awareness of linguis-
tic units of various sizes, ultimately towards words. One trend develops from smaller to larger units, that is, the infants first learn the smallest discriminative speech units, i.e., phonemes, and their categorisation, resulting in the infants’ awareness of phonemes, in their native language. Afterward, they pick up the structure of syllables, usually consisting of several phonemes in a hierarchical structure, and then, how syllables combine with each other to form words. Although these intra-word speech units carry certain suprasegmental information (e.g., tones over syllables), in general they do not seem to contribute, individually, much useful information about word boundaries. Allophonic cues are indeed an exception (see the next section for discussion).

Another trend is that the infants detect the existence of clauses (or utterances) as linguistic units in speech, and then phrase, and then words. The temporal progression of this trend is rather clear. It was first found in [152], using the headturn preference procedure with a pause insertion technique, that 7- to 10-month-olds demonstrated a preference for whole clauses over interrupted ones (i.e., ones with a pause inserted in the middle), although they were too young to understand the meaning of each clause. This result was interpreted as due to the infants’ sensitivity to the prosodic demarcation of linguistic units such as clauses. Later, infants as young as only 4.5 months old were reported to have a similar preference for clausal prosody [170]. More strikingly, evidence is further given in [229] that 2-month-olds have a certain capacity to use clause prosody to organise and remember phonetic properties of words, e.g., rat versus cat. The infants appeared to remember speech information in a word better in natural clause prosody than in an isolated word list, as indicated by their responses measured in terms of their high-amplitude sucking (HAS) rates. It is also shown that during the age between 6 and 9 months, infants are developing their sensitivity to sub-clausal units such as prosodic phrases, as reflected in the fact that the 9-month-olds, but not the 6-month-olds, demonstrated a preference for listening to passages with pauses inserted at phrase boundaries rather than to passages with pauses inserted within phrases [179, 135, 172]. By the age of 11 months, but not before 9 months, infants appear to have developed their sensitivity to word boundaries in fluent speech to such a degree that is testable by the pause insertion technique [259].

Also, it is reported in [313], interestingly, that even new-born babies of 1- to 3-days old were observed to be able to discriminate between lexical and grammatical words in spontaneous infant-directed speech, relying on their perceptual sensitivity to constellations of acoustic cues, including some salient prosodic characteristics of the
words, for example, grammatical words usually have a shorter vowel duration, weaker amplitude, simpler syllabic structure, and so forth.

From the above evidence, as a whole, we can see that the preverbal infants’ perceptual ability for prosodic information may be in place from a very young age, if not from the birth, and can be utilised to do speech segmentation and word discovery (although whether they have a sharp enough sensitivity to various kinds of prosodic cues so as to make use of them at the age around 7.5 months old when they start to recognise words in fluent speech [175, 174] still remains a question for the moment – see the discussion below). But what kinds of prosodic information (or cues) do the infants use to discover words in fluent speech? How effective are such cues and what problems may be caused by (over)using these cues? When problems are caused by one strategy of exploiting cues, how do the infants come up with a new strategy to utilise more information to overcome the deficiency of the old strategy?

According to Jusczyk [174], English speaking infants have a speech segmentation competence like a native adult around their second birthday, in terms of speed and accuracy. Many speech segmentation strategies are developed during the second half of the first year, and the skills are further elaborated in the second year. Utilising prosodic cues is just one of the strategies in the initial phase of lexical learning.

The term prosody denotes, in general, the suprasegmental attributes of speech sound, including pitch, stress, accent, duration, tone, intonation, rhythm, pause, etc., usually resulting from certain patterns of change of sound intensity and frequency. The recent studies on how prosodic cues facilitate pre-linguistic infants’ lexical learning focus on the rhythmic (or metrical) structures of a few languages, e.g., English. Linguistically, rhythm refers to the harmonic succession of sounds, in particular, certain regular periodicity of some sound attribute(s), that reflects the musical flow of speech in a language. In English, for example, the rhythm is the patterns of alternation between strong and weak syllables, where a strong syllable (whose vowel is non-reduced) is known as a stress. A stress foot in English consists of a stressed syllable and a following unstressed syllable (which contains a reduced vowel), if present. The typical English stress feet generally have a trochaic stress pattern, e.g., table and parent, as opposed to an iambic stress pattern, each being a weak-strong syllable pair, e.g., guitar and device.

In order to be able to make use of prosodic cues in speech segmentation, infants must first have a certain sensitivity to the cues. Inspired by the findings in [85] that most strong syllables are at the word onset in spoken English and hence the predominant
CHAPTER 2. LEXICAL ACQUISITION – A COGNITIVE PERSPECTIVE

stress patterns in English are trochaic, Jusczyk and colleagues investigated whether American infants are sensitive to this typical prosodic property of English words [176]. They presented a list of trochaic or iambic bisyllabic words to 6- and 9-month-old infant subjects in experiments, and found that the 9-month-olds listened significantly longer to the trochaic than to the iambic words, whereas the 6-month-olds did not show any preference. In addition to [176], other subsequent studies also give indications that it is in the period of 6 to 9 months of age that English-speaking infants have developed their sensitivity to the predominant word stress patterns in their mother tongue [256, 255].

The infants’ sensitivity to this rhythmic characteristic of English words appears to provide them with a basis for applying the metrical segmentation strategy (MSS) to start speech segmentation for word discovery. A number of interesting investigations were conducted by Jusczyk and co-workers [154, 268, 172, 173, 181] on how English-learning infants of 7.5 months make use of the predominant stress pattern in the language to segment words from fluent speech, following the MSS. They first tested whether the infants could detect words in the predominant strong-weak stress pattern in fluent English speech. In the experiment, the infants were familiarised with a pair of target words (e.g., “hamlet” and “kingdom”, or “doctor” and “candle”), and then presented with four test passages of a few sentences – two of these passages each carried a target word in each sentence and the other two did not. The experimental results showed that the 7-month-old infants listened longer to the passages containing the target words than the other passages, suggesting that they detected the familiarised strong-weak bisyllabic words in the passages.

However, there is another possibility in this experiment, that is, the infants might simply match the initial strong syllables of the target words, e.g., “doc” in “doctor” and “can” in “candle”, to those in the passages. To eliminate this possibility, the experiment was changed to familiarise the infants with the first syllables of the target words and then present them with the test passages. This time, however, the infants’ listening time showed no preference for the passages with the target words, suggesting that in the previous experiment the infants did detect the trochaic target words as a whole in the passages by whole-word matching, instead of matching the initial strong syllables of the words.

We know that the infants of this age follow the MSS to recognise trochaic words. How did they deal with words of the opposite stress pattern, namely, the weak-strong bisyllabic words? Jusczyk and co-workers repeated the first experiment with a change:
they familiarised the infants with a couple of iambic target words such as “beret” and “device” or “surprise” and “guitar”, and then presented to them four passages, two of which each contained a target word in each sentence and the other two, known as control passages, did not. This time the infants did not show any listening preference, suggesting that they did not detect any target words in the test passages.

What happened? The researchers guessed that since the infants were supposed to follow the MSS, it was possible that they inserted a word boundary at the middle of the weak-strong words that they heard in the familiarisation. To test his possibility, two more experiments were further conducted with iambic target words: the test passages were designed in a way such that each target and control word was followed by a particular unstressed word, e.g., “guitar is” and “device to”, to see how the infants would react to them. In one experiment, the subjects were familiarised with the iambic target words and then listened to the test passages – the result: no listening preference was detected. In the other experiment, the infants were familiarised with bisyllabic non-words of trochaic stress pattern such as “tar is” and “vice to”, and then presented with the same test passages – this time, the subjects did listen significantly longer to the test passages with the target trochaic non-words!

These experiments all together demonstrate that English-learning infants do follow the MSS to identify strong syllables as word onsets at the time when they start to do speech segmentation at around 7.5 months.

An experiment reported in [114] also gives further evidence for 9 month old infants’ possible use of the trochaic stress pattern to segment English speech input into word-level chunks, because the subjects in the experiment, after hearing various speech inputs in the familiarisation phase, could distinguish the trochaic syllable pairs embedded in the four-syllable input from novel trochaic pairs in the test phase, but did not make this distinction for the iambic targets and novel iambic distractors – this result suggests that the infant subjects could recognise the previously heard trochaic bisyllabic sequences in the speech input as familiar lexical units that stand out. Notice, however, that the four-syllable speech input is not naturally-occurring speech data from fluent speech.

However, entirely relying on the metrical segmentation strategy to do speech segmentation does lead to mistaken results – the infants will always miss the iambic bisyllabic words. It has not been clear what kind of strategy the infants would use to remedy the problems caused by the MSS strategy, although it is known that the infants at 10.5 months of age have gained the ability to recognise iambic words in fluent speech [174].
CHAPTER 2. LEXICAL ACQUISITION - A COGNITIVE PERSPECTIVE

It is reasonable to guess that infant of this age may resort to a constellation of available cues. Also, what strategy an infant would exploit to integrate multiple cues is also unclear, although there is evidence that, for 9-month-old infants, when both prosodic and phonotactic cues are available but conflict with each other, prosody overrides phonotactics [238].

Specifically for rhythmic cues, we have a few questions to ask: Where are they from? How do the English-learning infants get sensitivity to them and acquire the sense (or knowledge) that the stressed syllables are more likely to align with word onsets? Do they have such knowledge before knowing any words, or do they learn such knowledge through the experience of knowing words?

There is an observation [172] (pp.108) that English-learning infants may learn the rhythmic cues to word onsets and the MSS strategy from their experience of listening to isolated words, in particular, English first names (e.g., Peter, Tommy, David, etc.) and diminutive forms of words (e.g., doggie, cookie, kitty, daddy, mommy, etc.) that adults repeatedly use in isolation around the infants, mostly for the purpose of catching the infants’ attention. It is reported that the strong-weak stress pattern is common in English first names [86] and that infants start to recognise their names when they are around 4.5 months old [230]. From 4.5 to 7.5 months, the infants have quite a lot of time to experience the isolated trochaic words such as the diminutives and their names and those of their close relatives and caregiver(s), and consequently, develop their sensitivity to the rhythmic patterns of English words. This observation lends support to our argument that cues for words should be learned from known words.

Allophonic Cues

Allophonic cues for speech segmentation are the phonetic variants of some phonemes in a language that correlate with word boundaries. Each acoustic realisation of a phonetic variant of a phoneme is known as an allophone. For example, as noted in [68], the /t/ at the onset of a word in English, e.g., as in “tap” ([tʰ]), has a different pronunciation from the /t/ in other places within a word, e.g., as in “stop” and “hat”. The possibility that allophonic variants of phonemes can signify word boundaries was noted in a number of early studies in 1960-70’s, such as [217, 333]. A frequently quoted example in previous discussions is the pair “nitrate” versus “night rate”: in the former the first /t/ is aspirated, released and retroflexed and /r/ is devoiced, suggesting that /tr/ forms a cluster that only appears in a within-word context; whereas in the latter the first /t/ is
CHAPTER 2. LEXICAL ACQUISITION – A COGNITIVE PERSPECTIVE

un-aspirated and unreleased, and the /r/ is voiced – this /t/ and /r/ together indicate a word boundary in between. One more example to illustrate that allophones really can signify word boundaries is the pair “nice top” versus “nice stop”, as given in [172].

We know that infants perceive speech sounds categorically from a very young age – in general, they hear sounds in terms of their phonological functionality: speech signals with acoustic distinction but no distinctive phonological significance are heard as the same sound, namely, a phoneme. However, this categorical perception does not by any means deprive the infants of the ability to detect the acoustic difference between individual instantiations of the same phoneme in conversational speech. It is demonstrated in [155] that 2-month-old infants could detect the difference between the allophonic variants of /t/ and /r/ in the distinctive pairs “nitrate” and “night rate”.

However, this does not necessarily mean that 2-month-old infants can have the ability to make use of allophonic cues to segment speech into words. Whether the allophonic cues can be used by language-learning infants to facilitate speech segmentation depends on the following conditions, discussed repeatedly in Jusczyk and his colleagues’ recent study [180]:

1. Allophones are an orderly manifestation, instead of random acoustic variants, of phonemic contrasts;

2. Allophones have a distributional correlation with word boundaries;

3. Infants are able to discriminate the allophones from each other;

4. Infants are sensitive to the distribution of allophones within words;

The first three of these conditions are known to have had support from previous relevant phonetic and phonological studies, as discussed above. The psycholinguists’ task is to acquire empirical support for the infants’ sensitivity to allophonic cues and their actual use of the cues in speech segmentation.

There are four experiments reported in [180] that are aimed at testing whether and by which age English-learning infants can have sensitivity to the distribution of allophonic cues within words. In each experiment, 24 infants of 9 months old were tested with 4 words, two target words and two control words. Target words carried allophonic cues of interest, such as “nitrates” and “night rates” – the allophonic cue was the only difference between them; whereas control words carry no cues, but differ from each other significantly in other ways, such as “hamlet” and “doctor”.

In the first experiment, the infant subjects were each familiarised with two words: a target word and a control word. Each of these words was presented in isolation repeatedly to each subject until some familiarisation criterion (e.g., listening time accumulated up to 30s) was met. Then, each of the subjects was tested for each word with a passage of six sentences, each of which contained the particular word once – i.e., the word was repeated six times in the passage. In the test for each subject, all four passages were heard: two for familiar words – one carried an allophonic cue and the other did not – and the other two for unfamiliar words – also one carried an allophonic cue and the other did not – as below, for example:

<table>
<thead>
<tr>
<th>With allophonic cue</th>
<th>Without allophonic cue</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Familiar</strong></td>
<td><strong>Unfamiliar</strong></td>
</tr>
<tr>
<td><em>night rates</em></td>
<td><em>doctor</em></td>
</tr>
<tr>
<td><em>nitrates</em></td>
<td><em>hamlet</em></td>
</tr>
</tbody>
</table>

An infant in the experiment was either familiarised with “night rates” and “doctor” or with “nitrates” and “hamlet”, and then listened to all four passages. The experimental results based on the analysis of the mean listening time to the four passages by each infant indicate that the difference of listening time for the familiar and unfamiliar items is significant for “doctor” and “hamlet” – the pair without allophonic cue, but is not significant for “night rates” and “nitrates” – the pair with allophonic cue. This result suggests that the 9-month-olds did not use the allophonic cues to match the target words they heard during the familiarisation to the test passages containing the correspondent target words.

The second experiment used two monosyllabic words, namely, “night” and “dock”, instead of bisyllabic words, for the familiarisation, and then the subjects were tested with “nitrates”, “night rates”, “dock” and “doctor” passages, to see if the memory demands for the bisyllabic words in the previous experiment was what blocked the 9-month-olds from using allophonic cues. However, the result turned out to show that the allophonic cue in “night” was not used to match the word either to the “night rates” or the “nitrates” passage.

The next experiment tested whether the infants could recognise the word “night” in the “night rates” passage, to clear the doubt caused by the previous experiment. The experiment repeated most of the previous one, except that each occurrence of “night rates” in the test passage was changed to either “night time”, “night games” or some other “night X” item, for the purpose of introducing distributional regularities. The
CHAPTER 2. LEXICAL ACQUISITION – A COGNITIVE PERSPECTIVE

Experimental outcomes showed that the infants listened longer to the “night X” passage than the “nitrates” one, suggesting that the infants did recognise “night” in the passage. Based on this result, we can infer that the infants took “night rates” as an individual lexical item different from, and thus independent of, “night”.

These three experiments so far gave a clear indication that the 9-month-old English-learning infants were not sensitive to the allophonic cues and therefore could not make use of them to extract familiar words from fluent speech. However, would older infants possibly build up their sensitivity to allophonic cues later, say, at the age of 10.5 months old? It is known that infants of this age have developed many abilities to detect words in fluent speech, for example, they can detect words in the iambic stress pattern (beyond their detection of trochaic words since 7.5 months old with the aid of MSS) [154], and can also detect the interruption of words, in either the trochaic or iambic stress pattern, by a pause [259], suggesting that they have had a very good awareness about the well-formedness of individual words by this age.

When the fourth experiment was conducted by repeating the first experiment with 10.5-month-old infants, the result was that the difference of listening time for the familiar and unfamiliar items is significant not only for the pair without an allophonic cue, namely, “doctor” and “hamlet”, but also, more interestingly, for the pair with an allophonic cue, namely, “night rates” and “nitrates”. Comparing this result with that of the first experiment with 9-month-old infants, we can conclude that the English-learning infants have developed a testable sensitivity to allophonic cues between 9 and 10.5 months of age.

Notice, however, there is no direct evidence in [180] showing that the infants actually use the allophonic cues to do speech segmentation, although the experimental outcomes are no doubt consistent with, and even supportive of, this possible use of allophonic cues. More research is needed in this direction to obtain proof as well as other evidence about the effects of using allophonic cues in speech segmentation for lexical discovery.

Phonotactics

Phonotactics is concerned with what sound sequences (in particular, consonant sequences) are permissible and what are disallowed in a word or in some particular positions (e.g., the onset and offset) of a word in a language. For example, sound sequences such [db], [kt] and [zb] are common as the syllable onset in Polish words, but never in English or Chinese words. Also, the [mr] sequence is also disallowable as a syllable onset
in both Chinese and English words; whereas [st] is a frequent sound sequence in English (e.g., “stop”, “stone”, “best”, “least”), but not in Chinese.

Native speakers’ phonotactic judgements appear to be relative to the co-occurring frequency of sounds in their language experience that have shaped their intuition about what sound sequences are allowable and what are unlikely in their language. The extreme end of being unlikely is “disallowable”. Disallowable sound sequences are related to low co-occurring frequency, in particular, the frequency zero that indicates a sound sequence is never observed in the language. Usually we use the term phonotactic constraints (or patterns) to refer to the disallowed sound sequences in a language. Because of the close relation of the legality and illegality of this kind to the probability of co-occurrence, or more precisely, to the distributional regularities, they are also known as probabilistic phonotactics, e.g., as in [340, 237].

It is easy to infer that once a phonotactic pattern appears, it signifies a word boundary. For example, once the phonotactic sequence [⋯mr⋯] is heard in English or [⋯st⋯] in Chinese, it is quite clear to native speakers that a word boundary must appear in between. Fluent speakers, including adults, adolescents and even three- to four-year-old children, are highly sensitive to phonotactic regularities [287, 341, 340], in that they can respond faster to phoneme sequences (either words or non-words) of high-frequency than ones of low-frequency. More interestingly, many high-frequency non-word phonemic sequences are judged by children as more likely to be words than some real words consisting of some rare but legal phonemic sequences, and the children also pronounce such “words” more accurately than the “non-words” [245]. There is evidence that adult speakers exploit their sensitivity to phonotactics to facilitate word segmentation from fluent speech [242].

From the computational perspective, in addition to an early proposal by Church [68] to make use of phonotactic information to facilitate computational lexical processing, it has been demonstrated in Brent and Cartwright’s studies [54, 34] on computer simulation of lexical learning that phonotactic information can be used to significantly enhance the performance of unsupervised lexical learning (see Chapter 4 for a detailed review).

Language-learning infants also appear to have a proper sensitivity to phonotactic well-formedness of words at a very young age. It is reported in [178] that between a list of words with phoneme sequences legal in English but not in Dutch and a list of words with phoneme sequences legal in Dutch but not in English, English learning infants of 9 months old listened longer to the former, whereas Dutch infants of the same age
CHAPTER 2. LEXICAL ACQUISITION – A COGNITIVE PERSPECTIVE

prefered the latter, and both English and Dutch younger infants, of 6 months old, did not demonstrate any significant preference. Also, it is observed in another study [182] that English infants of 9 months old, but not of 6 months old, demonstrated a greater preference for monosyllabic non-words with phonotactic sequences of high-frequency (e.g., chuun) than the ones with phonotactic sequences of low-frequency (e.g., yush).

Furthermore, the young infants also appear to have a certain awareness of the legal position for some phonotactic sequences within words. For example, in Dutch, there are some typical consonant sequences for word onsets (e.g., [br]) and some for word offsets (e.g., [rt]). It is reported in [127] that Dutch infants of 9 months old, but not of 4.5 and 6 months old, can discriminate monosyllables with consonant sequences in permissible positions (e.g., [bref] and [muur]) from those with consonant sequences in impermissible positions (e.g., [febr] and [rtem]), according to their listening preference measured by listening time.

The young infants’ sensitivity to phonotactics, reviewed above, seems to provide a nice basis to support the hypothesis that they can exploit phonotactic information to detect word boundaries [47]. The phonotactic sequences, in particular the cluster of between-word sequences, are known to carry probabilistic information about word boundaries.

Interestingly, a couple of recent studies [238, 237] give further evidence that young language learning infants do make use of phonotactic cues to segment fluent speech into words based on their sensitivity to the alignment of phonotactic sequences to word boundaries. The first experiment in [238] gave conformation that 9-month-old infants prefer, in terms of their listening time, a two-consonant sequence C.C of the within-word cluster over that of the between-word cluster to appear in a bi-syllabic non-word CVC-CVC (e.g., “nongkuth” versus “nongtuth”, where the two clusters are intentionally selected so as to have equally high probability in terms of their frequency in the Bernstein child-directed corpus [17] so that the two clusters only differ in their likelihood of being within a word or at a word boundary). The major findings of the other experiments in [238] are more worth noting: the infants also listened longer to the bi-syllabic non-words with between-word phonotactic sequences showing up in between the two syllables than those with within-word sequences showing up in between, under either of the two following conditions, namely, (1) the second syllable is stressed, or (2) the first syllable is stressed plus a 500-ms pause of silence is inserted in between the two syllables. Recall that language-learning infants of this age tend to assume a stressed syllable is the onset
of a word, and also notice that the 500-ms pause was inserted as a delimiter between words. Thus, we can see that the experimental results gave not only an indication that the infants felt that the between-word phonotactic sequences were more natural aligned with word boundaries, but also gave supporting evidence for the view that the infants use the phonotactic regularities to locate potential word boundaries.

More remarkably, stronger evidence was given in [237] that the infants of 9 months old did exploit probabilistic phonotactics to segment words from fluent speech. Three experiments were conducted as the following in [237]: the subjects were first familiarised with a passage of six sentences with two target words – one of which was a real word (e.g., “gaffe”) and the other a non-word (e.g., “twe”) – either one of the targets was surrounded by a phonotactic sequence of low within-word (i.e., and high between-word) probability\(^2\) at both the onset and offset or at only the onset or offset of the word, respectively, in the three experiments, and then tested, in each experiment, with the two targets and two control stimuli – one of which was a real word (e.g., “pod”) and the other was a non-word (e.g., “fooz”). During the familiarisation phase, the chance for the real word and the non-word to be surrounded by phonotactic cues was half by half in each experiment. Accordingly, there were two types of targets in the familiarisation phrase: “P-cues present” (the one with some phonotactic cues) versus “P-cues absent” (the one with no cues). The experimental results on 24 infants were as follows: they listened significantly longer to the P-cues present target than the others, and their listening time for the P-cues absent target was not significantly different from the two unseen control stimuli. This means that the infants could recognise the target words they heard in the passage that were surrounded by phonotactic cues but could not detect the target word that they heard with no phonotactic cues. The three experiments give a clear indication that the infants did segment words from fluent speech using the probabilistic phonotactics, no matter whether the phonotactic cues appeared at one side or both sides of the target word’s edges.

**Distributional Cues**

Human infants are also found to be sensitive to the statistical regularities embedded in the speech input, in particular the co-occurring patterns. Since such statistical regularities give hints to word boundaries, we also refer to them, alternatively, as distributional cues. It is argued in [268] that the reason why English-learners do not misidentify “can”

\(^{2}\)The probability is also estimated in terms of the frequency in the Bernstein corpus [17].
as a single word in a passage talking about “candle” has to do with their sensitivity to the distributional cues.

How transitional probability may have an effect on infants’ segmentation tendencies is studied in [142]. In the study, 7-month-old infants were first trained with a conditional headturn procedure to discriminate two separate syllables [ti] and [de]. Then they were tested on how well they can identify these two target syllables in various two-syllable contexts. A target syllable may combine with a two-syllable context in one of three ways:

1. The contexts are invariant order strings, e.g., [koga], but the target can appear at either edge, e.g., [tikoga] and kogati;  
2. The contexts are variable order strings, e.g., [kogati] and [gakoti];  
3. The contexts are two identical non-target syllables, referred to as redundant strings, e.g., [kokoti];

Experimental results showed that the infants’ performance in identifying the two target syllables in the invariant order contexts was the best. The researchers concluded that since the two context syllables always co-occurred in the same order with a very high transitional probability, this coherence or distributional property enabled the infants to group the two context syllables into one unit. That is, the two fixed-order context syllables and the target syllable were, respectively, segmented into individual units. Similarly, the redundant context should lead to similar results, but did not. The researchers explained that the two identical syllables lacking coarticulatory cues might be the factor which led the infants not to group them as one unit.

This line of investigation was subsequently extended in [254] to examine how the distributional cues and rhythmic cues would interact to facilitate speech segmentation. In the experiments, the infants were first trained to discriminate the target syllables [ti] and [de], as in [142], and then tested to see how capable they were of identifying the syllable in various two-syllable contexts:

1. The context syllables were always trochaic and fixed-ordered – both rhythmic and distributional cues available;  
2. The context syllables were always trochaic but their order varied – rhythmic cues available but distributional cues not;
3. The context syllables were not ordered in any way – neither rhythmic nor distributional cues available.

The experimental results showed that the infants had a better performance in identifying the target syllables under the first two conditions than the last one, and more interestingly, when the testing lasted for some longer time, the infants’ performance under the first condition was significantly improved but such improvement did not take place under the second condition. These results suggest that the infants were able to integrate the distributional and rhythmic information to do speech segmentation when they were available in the input.

Later, this line of investigation was further extended in [256] with various experimental techniques, yielding the findings that 9-month-old infants’ performance were significantly better when both sequential (i.e., distributional) and rhythmic information were available in the context syllables than when only either sequential or rhythmic information were available. In contrast, 6-moth-old infants performed equally well when both sequential and rhythmic or only rhythmic information was available. These results suggested that infants, at the age of 9 month old, could integrate different sources of information such as the distributional and rhythmic cues, when available, to do segmentation; whereas at 6 month old they could only use the rhythmic information.

More direct proof that young infants use sequential statistics in speech segmentation and word learning is presented in [303]. In the experiments, 24 8-month-old infants were first familiarised with a continuous speech stream of only four tri-syllabic artificial nonsense “words” (e.g., tupiro, golabu, bidaku and podati) in a random order, generated by a speech synthesiser as a consonant-vowel sequence at the rate of 270 syllables (i.e., 90 words) per minute with no pause, no stress differences or any other acoustic or prosodic cues to the word boundaries. In the speech stream, the only cues to word boundaries were the transitional probabilities, which were higher within words (1.0 in all cases) than across words (0.33 in all cases). In the first experiment, after listening for 2 minutes, i.e., 180 words in total, the infants were tested with repetitions of two tri-syllabic words (e.g., tupiro, golabu) and two tri-syllabic non-words (e.g., dapiKu and tihado). Each syllable of the non-words appeared in the stream, but no bisyllabic sequence in the non-words was ever heard by the infants. The test results showed that the infants made a significant discrimination between the words and non-words by listening longer to the non-words – the result of novelty preference, suggesting that the infants could recognise the words they heard during the familiarisation, with the aid of the transitional probability. In the
second experiment, the familiarisation was similar but the test was different: the infants were tested with two tri-syllabic words and two tri-syllabic part-words. The part-words were tri-syllabic sequences each crossing a word boundary in the speech input and all having been heard by the infants during the familiarisation. Again, the infants listened longer to the part-words, suggesting that the infants judged them as novel items, in contrast to the words, which were familiar items – the infants must have learned the words with sufficient specificity and completeness, otherwise they would not have treated the part-words crossing a word boundary as unfamiliar. The infants’ performance in this more difficult discrimination task suggested that they were able to extract sequential statistic information from the input and use it to segment the word-like units out of the speech stream – this can be attributed to nothing else than transitional probability that the infants somehow computed based on their listening experience. This evidence indicates that language-learning infants have a learning mechanism to learn statistical information from speech input to facilitate lexical learning at a very young age.

However, the above experiments do not tell us reliably what particular kind of statistical computation the infants really performed for the purpose of segmenting words from the speech input, since in the experiment both the transitional probability and the co-occurring frequency of words were higher than those of the part-words. To tackle this specific issue, the same group of researchers adjusted the design of the previous experiments in several ways, as reported in [4]. First, the speech stream of four tri-syllabic words in a random order was generated in a way such that two words were twice as frequent as the others, instead of all words being equi-probably generated. More specifically, in the familiarisation phase when the infants heard the speech input for 3 minutes (i.e., 270 words in total), two words appeared 90 times each and the other two 45 times each. Second, the words and part-words used in the test were selected so that they all had the same frequency (i.e., 45 times) in the speech stream. The only difference between the test words and part-words this time was the transitional probability: The transitional probability of any bi-syllabic sequence within a test word remained 1.0 and that within a test part-word was either 0.5 (if the two syllables in question crossed two words) or 1.0 (if the two syllables were within a word). The experimental result was that the infants’ novelty preference for part-words was significant, indicating that it is the transitional probability, not the co-occurring frequency, that the infants computed during the statistical learning of words.

Unfortunately, viewed from a computational perspective, the belief that a learner
CHAPTER 2. LEXICAL ACQUISITION - A COGNITIVE PERSPECTIVE

would follow a piece of distributional information such as transitional probability to
determine the word boundaries in human speech is oversimplified and even naive. The
artificial speech stream used in the experiments reported in [303, 4] is an extreme case:
all words were of the same length, no word embedding, no words shared any common syllable,
all transitional probabilities within words were 1.0 and those between words were
0.33 (or 0.5), etc. What actually is behind the naive idea that the learner computes
transitional probabilities to determine word boundaries is the idea that the learners
would naively infer that the lower points (or local minima) of transitional probability
are word boundaries - computational studies have shown that only about half of all
words in a language like English can be discovered in this way [32]. The main evidence
against this naive idea is that there are so many local minima of transitional probability
appearing within a word that would give false-alarm for word boundaries, and also,
there are many word boundaries at which the transitional probabilities are not local
minima. More interestingly, there are so many kinds of statistical computation that
may give an indication to word boundaries with a certain reliability, how can we be sure
it is exactly the transitional (i.e., conditional) probability that plays the role of guiding
the infants to word boundaries? More specifically, as far as transitional probability is
concerned, how do we know whether the learners use the transitional probability of a
syllable given the preceding one syllable or two or three or more, or use the transitional
probability of two or three syllables given the preceding one syllable or two or three or
more? There are so many possibilities out there, and it appears impossible to follow the
approach of [4] to determine which one is used by lexical-learning infants by eliminating
all other possibilities! There is another possibility that some more comprehensive
statistical measure is used, e.g., our goodness measure - description length gain (DLG),
which is to be defined in Chapter 6.

At this point, we have a number of questions to ask: What is the rationale for the
idea that the learners have to use transitional probability? Is there any theoretical sig-
nificance or preference for using transitional probability over using other statistical mea-
ures? From what theoretical assumption(s) can we arrive at the point that the learners
are likely to, or have to, use transitional probability? So far, we have no satisfactory
answers for any of these questions from the psycholinguistic studies of lexical-learning
infants’ statistical learning ability. What we see is that transitional probability can be
relevant to some extent, just as many other types of statistical measure may be relevant.
In short, what we know for sure is that the infants do compute something statistically
CHAPTER 2. LEXICAL ACQUISITION – A COGNITIVE PERSPECTIVE

Based on their speech experience for the purpose of lexical learning, but we are not sure what they actually compute – it is yet to be explored. More importantly, the cognitive principle(s) underlying this computation is (are) also to be investigated, especially in a more principled way.

To explore the underlying principles beneath a lexical learner’s statistical computation is one of the major motivations for this thesis research. In our approach to computational lexical learning, the theoretical assumption is clear, straightforward and well rooted: a language learner learns linguistic regularities, including words, as speech patterns from naturally-occurring language data that are known to be generated by human speakers following the least-effort principle [361, 362]. It is reasonable, and necessary, to assume that a learner must have a least-effort learning mechanism to accommodate the data, because when the learner has learned a language, he or she also produces new utterances in the same way following the least-effort principle. Computationally, this least-effort learning mechanism is formulated as a strategy to seek for the least-effort (i.e., minimal-cost) representation for the observed language input. Lexical learning can be viewed as an application, or an implementation, of this learning mechanism that is specialized in inferring a lexicon from language data: it searches for a lexicon (i.e., a set of words) consistent with the data that can represent the data with a minimal cost. Computationally, the cost is measured in number of bits that are necessarily used to present the data and the lexicon itself. One goal of this thesis research is to explore how far this learning mechanism can go in the direction of learning word-like lexical items from real language data. More details of theoretical principles and algorithmic implementation of this computational lexical learning approach are presented in later chapters.

2.6 Summary and Discussion

In this chapter we have reviewed the cognitive, in particular psycholinguistic, aspects of recent studies on lexical acquisition by very young human infants. From the cognitive aspect, we can see that preverbal infants are well prepared, perceptually, for lexical learning, in that they can detect individual phonemes (and their acoustic variants), syllables, prosodic marking of various structures (including clauses and phrases), rhythmic patterns, etc., in the speech stream of their native language from an early age. This provides them with a ground to start lexical learning – they can identify the syllables and phonemes, which are the basic building blocks of words in speech stream. So, a
clear scenario for lexical learning is this: a language learning child receives a stream of these building blocks utterance by utterance and induce, with little supervision, the words from this stream with all aids useful and available, including prosodic (rhythmic) characteristics, allophonic cues, phonotactics, distributional regularities, and perhaps some other means we do not know about yet.

A problem with the cue-based approach to lexical acquisition is that the question of how children learn lexical items starting from an empty lexicon is transformed into the question of how they learn (or develop) the cues. How do infants learn words? Answer: They learn words with the aid of some cues. However, how do they learn the cues? We don’t know. More importantly, before knowing any words, how can they know anything that can be cues for words? Are they born with such knowledge? No, impossible. Notice, also, that at the beginning, the infants are not even aware of the existence of words in their language!

Without knowing the existence of words, why do they learn words? Why not some other recurring patterns or some arbitrary strings in the speech input? In other words, what kind of force drives them to learn words, instead of any things else? A possible answer to this question is that, as stated before, the learning infants are not aware of what they are learning, nor the existence of words in their language; what they do in general is use an innate mechanism to accommodate the input data with an economic representation (or storage). Segmenting the input into chunks is a preferable approach for achieving this accommodation at an early stage of language development — we call this stage *lexical acquisition*. Infants also have an innate statistical learning mechanism to detect the distributional characteristics in the input stream and guide the segmentation to arrive at a representation as economic as possible. When infants perceive more input, they have more chunks and more experience in doing the segmentation. Gradually, the chunks converge to a stable set — during the convergence, some other properties (e.g., reference, meaning) may be attached to individual chunks, and the infants also develop some rudimentary ability to string up the chunks to produce their own utterances. Linguists call this stable set the *lexicon* and the individual chunks *words*.

Of course, this sketchy scenario does not mean to deny the existence of cues during the lexical development. But it is reasonable to assume that the cues should be learned somehow from known words or from the learners’ experience of lexical learning. However, the most interesting question we have asked about lexical learning is how the learners start to learn words without any cues? Is it possible for a learner to learn words without
any given cues and constraints but entirely relying on the speech stream as a sequence of sounds with intrinsic distributional regularities? This can be the real situation that a preverbal language learner has to face and overcome, in that they have to derive the useful cues by themselves. Before the derivation of any cue, however, they have nothing more than the distributional information from the input stream. This idea of basing the lexical learning solely on distributional information is at the core of the theory known as the autonomous bootstrapping hypothesis [30], in contrast to other bootstrapping hypotheses such as prosodic bootstrapping and semantic bootstrapping which suppose information about linguistic structures of one type (e.g., prosody, rhythm, etc.) may give some clues to initiate the acquisition of linguistic structures of another type (e.g., syntactic structures like phrases and clauses, and lexical items).

We have argued that all reliable cues and constraints on lexical boundaries must be derived (or acquired) through perceptual experience during the lexical learning process, instead of given a priori from other resources before the learning process. It is logically reasonable that recognising something as a cue or constraint for words must be based on the learner’s knowledge about words. That is, one must know some words first and then can induce reliable cues and constraints on the words. The cues so learned can be applied to identify known words and discover new words in the speech stream. When more words are learned, the learner may induce more cues and other types of knowledge to facilitate word identification and discovery, and also remedy any existing deficiencies in the old cues and strategies of learning.

Our argument is that it is illogical, if not inconceivable, that a learner could acquire any cues for words without any experience of knowing some words. And a certain number of words that a learner learns at the beginning of lexical acquisition are unlikely to be learned solely with the aid of some prosodic and/or allophonic cues. Instead, distributional information (in particular, the repetition and co-occurring patterns) may play a more essential role in helping the infants to start lexical acquisition. This is not to deny the usefulness of various kinds of cues in some later phase of lexical acquisition – their usefulness has been evidenced by so many psycholinguistic experiments – but to argue that distributional information is what the learning can rely on before the availability of other cues. Many cues, e.g., phonotactics, are actually strongly related to and derived from the distributional regularities in the speech stream.

The autonomous bootstrapping hypothesis does not rule out the possibility of seeking help from other resources, in that any useful means available may be used. Basically,
it assumes that the learning should primarily rely on resources within the speech input data for lexical learning, i.e., the co-occurrence and distributional information. The critical point is that, even without external resources, the lexical learning can succeed to a great extent in the discovery of new words if the learners have a statistical learning capacity to detect distributional regularities in the input.

The achievements in computational studies on lexical learning so far have lent significant support to the autonomous bootstrapping hypothesis. Almost all computational models (to be reviewed in Chapter 4) rely solely on distributional information, and many of them have shown an outstanding performance on lexical learning. For example, de Marcken's concatenative model achieved a recall rate higher than 90%, and Brent's MBDP-1 algorithm, based on his probabilistically sound model, achieved an impressive performance with both precision and recall at the level of 70-80%. More importantly, Brent's recent progress in his own work has demonstrated that a better model to utilise distributional information is more critical, and therefore much closer to the underlying learning mechanisms, than a poor model with the aid of heuristic cues such as phonotactics. All these successes in machine learning of a natural language lexicon indicate that the distributional regularities in the data can play a fundamental role in human lexical acquisition, and it is essential to assume that human infants have an innately endowed mechanism to utilise such statistical information in language learning. Although the operational details in machine learning are undoubtedly different from those in human learning, the fundamental mechanisms involved in the human and machine learning to utilise the same information source to detect the same linguistic patterns or regularities should be highly similar in principle, and the studies on both sides should shed light on each other.

In Chapter 4, after we have introduced necessary fundamental conceptions and principles for machine learning in Chapter 3, we will review, at length, a number of representative computational models in the field of machine learning of natural language lexicons.
Chapter 3

Learning as Inductive Inference

3.1 Overview

This chapter introduces the key terms and theoretical basis for our approach to unsupervised lexical learning as inductive inference. It outlines a framework of machine learning from natural language data and gives the theoretical scope of the research covered in this thesis.

Notice, however, that although our study is conceptually rooted in the theories and principles introduced in this chapter and technically related to the technology of modelling, searching and compression, our learning approach does not depend on any profound details of any of these theories and techniques. Our learning approach can be developed straightforwardly from a few basic concepts of information theory following our assumptions (as given in Chapter 1). Therefore, we will not go into any details of formal definition for Kolmogorov complexity nor give the mathematical proof of any principle. Rather, we will present the basic concepts and principles as simply as possible – simplicity is not only what our lexical learner needs to follow to discover the lexicon but also the beauty we intend to pursue in the presentation.

The discussions in this chapter start in Section 3.2 with a number of essential questions about learning. The first question is the basic one – what is learning? Instead of attempting to develop a perfect answer for this question, our discussion aims to achieve a better understanding of the concept of learning. We will first take a brief look at the traditional AI wisdom on the concept of learning, and then move on to define what learning is all about in the scope of our research on unsupervised language learning. The goal of learning in our study is, in general, to trace, or more precisely, to approach as close as possible to, the underlying machinery (or mechanism) that has generated the
CHAPTER 3. LEARNING AS INDUCTIVE INFERENCE

observed data. In our case, the input data is a language corpus in some text format (e.g., orthographic written text or phonetic transcription). In order to represent the underlying machinery, we need to have a model. Straightforwardly, the best model we can arrive at is the one that captures as many regularities as possible in the data – the more regularities found, the closer the model is to the true machinery. From this perspective, the essence of unsupervised learning is to infer the model containing the best set of regularities from the data. In this sense, the term “learning” means, specifically, inductive inference of regularities from observed data.

However, a number of other questions concerning such a concept of learning arise accordingly. What are regularities? How can regularities be represented? How do we know which set of regularities is the best set in the observed data? To answer these questions, we will follow Solomonoff’s insight about the duality of regularity and compression: any regularity can be used to compress the data and anything that can compress the data is a piece of regularity. From this perspective, we can develop a learning-via-compression approach to the unsupervised lexical learning: the set of regularities that can compress the data the most is the optimal model that the learner should look for. To achieve this target, the learner needs proper guidance from some theoretically sound goodness criterion, in order to squeeze out the regularities embedded in the data. Finally, we close this section by giving a scenario of machine learning by exploring how many parties are involved in the learning and what particular roles they play. Special attention is given to the initial (or “innate”) abilities that a lexical learner in the learning-via-compression approach needs to have.

In Section 3.3, we introduce the theoretical basis of inductive inference in relation to language learning, including the concept of Kolmogorov complexity, two versions of MDL principle: Vitányi and Li’s ideal MDL and Rissanen’s MDL, both of which can be derived rigorously from the Bayesian framework of learning. The complexity in the application of the MDL principle to practical learning problems lies in the estimation of description length for a given set of data (given a model) and the model. Different approaches to MDL learning have different “arts” and techniques to pursue a better estimation. Since we adopt a learning-via-compression approach, we need to follow information theory to develop, in a later chapter, a goodness measure to guide the learner to search for a model that has a minimal description for the data. We close Section 3.3 with a review of Solomonoff’s idea of inferring a compact representation for a string with “intermediate codes” to utilise the distributional characteristics within the
CHAPTER 3. LEARNING AS INDUCTIVE INFERENCE

string. This idea gives a special inspiration to our research.

In Section 3.4 we discuss some aspects of language modelling, searching and data compression that are closely related to our research. Learning can be formulated as task of modelling – seeking for an optimal model for given data in terms of some goodness criterion. Learning can also be formulated as a search problem – searching for the hypothesis in the hypothesis space (which is determined by the representation schemata in the learner) that is closest to the true hypothesis. Available technology for searching thus plays a critical role in the learning. In Section 3.4.3, we introduce some fundamental concepts and principles for compression, aiming to give a background for the formulation of the learning-via-compression approach to lexical learning in later chapters. Special attention is given to the approaches of dictionary-based text compression, which aim to compress a given text with the aid of some “words” (or chunks of text) derived from the text.

In the last section, Section 3.5, we discuss the evaluation of learning results, highlighting the theoretical and empirical aspects.

3.2 Learning

3.2.1 What is Learning?

The first question one may ask about the studies of language learning problem can be, what is learning? It is also a question that researchers in the field of machine learning keep asking from time to time. Although many scientists have attempted to answer it by giving various formal definitions, it still remains an important question for us to think about from different perspectives. Successful research in machine learning relies on deep thinking about and a sound understanding of this question.

There have been a number of widely known philosophical answers to this question from the field of (traditional) artificial intelligence (AI) in the past half century. A brief summary can be found in [40]. Simon [315] gives the definition, with an emphasis on task performance improvement, that learning is the change in a system that leads it to perform better next time on the same task or another one from the same population. Scott [311] proposes, while arguing against Simon’s definition, that learning is a process in which a learning machine constructs a retrievable representation of its ongoing interaction with its environment. This viewpoint recognizes that the learner may learn something that is unknown, at that time it is learned, whether or not it is
useful to improve the learner’s later performance. Michalski [246] also gives a general definition, that “learning is constructing or modifying representation of what is being experienced”. Carbonell [48] presents the view that learning involves, operationally, the ability to perform new tasks (that could not be performed before) or perform old tasks better. He defines two types of learning, one to gain new information about the world (i.e., knowledge acquisition) and the other to improve existing knowledge, including improvement of knowledge representation that leads to performance improvement with existing knowledge. Knowledge representation determines what can be held by a machine learner and therefore further determines what can and can’t be learned by the machine.

Mitchell [251] gives a more operational definition:

**Definition:** A computer program is said to **learn** from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*. (p. 2)

From this point of view, the expectation-maximisation (EM) algorithm [107] conducts a typical process of learning, because this algorithm guarantees to improve the probability of the evidence given a language model via tuning the parameters in the model from iteration to iteration of training. When no more improvement can be achieved, the learning ends.

While appreciating the wisdom of the traditional AI philosophy in the above answers and definitions, we also notice that not every particular aspect in these answers is directly relevant to our investigation of unsupervised language learning, nor can we rely on these thoughts to resolve any particular issues in our approach to the learning. Thus, instead of attempting any more fancy answer to the question, or arguing for (or against) any point in the existing philosophical answers (or definitions), we will first identify what kind of learning will be explored in our investigation, and then try to arrive at a better understanding of it by giving a scenario of language learning and looking into what parties are involved and what roles they will play.

**What is to be learned?**

There are many schools of thought about (machine) learning, each for different purpose(s) and aiming at different target(s) of learning [40, 251]. No matter following
which particular thought(s) a machine learner is designed, we suppose it must learn from some data. But, what are there in the data for it to learn? This “what to learn” issue appears to provide a more substantial angle to look into the concept of learning than any general philosophical thinking about what learning is.

As far as machine learning of natural language is concerned, it is reasonable to assume that there is a cognitive machinery (or mechanism, or grammar) in humans to have generated the data – a set of naturally-occurring utterances. What is this machinery? Researchers have been attempting to look into it for a long time, and we are still unable to get a clear picture. We know that a human learner can learn a language from received data and then proceed to produce new data – some well-formed utterances seen or novel. Can we have a machine learner to induce some underlying principles from the data, and describe to us in a principled way about how the data is generated? The answer is no doubt “yes”, what is uncertain is the extent to which it can go.

Technically, we call such a description a model, and sometimes a theory, hypothesis or grammar, depending on the nature of the text where it appears. In the case of lexical learning, the model is a lexicon\(^1\) – a set of individual words. A model induced from observed data is not identical to the real machinery that generated the data – in this sense, we call the real machinery the true model or target model.

In [297], Rissanen identifies the goal of modelling as to trace the underlying machinery that has generated the observed data. Basically, this is also what we want to use the term learning to refer to in the scope of our studies on unsupervised language learning. Notice, however, that a subtle difference between modelling and learning is that the former (e.g., language modelling for speech processing) has an emphasis on the estimation (i.e., tuning) of probabilistic parameters in a model to fit some given data, whereas the latter focuses more on the induction of a set of laws (or rules) as a description for the data-generation machinery or how the data would be generated. We will have some more discussion along this line in Section 3.4.

Although the language data available for learning is usually far from complete, we have identified a clear target for the learning – it pursues a model that is as close as possible to the true one based on the limited data. But what should be there in a model to describe (or represent) a cognitive mechanism? Since the cognitive mechanism for

\(^1\)Henceforth, we allow the terms model, hypothesis, grammar and lexicon to be used interchangeably, unless clarity is a problem in the context.
language generation is so complicated that it is far beyond what the current machine learning technology can describe precisely, the best things that can be put in the model to describe the mechanism based on the data it has generated are the regularities embedded in the data – the best choice available for the approximation.

Since the true language generation machinery is not reachable by our technology, another question concerning the learning is, how do we know which model among so many choices is the closest one to it? Equivalently, how do we know how close a model is to the invisible true one? Actually, we don’t know; and there is no good way to know it directly. This is an intrinsic property of unsupervised language learning. However, calculating the divergence between the probability distribution over the observed data and that over the language that a model can generate in terms of information theory provides an approach to approximate the distance between a learned model and the true one. This approximation is useful, in particular, in the situation of there being no other more reliable means available. Much previous research has relied on it. One of the information-theoretical measures for this divergence is the relative entropy. Another goodness measure for a model is its goodness-of-fit to the data, seen or unseen. We will also have some discussion on this topic later in Section 3.4.

3.2.2 Learning and Regularity Discovery

We have identified the goal of unsupervised learning in our studies as to trace the true model that has generated the data and have argued that the best description we can have for it is an approximation – a model consisting of regularities learned from the data.

For example, we have some invisible cognitive mechanism in our mind to generate language data. This true grammar does not generate natural language data randomly; rather, it functions in a patterned way, and so there must be some regular features left behind in the data that we can look for. If we can dig out all these regularities, describing the true grammar in terms of these regularities can be the best way for us to approximate the true grammar. The major task for unsupervised learning is to dig out such regularities. The more regularities it digs out, the closer the approximation to the true grammar can be.

From this point of view, the theme of unsupervised learning becomes the discovery of regularities embedded in the given data. The regularities found in this way can be utilised not only to describe the true grammar but also to predict future data. In the
literature, this kind of learning to derive regularities (or laws) from given data is known as induction or inductive inference.

A typical inductive inference procedure can derive many kinds of regularity from the data. But this does not mean it can derive all regularities embedded in the data. What regularity can be learned by a learner from some given data depends on many factors. The most salient factor is what knowledge representation formalism the learner is equipped with. We ought not to expect a learner to learn what it cannot represent. In some language learning research, researchers intend to make a learner able to detect, and further characterise, the regularities in language data with a phrase structure grammar, in particularly, a context-free grammar (CFG). There is no doubt that in general, many characteristics in a natural language can be well captured by a CFG. However, there are some regularities in natural language that cannot be captured by a CFG, nor by the mild context-sensitive grammars, e.g., Chinese numbers [292] and word scrambling in some languages like German, Japanese and Hindi [12, 13]. The incapacity of context-free grammars to characterise some linguistic phenomena in natural languages has also been studied by many scholars, e.g., in [167, 314]. From this point of view, we can see the importance of the knowledge representation formalism in language learning – it determines what the learner can and cannot learn.

As stated above, the more regularities a learner can dig out from the data, the better is the resultant model – that is, the closer is the resultant model to the true model. Notice, however, there is no logical or mathematical proof for this, although there are some arguments for it, e.g., as in [337, 338]. What we can have are the empirical criteria to judge the validity of the learning results. For example, when we design a learner to learn words from text input as a long sequence of characters or phonemic symbols, we can evaluate the learner’s performance based on the correct rate that it outputs words. More words correctly output indicate that it has acquired more regularities underlying how characters (or sounds, loosely speaking) form words (or word-like lexical items). This viewpoint goes after Solomonoff’s intuitive evaluation that the results of inductive inference should, finally, be judged by human experts’ empirical knowledge [322].

**Regularities and compression**

We know that there are many regular features in natural language data, and we have defined unsupervised learning in the scope of our research as a process of inductive inference to derive regularities from given data. Once the knowledge representation
formalism to be used in the learning is determined, we can further specify that learning is the process of deriving as many regularities as possible from the data that can be held by the chosen representation formalism.

However, what are regularities? A sound answer to this question will lay a cornerstone for our research; otherwise, all the conceptions about learning we have developed above would be in vain. Solomonoff provides an answer in [322], a piece of pioneering work in inductive inference:

“... any regularity in a corpus may be utilized to write a shorter description of that corpus. Remaining regularities in the descriptions can, in turn, be used to write even shorter descriptions, etc.” (Section 3, p. 8)

If we use the term compression to refer to the process of writing a shorter description (or representation) for some given data, we can paraphrase what Solomonoff says above about regularity as this: any regularity can be used to compress the data; and, anything that can be used to compress the data is a piece of regularity. In this sense, repeatedly co-occurring patterns are typical regularities in natural language. Here, let us have a simple illustration with two binary sequences of an extensive length (say, 10^6 bits):

\[100110011001100110011001\cdots100110011001100110011001\]  
\[10001001100101110011001\cdots1001111000010111010100\]  

The first sequence is generated by repeating 1101 250,000 times. We can write a very short program in C++ language to describe it:

```c++
for (int i = 0; i < 250000; i++) cout << "1001";
```

There are in total 44 symbols in this program, including spaces. If these symbols are in ASCII codes and each is represented by 8 bits in a computer, we need only 320 bits to describe the program. That is, with the aid of the known pattern 1001, we can describe the sequence (3.1) of one million bits using only 352 bits – what a great saving!

In contrast, (3.2) is a bit sequence generated by tosses of a fair coin and contains no regularity. Any program to print it must involve repeating the whole sequence, e.g., the following one in C++:

```c++
cout << "10001001100101110011001\cdots100111100010111010100";
```

We can see that there is no compression available for it. Any description for it must contain the sequence itself. A sequence like this one containing no regularity is known
as a random\textsuperscript{2} sequence and is therefore incompressible: any description for it must be at least as long as itself. In this sense, we identify the notion of regularity with the notion of compressibility. Regularity only exists in compressible data. Consequently, unsupervised learning as inductive inference is possible only on non-random data.

There are many kinds of regularity in natural language data. As a particular language learner is concerned, some pieces of regularity in the data are visible to it and others are invisible, depending on what knowledge representation formalism the learner uses. A piece of regularity is said to be invisible to a learner if its knowledge representation formalism is unable to represent it. In this sense, the concept of learning is narrowed down to the process of digging out visible regularities. Along this line, the ultimate goal of learning is to infer the set of visible regularities that can best compress the data. To perform this task, there needs to be an optimisation process in the learning to derive the optimal set of visible regularities with respect to its compression effect on the data. In general, a learner must first detect all regularities it can in the data, and then output the optimal set as the learning result.

\subsection{3.2.3 A Scenario of Language Learning}

After we have clarified that learning is to induce the optimal set of regularities in the data with respect to its compression effect, it is time to sketch a scenario of machine learning of natural language. We want to have a picture about

- Who learns what from what? (1)

This mainly concerns the input for learning, the learning results as output and the internal representation the learner has to manipulate during the learning. Other question we will also discuss include

- What enables learner to learn? (2)

- What can it learn and what can it not (or what is beyond its ability to learn)? (3)

- What is to be learned from natural language data? (4)

As far as question (2) is concerned, we are particularly interested in the initial capabilities that the learner has to start the learning. That is, from where does the learning begin. The question of how the learning is conducted will be postponed to later chapters – all remaining chapters are devoted to exploring a solution for it.

\textsuperscript{2} We are not going into the details about the randomness here. In general, a random sequence is a non-compressible sequence. See [234, 235, 128] and [219] for more properties of random sequences.
CHAPTER 3. LEARNING AS INDUCTIVE INFERENCE

From what to learn?

First, we expect our learner to learn from naturally-occurring language data instead of artificial data that are generated from some artificial grammar, because our ultimate interest is to study human language learning mechanisms.

However, this does not mean that our machine learner has to learn from speech in the form of sound waves as a human learner does. Learning from speech signals may involve many details of signal processing that are not relevant to language learning mechanisms that are specialised at higher linguistic levels. For example, grammatical learning does not relate to any details of speech processing, so it can do with input consisting of utterances as word sequences, with each word as an individual input unit. Similarly, lexical learning can do with input consisting of utterances as phonemic sequences, with each sound unit represented as a phonemic symbol, e.g., as in [32]. What string is used to represent an individual phoneme is simply a representation issue for the transcription, irrelevant to the learning. Another choice is that the input for lexical learning can be a set of utterances as syllabic sequences, where each syllable may be represented as a non-decomposable string of phonemes.

That is, for language learning, the input can be some transcription of speech in text format. When the speech units are represented as individual transcript symbols from an alphabet, the input is a list of sequences of such symbols – this is the common nature of the input data in the field of language learning at present.

In the case of transcription of spontaneous speech into phoneme sequences being unavailable, orthographic text can be used as input for learning, e.g., as in [97]. The orthographic text can be either written text or transcription of spontaneous speech. We know that the orthographic transcription of speech is significantly different from the phonemic transcription, in that the same phoneme may be transcribed into different symbols in different orthographic words, e.g., [a] may appear as “-er”, “-or”, “-ur” or other forms in the orthographic text.

However, this kind of inconsistency (or irregularity) can be regarded as a kind of distortion or noise in the input data that can help, in a sense, test how capable a learner is of learning. For example, a lexical learner based on some statistical learning mechanism would be problematic if it learned words only from input data of one particular language but not that of any other language. A good learner needs to have adequate capacity to deal with distributional characteristics in the data of different languages. Similarly, a learner would be problematic if it worked only with data in one form (e.g., phonemic
transcription) but not with those in another form (e.g., orthographic transcription), both bearing idiosyncratic distributional characteristics to facilitate the learning.

Our arguments here give a justification for using orthographic text to do lexical learning experiments in our research, to be reported in Chapter 7.

Another critical issue concerning input data for language learning is the \textit{incompleteness} of data. The incompleteness of data for syntactic learning may appears as unavailability (or sparseness) of instance of some sentence (or phrase) patterns, resulting in the absence of these patterns in the learning results (although the true grammar has no problem producing them). The incompleteness of data for syntactic learning appears as the low frequency (e.g., frequency 1 or 2) of string patterns for some words, leading to the learner’s incapability of detecting them. This can explain why so many low frequency words could not be correctly recognised in previous unsupervised lexical learning such as in [273, 97, 32]. Although there are cases that some such low frequency words are learned by chance or by some heuristics, this does not reflect, in general, the learner’s capability of learning following the underlying learning mechanism it has.

\textbf{Who learns?}

This is not a trivial question. Interestingly, very few texts in AI literature explicitly specify (or formally define) what a learner is in terms of its initial ability. It is just taken as granted that a learner, which by default is a computer, is there ready to start to learn. But what of its abilities that have to be utilised to perform the learning? These are not discussed.

Here, when we ask “who learns?” or “what is a learner”, we are not interested in a trivial answer like “a computer learns” or “a computer is the learner”. Instead, we want to specify \textit{what defines a learner}, namely, to identify a basic set of operations the learner can perform on the input data to carry out the learning. Beyond this set of operations, the learner does not resort to anything else.

In the scope of language learning, what is this set of operations? First of all, a learner must be able to receive and store the input data, which are sequences of some atomic symbols, as discussed above (for our purposes, we need not care about how the learner, e.g., a computer, handles the bit sequence representing a symbol). This ability of receiving and storing data assumes that the learner must be able to differentiate between the symbols and copy symbols (e.g., copy symbol sequences from the input stream into the learner’s memory). Other abilities that the learning requires are a few
CHAPTER 3. LEARNING AS INDUCTIVE INFERENCE

essential string operations that can be extended from these two rudimentary operations on individual symbols.

A string operation is carried out by operations on individual symbols within the string(s) involved. The basic string operation that is most needed in our lexical learning is string counting – it provides frequency information for the learner to calculate the goodness of determining a (sub)string in the input to be a word. Other operations such as string splitting and copying are also involved during the learning process. Their role is to smooth the progress of the learning, different from the one that the string counting plays – it enables the learning. All the rest of the string operations, such as concatenation (the reverse of splitting), substitution, extraction, insertion, etc., can be helpful, but the learning can do without them.

String counting is based on the ability of string comparison – which is based on symbol differentiation – and the arithmetic operation addition. It outputs the frequency of a string in the input for calculation of how good this string is to be selected as a word during the lexical learning – the details of the calculation will be given in Chapter 6. Other arithmetic operations involved in the learning, such as multiplication and logarithm are extensions from addition. (It is straightforward that multiplication is an extension of addition. Logarithm is the reverse of power – an extension of multiplication.)

Would such minimally necessary “inmate” abilities (namely, string counting and a few arithmetic operations) enable a machine to learn words from language data without any supervision or guidance from a teacher but merely relying on the distributional regularities in the data? Seeking a positive answer to this is the major theme of this thesis research. The framework of our lexical learning can be sketched as below.

1. Count the frequencies of sub-strings in the input (Chapter 5);

2. Calculate the description length gains (DLGs) for sub-strings based on their frequency (and the atomic symbols’ frequencies) in the data (Chapter 6);

3. Conduct lexical learning based on the DLGs in the following steps (Chapter 7):

   (a) Optimal segmentation: Segment each utterance in the input into an optimal sequence of chunks (i.e., lexical candidates), in that the sum of DLGs over this sequence is greater than any other possible sequence the utterance;

   (b) Lexical Refinement: Repeat the optimal segmentation on each lexical candidate to derive the finer-grained lexical items;
(c) Utterance segmentation: Apply the learned lexicon to segment input utterances into words.

The evaluation of the learning performance will be based on the outputs from the last two steps, to see what percentage of words it can learn and segment correctly from the input.

**What to learn from natural language data?**

As discussed before, the unsupervised learning experiments are to derive an optimal set of regularities from data that can compress the data the most. However, not every piece of regularity in the data can be learned by a learner. What can be learned is determined by the learner's knowledge representation capacity – it cannot learn anything it is unable to represent.

Also, different learners are designed to learn different types of regularity. Lexical learning is to learn words, each being a sub-string in the input corpus. Syntactic learning is aimed at learning a grammar, e.g., in CFG format, from a corpus of utterances, each being a sequence of words. In order to do this, the learner has to be able to first categorise words into syntactic categories (POS) and then detect how they combine with each other into phrases, and how phrases are categorised into phrasal categories and then how the phrases combine with each other to form larger phrases, and so forth. All words as string patterns, phrases as patterns of word sequences and syntactic categories as word classes are typical regularities in natural languages, capturing the relationships of various natures among data items in the input.

There are many types of regularity in natural language data, e.g., coherent repeated patterns, discontinuous patterns. In the structure aspect, the two most salient kinds of regularities in natural language are the syntagmatic and paradigmatic relationships among speech units and sub-structures at various linguistic levels, following the wisdom of traditional linguistics since de Saussure [98]. The former is about how small units (or constituents) combine into larger constituents and the latter about how elements or constituents form syntactic categories.

When semantics is taken into consideration, the situation of language learning becomes more complicated. The major task is to learn the association of meanings with linguistic forms at various levels. In order to enable unsupervised learning for this target, the meanings must be encoded in a suitable form with some elemental meaning symbols and input to the learner – thus the learner has two input streams, one for linguistic form
CHAPTER 3. LEARNING AS INDUCTIVE INFERENCE

and the other for meaning. When meanings are represented as sequences (or structures) of meaning symbols, semantic learning becomes derivation of the association between the form structures and meaning representations. Since the data for the meaning stream is so inadequately available, computational meaning learning is limited to a small scale at preliminary levels, e.g., as in [316, 97].

In contrast, our research focuses on a specific task of learning linguistic structures – lexical items – from the input stream as a sequence of form symbols. Only after the linguistic forms are properly learned can a learner, in particular a human learner, move on to learning the associations of form structures with meanings, no matter how the meanings are represented or whether they also have to be learned by another learning module.

Where are what are learned?

This question seems to ask, trivially, where in a computer will learning results be held. Actually, what underlies this question is that there is some form of representation to store what are learned by the learner in its memory, but how are the learning results really stored?

In our case, the learning result is a lexicon as a list of words, and each word is a string of the atomic symbols in the input stream – the simplest representation. We do not need any more sophisticated formalism for the representation. The adequacy of this representation is clear from the point of view of representation deciding what a learner can learn, in that this simple representation allows the learning by making all possible targets of learning (namely, all possible words in the input) visible to the learner. Also, this representation is neutral, in that it does not make the learning any easier. Both the adequacy and neutrality are what an ideal representation formalism needs to have in unsupervised learning.

A lexicon represented in this formalism is actually a regular grammar (RG), the simplest type in Chomsky’s hierarchy of grammars. For example, a lexicon learned from the tiny corpus (without spaces)

\[
\text{acatsavamousbuthemousedidnotseethecatsothedatcatchthemouse}
\]

can be the following with indices in the left-column below. (Notice that whether we need explicit indices like these in real implementation is another issue, because the order of words in the lexicon already provides information of this kind).
1: a 1 → a
2: the 2 → the
3: cat 3 → cat
4: mouse 4 → mouse

Equivalently, the lexicon in the form of regular grammar, with the indices as the left-hand sides and the words as the right-hand sides, is in the right-hand column.

How good is this lexicon? It depends on how it compresses the data, measured by the difference between the description length of the original corpus and that of the following one, where each lexical item is replaced by its index.

13saw14but24didnotsee23so233ch24

Intuitively, this representation for the corpus is shorter than the original one above, although the word catch is unexpectedly separated into 3 (for cat) and ch – an example of one of the many mistakes that may take place in unsupervised lexical learning.

3.3 Inductive Inference

In general, the term inductive inference, or induction, refers to the process of drawing general laws, principles or propositions from observed evidence. Induction is opposed to deduction, a process of deriving true propositions from given axioms. Generating data following some given rules, such as deriving a sentence from an initial symbol (e.g., $S$) using some (syntactic) rules, can be viewed as an example of deduction. In contrast, inferring rules to cover (or explain) some given language data is an example of induction; our unsupervised lexical learning is another example.

In Solomonoff’s view [322], the goal of inductive inference is, in general, either

- to predict (or extrapolate) the next element (or event) $a_{n+1}$ given a preceding sequence $x = a_1a_2\cdots a_n$ – along this line, finding a prior probability $\mu(\cdot)$ for the estimation of probability

$$p(a_{n+1}|x) = \frac{\mu(xa_{n+1})}{\mu(x)}$$

(becomes the central task of the inference; or

- to infer the underlying process that generated $x$ – another way to predict the next coming event $a_{n+1}$. 

CHAPTER 3. LEARNING AS INDUCTIVE INFERENCE

The second goal is exactly what we want our learning to do, as discussed in the previous section, but our learning is aimed at discovering regularities (or patterns) in the language data that correspond to string patterns for words, rather than at predicting future data.

In this section, we will outline a number of principles for inductive inference, many of which involve probability to various extents. Probability is an abstract notion that can be interpreted, conceptually, either as the ultimate relative frequency of an event or as the degree of our belief in the event. In this sense, the term inductive inference we use here overlaps a bit what is meant by inductive reasoning, which is identified in [219] as a process of deriving or adjusting the probability of a proposition in terms of given data.

3.3.1 Occam’s Razor

A celebrated principle that people have followed for many years while doing inductive inference is the principle of Occam’s razor, which states that

- entities should not be multiplied beyond necessity.

However, more interesting are the interpretations of this principle by various scientists. According to [219] (2nd ed, p. 317), Isaac Newton stated, in reference to the principle, “... Nature does nothing in vain, and more is in vain when less will serve; for Nature is pleased with simplicity, ...”, and Bertrand Russell re-paraphrased the principle as “it is vain to do with more what can be done with fewer” and interpreted it as “among the theories that are consistent with the observed phenomena, one should select the simplest theory”.

Thus, the focus of the principle becomes the notion of simplicity. What is simplicity, or complexity? And how do we measure the simplicity (or complexity), such that we can know some theory is simpler (or more complicated) than another one?

3.3.2 Kolmogorov Complexity

A good measure for simplicity is Kolmogorov complexity, also known as algorithmic complexity [322, 198, 56, 219, 131]. Conceptually, it is the length of the shortest representation (or description) for a theory (or an object, such as a string). Theoretically, it is the size in bits of the shortest effective program to compute (or generate) a description of the theory on a universal computer. Although it is named after Kolmogorov, the one who first identified this notion is Solomonoff [321, 322, 323].
CHAPTER 3. LEARNING AS INDUCTIVE INFERENCE

Here, we will not go into any theoretical details, but highlight two points. The first is that this notion is asymptotically universal and absolute, in the sense that it is independent of any particular machine model up to an additive constant (because universal machines can simulate each other on the execution of any effective process with at most a constant number of extra bits). The second point, which is more important to our research, is that although it is non-computable in general (that is, there is no computationally effective approach to deriving the shortest description for a given theory or object), it provides a basis for a rigorous connection between data compression and learning as inductive inference: among all theories consistent with the data, we can identify, ideally, the one with the least Kolmogorov complexity as the most likely one. It is straightforward in both classic information theory and Kolmogorov complexity theory that a theory (or an object) with a shorter description is more probable (e.g., a shorter string of binary bits is, in general, more probable than a longer one, except that the longer one can be turned into an even shorter description).

A general belief underlying this connection is that the more a theory, being the result of learning, can compress the data, the more it has learned (or generalised) from the data, and thus the better it can predict future data. There is an argument in [336, 338], with quite some mathematical reasoning, that compression is “almost always the best strategy” for learning as inductive inference. Along this line, the ideal learning is the one that can extract all regularities in the data and rewrite the data into the most compact description. In this sense, this shortest description is non-compressible; otherwise, there would still be some regularities in it and thus could be utilised to re-write it into a more compact description.

3.3.3 The Minimum Description Length Principle

The minimum description length (MDL) principle was developed by Rissanen [294, 297], inspired by Solomonoff’s ideas on inductive inference. A similar approach, known as the minimum message length (MML) principle, was proposed earlier by Wallace and Boulton [344], independently of Solomonoff’s work, and was later refined by Wallace and Freeman [345] and related to Kolmogorov complexity. Although the two principles are subtly different (e.g., in their philosophy and derivation), both of them are actually similar.

---

3A comprehensive introduction to Kolmogorov complexity and its applications is in [219], and some recent discussions on Kolmogorov complexity and its approximation by the MDL/MML principle can be found in [131], a recent special issue of The Computer Journal.

4See [331] for discussion at length by the founders and their co-authors on the subtle differences between the MDL and the MML.
CHAPTER 3. LEARNING AS INDUCTIVE INFERENCE

computable approximations for the non-computable approach to inductive inference based on the notion of Kolmogorov complexity – they are similar in essence not only in the theoretical aspect but also in their application to practical problems [219]. From the viewpoint of our research, it is unnecessary to differentiate between them.

The MDL principle states that given some data $D$, the best model (or theory) $M_{MDL}$ in the set $\mathcal{M}$ of all models consistent with the data is the one that minimises the sum of

- the length in bits of the description of the model, and
- the length in bits of the description of the data with the aid of the model.

It can be expressed in a formula as below.

$$M_{MDL} = \arg \min_{M \in \mathcal{M}} l(M) + l(D|M) \quad (3.4)$$

where $l(M)$ and $l(D|M)$ denote, respectively, the description length of a model $M$ and that of the data $D$ under the model $M$.

The MDL principle looks for an optimal balance between the regularities (in the model) and the randomness remaining in the data, that is, a balance between the complexity of the model and the fitness of the model to the data. The extreme case is that the entire data set is trivially taken over as a model; then you need only one bit to describe the data. However, this is not a good model, because it makes no generation from the data.

A Bayesian framework for learning

The MDL principle above can be derived, straightforwardly, from the Bayesian framework of learning (or modelling). A learning problem can be formulated in the Bayesian inference framework as follows: given a set of data $D$ (e.g., a set of natural language utterances) as observed evidence, look for the model $D$ (e.g., a grammar or a lexicon for a language) in a hypothesis space $\mathcal{M}$, of all allowable models consistent with the data, that is most likely to have generated the evidence. The hypothesis space is the set of models that are consistent with observed data and can be represented by the learner’s knowledge representation formalism. This inference framework can be expressed more formally as below:

$$M = \arg \max_{M \in \mathcal{M}} p(M|D) \quad (3.5)$$

$^5$This set is also known as a hypothesis space.
however, it is unknown how this posterior, or inferred, probability of a model \( M \) given the evidence \( D \) can be calculated directly. Fortunately, we can seek help from the Bayes rule, which is derived from the definition of conditional probability.

\[
p(M|D) = \frac{p(D|M) p(M)}{p(D)}
\]  

(3.6)

After applying the Bayes rule to the right-hand side in (3.5), we have another expression for the learning problem:

\[
M = \arg \max_{M \in \mathcal{M}} \frac{p(D|M) p(M)}{p(D)}
= \arg \max_{M \in \mathcal{M}} p(D|M) p(M)
\]  

(3.7)

This derivation uses the fact that \( p(D) \) is a constant, in that it is independent of which \( M \) is chosen.

The above derivation indicates that if the prior, or a priori, probability distribution \( p(M) \) and the conditional probability distribution \( p(D|M) \) of the evidence \( D \) being generated by the model \( M \) are available to the learner, there is a mathematically principled approach to reach to the most likely hypothesis in the given hypothesis space. Accordingly, the learning problem becomes a search for a model that is exactly, or most close to, the target model. If the probability is interpreted as the proper degree of our belief in something, the prior \( p(M) \) can be understood as the degree of the learner’s belief in \( M \) being true before any evidence is seen. In principle, it reflects the learner’s preference for some models over the others in the hypothesis space.

If all models are assumed to have equal probability of being true before any evidence is seen, i.e., the learner has no preference for any hypothesis, then (3.7) becomes

\[
M_{\text{ML}} = \arg \max_{M \in \mathcal{M}} p(D|M)
\]  

(3.8)

This is known as the maximal likelihood (ML) criterion, which has many successful applications in the field of data modelling. It is a special case of the MDL principle - straightforwardly, (3.8) is a special case of (3.7).

However, in order to avoid any unjustified assumptions before the learning takes place, we have to follow (3.7). When taking a negative logarithm of the left-hand side of (3.7), we have (3.9) below.

\[
M = \arg \min_{M \in \mathcal{M}} - \log_2 p(D|M) - \log_2 p(M)
\]  

(3.9)
CHAPTER 3. LEARNING AS INDUCTIVE INFERENCE

These two negative logarithms are the correspondent Shannon-Fano (or Huffman) code-word lengths for the data with the aid of the model and for the model itself, respectively. When we denote them as $l(D|M)$ and $l(M)$ in bits and denote the selected model as $M_{MDL}$, we have the MDL principle expressed in (3.4).

Vitányi and Li’s ideal MDL

However, the Shannon-Fano or Huffman coding uses only the frequency information (as an approximation for the probability) of individual symbols in the data to re-write the data into a compact code with the entropy as the expected code-word length, regardless of all other regularities in the data that can be made use of to reach to a shorter code-word length. If we can have a proper means to utilise other regularities in the data, we can have a shorter representation for the model and for the data under the model, and thus we can arrive at a better form of the MDL principle. Following the Kolmogorov complexity theory, ideally, the most compact code we can think of is the one with Kolmogorov complexity. Along this line of thinking, Vitányi and Li formulate\(^6\) the *ideal* MDL (iMDL henceforth) principle in terms of Kolmogorov complexity in [336, 338] as below in (3.10), and recognise it as “the code-independent, recursively invariant, absolute form of the MDL principle”.

$$M_{MDL} = \arg \min_{M \in \mathcal{M}} K(D|M) + K(M)$$

(3.10)

We know that $K(\cdot)$ is non-computable, so why is iMDL so important to our lexical learning research? First of all, it locates the theoretical ground for the MDL principle. Second, it clearly appeals that the ideal learning bears an attempt to squeeze out all regularities in the data. One thing that is particularly worth pointing out here is that the regularities in the data part has caught people’s attention in the MDL framework of inference, but those embedded in the model part seem to have not. Although the iMDL does not explicitly tell us what should we do if any regularities appear in the model part, an answer that is implicitly meant by $K(M)$ here is that we need to squeeze them out. In order to achieve this, we may need to have another model to describe this model. Similarly, when we have that model, we may need one more model for it – we keep going on this way recursively until no regularities are left in the last model. Then, we can say all regularities have been squeezed out. This is why in our lexical learning we have a

\(^6\)The formulation of iMDL involves replacing the probabilities in (3.7) with the so-called *universal probability* $m(\cdot)$ and the relation between $K(\cdot)$ and $m(\cdot)$ – we skip the details here. See [336, 338] and/or [219] (2nd ed., pp.251-256).
CHAPTER 3. LEARNING AS INDUCTIVE INFERENCE

lexical refinement step following the optimal segmentation procedure to further extract embedded words from the learned lexical candidates.

3.3.4 Solomonoff’s Coding Method for Inductive Inference

In the second part of his famous 1964 paper, Solomonoff illustrates two methods to calculate probabilities for symbol sequences (e.g., strings of linguistic tokens), one calculating the \textit{a priori} probability of a corpus based on some “intermediate code” to transform the original corpus into a more compact representation and the other calculating the probability for a left-most derivation of a string in terms of a given phrase structure grammar - an early (probably the earliest) illustration of probabilistic languages. Both the philosophy and technique in Solomonoff’s coding method of inductive inference with the idea of “intermediate code” are of particular importance to our research on unsupervised lexical learning.

The key point in the “intermediate code” is that given a long sequence of symbols (or tokens), it is possible to introduce new symbols to represent some sub-sequences, in particular the frequent ones, such that a more compact representation of the sequence can be achieved. Introducing new symbols in this way means defining phrase structure rules, e.g., \( \alpha \rightarrow AB \). Solomonoff considers only binary branching rewriting rules: “\ldots we shall consider only definitions that involve the concatenation of two symbols. Since either or both of the symbols may in turn represent a concatenation of two symbols, it is clear that we can in this way define sequences containing any desired number of symbols of the type used in the original uncoded sequence” (p. 233). The possibility of achieving an even more compact representation for the original sequence with multiple branching rules than with the binary branching ones is not considered by Solomonoff. This possibility lies in the idea that a multiple branching rule may need fewer new symbols in the resulting representation.

Solomonoff illustrates the “intermediate code” idea with the example

\[
\text{CABCBABBABABABAAB}
\]

If the rule \( \alpha \rightarrow AB \) is defined, the above sequence can transformed into

\[
\text{AB}_{(\alpha)}, \text{CaCBaB}_{\alpha} \alpha B_{\alpha}
\]

where the comma is a delimiter to separate the rules and the coded sequence. The right-hand side of the rule, i.e., \( \alpha \), which is shown as a subscript for readability, does
not need to appear in the rule part before the comma, because it is simply an “index” for the first rule. It is straightforward to figure out what it stands for: the first pair of symbols before the comma.

Furthermore, when another rule $\beta \rightarrow B\alpha$ is defined, the above intermediate code can be turned into a more compact one:

$$AB_{(a)}\beta_{(\alpha \beta)}C\alpha C\beta \alpha \beta$$

Similarly, it is redundant for $\beta$ to appear before the comma. It is the “index” for the second rule, standing for the second pair of symbols in the rule part.

In the same way, another string

$$CABAABAB$$

can be encoded as below with the aid of the rules $\alpha \rightarrow AB$ and $\beta \rightarrow \alpha A$

$$AB_{(a)}\alpha A_{(\beta)}\beta \alpha \alpha$$

However, whether the $\beta$ rule here can have any effect on reducing the size of the representation is a problem. Thus, we have to consider what rules are worth introducing and what rules are not. That is, we need to have some proper means to differentiate between good rules versus bad rules.

Solomonoff notes that the coding effect of a supposed “regularity” (e.g., $\alpha \rightarrow AB$) can be divided into two parts: one is “the cost of defining the ‘regularity’ ” and the other is “how much we increase the a-priori probability of the code by using this regularity in recoding the data” (p.239). He gives a few complicated formulae to estimate the goodness of introducing a rule (e.g., $\alpha \rightarrow AB$) in terms of “the ratio of increase of a-priori probability of an intermediate code that results from defining $\alpha \rightarrow AB$” (pp.238-9).

We adopt a more straightforward approach to the evaluation of a rule (representing a possible piece of regularity in given data) in terms of the difference of the description length of the intermediate codes before and after the rule is introduced. For example, the gain from defining the rule $\alpha \rightarrow AB$ for the first string above is estimated by

$$|CABABCBBABABAAB| - |AB,C\alpha CB\alpha B\alpha B\alpha|$$

and the gain from the rule $\beta \rightarrow B\alpha$ is

$$|AB,C\alpha CB\alpha B\alpha B\alpha| - |AB\beta,\alpha C\beta \alpha \beta|$$
where the description length $|\cdot|$ is in bits. An option to calculate this length in practice is in terms of Shannon-Fano or Huffman coding. The length difference, as in the above examples, is the gain or goodness of introducing a rule, measured in bits. The formulation of such a goodness measure to guide our lexical learning is postponed to Chapter 6.

Also, it appears unnecessary to limit our approach to using only binary branching rules. It looks more appropriate to let the learning process determine which one among all possible binary or multiple branching rules is more beneficial to learn, in terms of the goodness, i.e., the compression effect, of inducing each possible rule. One of the major tasks in the learning is to determine the best set of rules that leads to the optimal compactness in encoding the given data. The algorithms following this approach for unsupervised lexical learning are presented in Chapter 7.

### 3.4 Modelling, Search and Compression in Relation to Learning

As discussed in the previous sections, learning is to search for the best reachable model for observed data, that is, the one closest to the true model generating the data. There is a close relationship among learning, modelling and searching. The terms “learning” and “modelling” are actually synonyms referring to a similar task with emphasis on different aspects. For example, the general goal identified by Rissanen for data modelling is identified as the goal for our unsupervised learning. Searching is a bridge from the given data to the final model as the result of learning. Compression is closely related to the learning-via-compression approach we are undertaking, although all currently available compression methods, in particular the dictionary-based text compression methods, have not shown any good performance in learning linguistic units from natural texts and it is unrealistic to modify any of them for lexical learning.

In this section, we will briefly lay out some basic concepts, ideas and principles of language modelling, searching and text compression, to provide a wider background for our work on unsupervised lexical learning.

#### 3.4.1 Learning as Modelling

As discussed earlier, learning attempts to achieve better modelling. Language modelling aims at estimating the probability of a given string, particularly a relatively long string,
of some atomic symbols (e.g., characters, words) in terms of a well-established language model based on the probability (or conditional probability) of its sub-strings.

A straightforward estimation of the probability for a string is that we take the relative frequency of the string in an adequately large corpus as the approximation for its probability. This estimation is fairly reliable for short strings, especially, of length one or two. But it becomes less and less reliable when the string in question gets longer. Thus, we have to resort to a reliable language model for the estimation of probability for longer strings, based on the reliable probabilities of the shorter strings involved in the long string. Context models are a popular class of probabilistic models that estimate the probability of a long string based on the conditional probability of each symbol in the string given a preceding sub-string, known as the context of the symbol.

In this section, we will introduce some fundamental concepts about probabilistic languages as the basis of language modelling, the structure and parameters of a model and the computation of probability for a given string in terms of a language model.

**Probabilistic Languages**

A probabilistic language is a language in which each string (e.g., a sentence) is associated with a probability. The early studies on probabilistic languages can be traced back to Solomonoff’s work in early 60’s [322], and even in the late 50’s [319, 321]. The probability of a string can be understood as the ultimate relative frequency of the string or as the degree of our belief in the chance for the string to appear in its language. The former is a frequentist interpretation of probability and the latter a subjective interpretation. The compatibility of the two is shown in a classic paper by Cox [79].

If the frequencies of all strings in a language, say $L$, can be reliably counted in a large enough corpus\(^7\) of the language, the probability of a string $s$, having a count $c(s)$, can be defined as

$$p(s) = \frac{c(s)}{\sum_{s' \in L} c(s')} \quad (3.11)$$

However, not all strings in a language, in particular the rare strings, can be counted because no matter how large a corpus you use for frequency counting, there are many possible strings that are rare enough not to appear in the corpus. In general, a longer string tends to be rarer. Given a finite alphabet $A$, the set of all strings of length $n$ has a size $|A|^n$, which increases exponentially with $n$. This gives a rough idea about how

\(^7\)The ideal corpus would be the one containing all possible strings of the language, which can hardly be available in the real world.
CHAPTER 3. LEARNING AS INDUCTIVE INFERENCE

78

easy it is for a string to fall outside a corpus and how serious the sparse data problem
in language modelling is – keep in mind that the major task of a language model is to
estimate the probability for a given string (we will return to this point later).

The probability of a string \( w_1w_2 \cdots w_n \) (denoted as \( w_{1,n} \) hereafter\(^8\) for conciseness)
can be precisely derived in term of conditional probability as below, where the probability
of \( w_i \) preceded by \( w_{1,i-1} \) is denoted as \( p(w_i|w_{1,i-1}) \).

\[
p(w_{1,n}) = p(w_1)p(w_2|w_1)p(w_3|w_1w_2) \cdots p(w_i|w_{1,i-1}) \cdots p(w_n|w_{1,n-1})
\]

\[= \prod_{i=1}^{n} p(w_i|w_{1,i-1}) \tag{3.12}\]

Notice that \( w_{1,i-1} \) would denote an empty string if \( i \leq 1 \) in the subscript. The proba-
bility \( p(w_i|w_{1,i-1}) \) can be estimated by the relative frequency \( f(w_i|w_{1,i-1}) \) in terms of
the counts \( c(w_{1,i}) \) and \( c(w_{1,i-1}) \) obtained from a training corpus, as below.

\[
p(w_i|w_{1,i-1}) \doteq f(w_i|w_{1,i-1}) = \frac{c(w_{1,i})}{c(w_{1,i-1})} \tag{3.13}\]

The string \( w_{1,i-1} \) preceding \( w_i \) in \( p(w_i|w_{1,i-1}) \) above is known as the context\(^9\) of \( w_i \).
Accordingly, the formula (3.12) represents a context model\(^10\) involving variable-length
contexts.

\(^{8}\)It is also denoted as \( w_{1}^{n} \) in some language modelling literature. We will use the denotation \( w_{1,n} \)
throughout this thesis, in order to avoid any inconsistency or confusion in the usage of subscript and
superscript.

\(^{9}\)The preceding string \( w_{1,i-1} \) is also referred to as the history of \( w_i \), e.g., as in [164].

\(^{10}\)When the contexts are treated as individual states, the context models become Markov models. A
Markov model describes a Markov chain, which has the Markov property that a state in the chain only
depends on its predecessor, that is, \( p(x_i|x_1, x_2, \cdots , x_{i-1}) = p(x_i|x_{i-1}) \), where \( x_i \) (for \( i = 1, 2, \cdots \)) is a
state and \( p(x_i|x_{i-1}) \) is known as transitional probability (or transition function). Accordingly, we have
(3.15) below. With respect to the fact that \( p(w_i|w_{1,i-1}) \equiv p([w_{1,i-1}]w_i|w_{1,i-1}) \), if we define a series of states
\( x_i \equiv w_{1,i} \) (for \( i = 1, 2, \cdots , n \)), (3.12) above becomes (3.14), and finally ends up with (3.15),
which observes the Markov property.

\[
p(w_{1,n}) = p(x_1, x_2, \cdots , x_n) \tag{3.14}\]

\[
p(x_1, x_2, \cdots , x_n) = \prod_{i=1}^{n} p(x_i|x_{i-1}) \tag{3.15}\]

Hidden Markov models (HMMs) [312, 289, 291] are a model class extended from Markov models.
A HMM allows transitions, each from a state to another state yielding an output symbol, to generate
observable data as a sequence of symbols, but the sequence of transitions generating the data is unob-
servable – they are hidden, in the sense that there may be more than one possible sequence of transitions
that can generate an observed sequence of symbols but there is no way to figure out which sequence
actually carries out the generation. There is a large volume of literature on HMMs and their applica-
tions in the fields of speech processing and natural language processing (e.g., part-of-speech tagging).
A comprehensive introduction to HMMs and the search and optimisation algorithms involved can be
found in Jelinek’s book [164]. We do not go into any technical details that are of insignificant relevance
to our work. The Viterbi algorithm [309] with dynamic programming techniques is an exception, which
is adapted for our lexical learning algorithm in Chapter 7.
CHAPTER 3. LEARNING AS INDUCTIVE INFERENCE

However, a serious problem with this context model is that the number of parameters in the model is too large. In particular, this number increases exponentially when the context length gets longer—recall that given an alphabet \( A \), the number of possible contexts of length \( n \) is clearly \( |A|^n \). Just think about the English alphabet of 26 letters; the number of all strings of length 5 is \( 26^5 = 11,881,376 \) – awful! If the letter’s case has also to be taken care of, the problem would be 32 \((= 2^5)\) times worse. In this sense, the formula (3.12) with lengthy contexts is of very little practical use, no matter how precise it is in theory. In addition to the computational difficulty with a large \( n \), an even worse problem with this model is the \textit{sparse data} problem: because there are many unseen strings \( w_{1,i} \) which have zero frequencies in the training data, thus \( p(w_i|w_{1,i-1}) = 0 \), following (3.13). This is undesirable, because every string is possible, whereas a zero probability indicates impossibility. This is the \textit{sparse data} problem in language modelling. The longer the contexts used, the worse the problem.

In order to alleviate this problem and put the context models in practical use, the art of language modelling has to manage to group the contexts \( w_{1,i} \) of variable-length into some \textit{equivalence classes} \( \Phi(w_{1,i}) \) such that a non-zero \( p(w_{i+1}|w_{1,i}) \) can be computed in terms of \( p(w_{i+1}|\Phi(w_{1,i})) \). A choice of such classification is \( \Phi(w_{1,i}) = w_{i-1}w_i \), that is, we use the string of the last two symbols in a long context to represent the entire context. Accordingly, \( p(w_{i+1}|\Phi(w_{1,i})) = p(w_{i+1}|w_{i-1}w_i) \). This is known as a trigram model, which computes the probability of a given string \( w_{1,n} \) using the formula in (3.16). Similarly, we have a bigram model in (3.17) and a unigram model in (3.18). Since a unigram model disregards the interdependency of data items while computing the probability, the probability estimated by a unigram model is known not to be as reliable as that by a bigram or a trigram model for a natural language utterance.

\[
p(w_{1,n}) = \prod_{i=1}^{n} p(w_i|w_{i-2}w_{i-1}) \tag{3.16}
\]

\[
p(w_{1,n}) = \prod_{i=1}^{n} p(w_i|w_{i-1}) \tag{3.17}
\]

\[
p(w_{1,n}) = \prod_{i=1}^{n} p(w_i|\phi) = \prod_{i=1}^{n} p(w_i) \tag{3.18}
\]

Context models of practical use are those with short contexts, particularly, of length 1 or 2, namely, the bigram and trigram models. However, shortening contexts does not mean that we have dispelled the sparse data problem. For example, in a trigram model, there can be many three-symbol strings whose counts \( c(w_{i-2}w_{i-1}w_i) = 0 \) in the training
corpus are zero, resulting in \( p(w_1 | w_{i-2} w_{i-1}) = 0 \), in terms of (3.13). Thus, we need to smooth the probabilities (or relative frequencies) — *smoothing* techniques\(^\text{11}\) have become a crucial part in language modelling technology.

**Model Structure and Parameters**

A language model is mainly determined by three factors, namely, model *class*, model *structure* (or topology) and probabilistic *parameters* assigned to items (mostly, rules in some form) in the structure. Usually, the choice of model class is left for a human designer to make. The most popular model classes in speech processing and statistical NLP are hidden Markov models (HMMs) — a type of probabilistic regular grammar (PRG) — and probabilistic context-free grammar\(^\text{12}\) (PCFGs). The model structure is the set of rules\(^\text{13}\) (or productions), each of which is assigned a quantity, usually, a probability. This set of probabilistic quantities, some of which may be zero (the sparse data problem may thus arise), are known as probability parameters of the model.

Given a language model, the computation of the probability for an input string (e.g., an utterance) may not be as straightforward as (3.13), because there can be many possible ways to generate the string. Each way of generation is known as a *derivation*. The computation needs to take care of each derivation: the probability of a string is the sum of possibilities of all possible derivations that can generate the string. In an HMM, a derivation is a sequence of transitions yielding the string; in a PCFG, a derivation is a parse of the input into a hierarchical tree structure. For example, the probability of a string \( w_1 \ldots w_n \) generated by an HMM \( M \) and by a PCFG \( G \) are, in general, as below in

\[ p(w_1 | w_{i-2} w_{i-1}) = \lambda_3 f(w_i | w_{i-2} w_{i-1}) + \lambda_2 f(w_i | w_{i-1}) + \lambda_1 f(w_i) \]  

(3.19)

where the weights \( \lambda_i > 0 \) (for \( 1 \leq i \leq 3 \)) and \( \lambda_3 + \lambda_2 + \lambda_1 = 1 \). The optimal weights \( \lambda_i \) for different trigram items have to be obtained by training the model on a large-scale corpus in terms of some theoretically-sound criterion, e.g., the maximal likelihood criterion. There are also other smoothing methods, e.g., Good-Turing estimation\(^\text{140, 71}\), held-out estimation\(^\text{165}\) and backing-off method\(^\text{187}\). Nice discussions on them can also be found in\(^\text{164}\).

\(^\text{11}\)There are many ways to do the smoothing. One choice is the *linear smoothing* via interpolating the trigram, bigram and unigram relative frequencies to estimate a probability, as below.

\(^\text{12}\)PCFGs are also known as *stochastic context-free grammars* in some of the literature.

\(^\text{13}\)Markov models are known to be PRGs with rules transformed into a different form of representation, e.g., with states and transitions. The equivalence of Markov models and PRGs is straightforward: the *transitional* probability \( p(x_i | x_{i-1}) \) in a Markov chain can be understood as \( p(x_i \rightarrow x_{i-1}) \) (or \( p(x_i \rightarrow \epsilon x_{i-1}) \)), where \( \epsilon \) is an empty symbol) in its PRG counterpart; similarly, the *output* probability \( p(o | x_i, x_j) \) of generating the symbol \( o \) during the transition from state \( x_i \) to \( x_j \) in a HMM can be transformed trivially to \( p(x_i \rightarrow a x_j) \) in the ordinary representation for PRGs, where the states \( x_i \) and \( x_j \) are read off as non-terminal symbols.
(3.20) and (3.21), respectively.

\[
p(w_{1,n}) = \sum_{D \in \mathcal{D}(w_{1,n})} \prod_{i=1}^{n} p(s_i|s_{i-1}) p(w_i|s_{i-1}, s_i)
\]

(3.20)

\[
p(w_{1,n}) = \sum_{D \in \mathcal{D}(w_{1,n})} \prod_{\alpha \to \beta \in D} p(\alpha \to \beta)
\]

(3.21)

In these formulae, \( \mathcal{D}(w_{1,n}) \) denotes the set of all possible derivations for \( w_{1,n} \) in terms of a model, \(<s_{i-1}, s_i> \in D \) indicates the transition from \( s_{i-1} \) to \( s_i \) within the derivation \( D \) (which is a sequence of transitions in a HMM), and \( \alpha \to \beta \) is a CFG rule. The assumption underlying these formulae is that the transitions or rules involved in a derivation are independent of each other.

Another assumption that has been widely accepted in language modelling is that the utterances in corpus \( C = \{u_1, u_2, \ldots, u_n\} \) are generated independently of one another by a language model (i.e., a grammar). Accordingly, the probability \( p(C|M) \) of the corpus given a model \( M \) can be calculated as (3.22).

\[
p(C|M) = \prod_{u \in C} p(u|M)
\]

(3.22)

A supposition to alleviate the complexity in exhausting all possible derivations of a string for the computation of its probability is that most, almost all, of probability of the derivations is allocated to the most probable derivation. Based on this idea, it is popular to approximate the probability of a string with the probability of the most probable derivation, with the aid of the Viterbi algorithm [339].

In the field of speech processing, where HMMs are popular, the structure of a language model is, usually, pre-defined by some speech engineers. Thus the art of language modelling concentrates on obtaining an optimal set of parameters with respect to some given data, which is usually in a large volume. This process is known as training. The question is, which parameter setting is the best one available? The Bayesian framework we outlined in Section 3.3.3 with the equation in (3.7) applies to this problem – we want the best, that is, the most probable model given the data. Under the circumstances of the model having been fixed, (3.7) becomes (3.8), a ML criterion.

However, the ML criterion is easy to reach through training the model on given data. There are may technical details to be dealt with. The most popular training algorithms for HMMs and PCFGs are, respectively, the forward-backward (or Baum-Welch) algorithm [10, 9] and the inside-outside algorithm [6, 208, 277]. Both can be thought of
CHAPTER 3. LEARNING AS INDUCTIVE INFEERENCE

as incarnations of the expectation-maximisation (EM) algorithm [11, 107] in essence. Both forward-backward and inside-outside algorithms exploit a critical dynamic programming technique of using two sets of variables as probability accumulators, each of which accumulates the probability for all derivations for a prefix or suffix fragment of an input string in terms of a given model. In principle, an EM procedure iterates between an expectation step (E-step) and a maximisation step (M-step) until convergence: the E-step computes the posterior probability of the training corpus under the current settings of the language model, and the M-step attempts to adjust the current setting of the language model to maximise the expected posterior probability of the corpus. The EM algorithm guarantees that each iteration monotonically increases the posterior probability, and thus asymptotically approaches to a local optimum. We are going into the details of model training here. Some more details of the forward-backward algorithm will be discussed in Section 4.5 when de Marcken’s concatenative model is reviewed.

Two factors lead the current language modelling technology for speech processing to deviate from our framework of unsupervised language learning outlined before. One is that the structure of a language model is fixed in advance and only probabilistic parameters are adjusted through training. In language learning, it is more important for the learner to induce the model structure, e.g., the lexicon as a list of words. The training algorithms mentioned above did not bring any significant success in language learning before de Marcken’s work [97], in which the adjustment of the model structure was carefully taken care of. Several empirical studies show that applying the inside-outside algorithm to learning various types of stochastic context-free grammar that maximise the posterior probability of given evidence yield, surprisingly, many linguistically ill-formed structures [51, 52, 277, 94].

Another factor is that the iterations in the EM algorithm are cognitively implausible for human learners. Human learners do not repeat a learning process on the same data again and again tens or hundreds of times. Instead, they learn by passing the input only once, although some optimisation processing may take place in the one-pass process. Brent’s recent probabilistic model [32] for unsupervised lexical learning is a good example to simulate this one-pass on-line incremental learning process.

3.4.2 Learning as Searching

Learning can be formulated as a search problem. For example, unsupervised language learning within the MDL framework can be formulated as a search for the most compact
allowable representation for the input. When the search is to find the best object with
regard to some goodness criterion, instead of looking for a particular object, the search
problem becomes an optimisation problem. In this sense, learning is to carry out an
optimisation process following a pre-defined goodness criterion.

**Hypothesis space and the true hypothesis**

Conceptually, the *hypothesis space* for a learner is the set of all possible hypotheses (or
models) that are consistent with the observed data and also allowable by the repre-
sentation format that the learner uses. When a hypothesis space consists of individual
grammars of some type as candidate models, we also call it a *grammar space*.

A basic assumption in machine learning is that there is a *true or target hypothesis*,
which is the underlying machinery that generated the observed data, from which the
learner will learn. Learning is, in general, to approach to this true hypothesis. But
problems arise: is it in the hypothesis space? How close can we approach to it?

For some learning problems, the true hypothesis is in the hypothesis space and
all possible hypotheses can be effectively enumerated. As far as language learning is
concerned, however, the true hypothesis is not necessarily in the hypothesis space. Why?
Because the grammar formalisms that we can use for computational language learning
today are limited to (stochastic) regular grammars (RGs) (e.g., Markov models) and
context-free grammars (CFGs). It is known for sure that many linguistic phenomena in
a natural language such as English cannot be captured by these grammar formalisms
in a linguistically and cognitively adequate way. Thus, if the grammar in a native
speaker’s mind is supposed to be the target grammar for machine learning of natural
language, there is no doubt that it certainly falls outside a hypothesis space consisting
of (stochastic) RGs or CFGs.

Consequently, language learning can only reach to a hypothesis within the hypothesis
space that is as close as possible to the true hypothesis outside the hypothesis space.
This defines the problem of learning as searching: it searches for the true (or best)
hypothesis in the *searchable* hypothesis space.

**Searching via construction versus enumeration**

A straightforward approach to searching through a finite hypothesis space is by enumer-
ating the individual hypotheses one by one. But for language learning this is seriously
problematic. Although there has not been any formal proof that a grammar space of
CHAPTER 3. LEARNING AS INDUCTIVE INFERENCE

RGs or CFGs consistent with a given corpus of utterances is enumerable, it can be infinitely large and its size may increase exponentially faster than the corpus size. The computational complexity of exhausting a grammar space via enumeration causes a serious problem for the learning. In practice, no one has attempted to do language learning by brute-force searching through the grammar space via enumeration.

This does not mean that language learning is impossible. We can have an alternative way to do the search, namely, search by construction. A very good example of searching by construction is genetic algorithms [156], where the crossover operator is applied to construct a new chromosome – which represents a possible solution for the problem in question, that is, a hypothesis – by integrating parts of some (usually, two) chromosomes of the previous generation into one, such that the new chromosome can have a better chance to reach a higher fitness score in terms of a pre-defined objective function. Keller and Lutz have illustrated the application of a genetic algorithm to induce CFGs for a number of formal languages [190, 189].

The basic idea of searching by construction is that we can reach an optimal hypothesis by putting small good pieces together to build it up. It complies with the divide-and-conquer principle for computing. Although it is known that putting all the best pieces together may not necessarily form the globally optimal outcome, if we know how much good a small piece can do for the possible outcome, this knowledge can certainly facilitate the process of searching via construction. Following this idea, we will develop in a later chapter a goodness measure for each piece of possible lexical item, for the purpose of approaching to a globally optimal lexicon in terms of its compression effect on the input.

What guides the search towards the best hypothesis?

There are many search and optimisation algorithms that can be adopted for language learning, ranging from the very straightforward ones such as best-first (or hill-climbing) search\textsuperscript{14} and beam search, to advanced ones like the EM algorithm [107], genetic algorithms [156], simulated annealing [192], and even ant colony optimisation [108, 110, 109]. All these algorithms need a pre-defined objective function to guide the searching process, by giving feedback to the learner about the goodness (or gain) of each possible move or a sequence of possible moves in the search process, so as to tell the learner where to go next. For language learning, the question is, what guides the search towards the best

\textsuperscript{14}Cook and co-authors’ early work [75] on grammar induction on a number of formal languages is a noticeable application of hill-climbing search in the field of language learning.
CHAPTER 3. LEARNING AS INDUCTIVE INFERENCE

hypothesis within or, more likely, outside the hypothesis space? More interestingly, how do we know if an objective function, which is formulated to denote a goodness criterion, leads the learner to the best hypothesis?

Actually, we don’t know; and the learner never knows either! Whether the learner can reach to the right hypothesis by the guidance of a given criterion depends on how the criterion is formulated. This goes back to our basic assumption(s) of the learning problem and to the theories that are applied to formulate a goodness criterion for the learning. At the point when the formulation is finished, it is hard to foresee if the learner can finally pick the right hypothesis as the learning result with the aid of the criterion. This is the most interesting point in unsupervised learning: the learning result is the only thing that can verify or falsify the rightness of the assumption(s), theories, the criterion and the search strategy formulated for the learning. This also appeals to an empirical evaluation for language learning results in terms of our language intuition.

3.4.3 Compression in Relation to Lexical Learning

It has been argued that compression is the best strategy for learning regularities from data for the purpose of prediction [336, 338]. In the last few sections, we have developed a theory for the learning-via-compression approach within the MDL framework, based on Solomonoff’s idea of inductive inference and Kolmogorov complexity theory. In this section, we take a close look into a number of data compression issues in relation to our lexical learning research.

Coding

Compression is to reduce redundancy in the representation of data. It is possible only if there is redundancy in the data. The simplest idea underlying compression via coding is that we give shorter codes to more frequent symbols in the input, e.g., the Morse codes for English letters. Following this idea, only variable-size codes, in contrast to fixed-size codes (i.e., every code is of the same number of digits, e.g., ASCII codes), are used in data compression.

There is a kind of variable-size code that does not need to have a delimiter in between two codes. It is the prefix codes, which observe the prefix property – no individual code is the prefix of any other code. Shannon-Fano coding [312] is the earliest method for constructing variable-size prefix codes for individual symbols in given data\(^{15}\). Huffman

\(^{15}\)Shannon-Fano coding first sorts the entire list of symbols according to their probabilities (actually,
CHAPTER 3. LEARNING AS INDUCTIVE INFERENCE

coding [158] is a more popular method for constructing prefix codes. It constructs a binary code tree from the bottom up and then traverses the tree to output codes for the symbols\(^\text{16}\), in contrast to the Shannon-Fano coding which works from top down.

However, a problem with this kind of coding for individual symbols is that the number of digits that are actually used for each symbol can be greater than expected. For example, for a particular symbol whose probability is \(p\) (e.g., \(p = 0.1\)), the expected code-word length is \(-\log_2 p\) (e.g., \(-\log_2 0.1 = 3.32\)), but in the coding we can only use \(-\log_2 p\) (e.g., \([3.32] = 4\) bits). That is, if \([\log_2 p]\) \(\neq \log_2 p\), we always waste \(-\log_2 p - (\log_2 p)\) (e.g., \(4 - 3.32 = 0.68\) bits for a symbol of probability \(p\).

This disadvantage in the coding for individual symbols can be remedied by assigning shorter codes to more frequent sequences of symbols, in contrast to assigning a code to a symbol. An example is the arithmetic coding\(^\text{17}\) by Rissanen and other researchers [276, 293, 298, 207], which assigns one code, usually a very long code, to the entire relative frequencies as approximation), and then constructs codes for the symbols in the following steps:

1. Divide the list into two sub-lists that have a sum of probability as even as possible;
2. Assign 1 to symbols in one sub-list and 0 to the rest as the next digit for their codes;
3. Repeat the previous two steps recursively on each sub-list until no division is possible.

\(^{16}\)Huffman coding first takes all symbols as (leaf) nodes, and then keeps merging two available nodes of the least probabilities into a sub-tree — each sub-tree has only its root available for further merging, whose probability is the sum of its daughters'. When the tree is finished, each left and right branch is labelled, respectively, with 0 and 1. The code for a leaf node is the sequence of digits along the path from the root to the leaf.

\(^{17}\)The basic idea of arithmetic coding is to map an input stream to a number in the range \([0, 1.0)\) with respect to the probability distribution over the symbols and their occurrence in the input, in the following steps:

1. Compute symbol probability via counting (and output symbols and their probabilities);
2. Divide the current interval (initially, \([0, 1.0)\)) into proportions by individual symbols' probabilities;
3. Take the next input symbol's proportion in the current interval as the new current interval;
4. Repeat steps (2) and (3) until all symbols in the input are read;
5. Output a number in the current interval — usually, the low bound of the interval is output.

The decoding procedure follows the following steps in principle:

1. Locate the current code \(C \) (initially, it is the result of the coding) in the interval \([0, 1.0)\), output the corresponding symbol and identify the low bound \(L\) and the percentage \(P\) of its proportion in \([0, 1.0)\);
2. Compute the new current code: \(C = (C - L)/P\);
3. Repeat the previous two steps until the current code becomes 0.

In practice, the arithmetic coding needs to take care of a number of technical issues, e.g., how to hold the number which will become smaller and smaller, thus longer and longer, during the coding. A solution is that when the high bound and the low bound of the current interval share a common prefix, this prefix will never change, so we can output this prefix and only need to keep the remaining parts of the numbers.
input. Adaptive arithmetic coding improves the compression efficiency by keeping a
dynamic table for probability (actually, frequency) accumulation in the same way in
both the coding and decoding, instead of outputting a static table of symbols and their
probabilities together with the coding result.

PPM\textsuperscript{18} by Cleary and Witten [73, 252] is a more sophisticated, state-of-the-art com-
pression approach that exploits the adaptive arithmetic coding and bases the probability
estimation on a context-based statistical model, namely, a variable-length n-gram model
with the context\textsuperscript{19} length $n$ relatively short, usually in the range of 2 to 10 in practice
[305]. It is an implicit principle in the statistical modelling for PPM compression that
the longest context available is used for the prediction. The underlying belief is that a
longer context can predict better in general. This is particularly important under the
circumstance that only relative short contexts are used. For example, when the letter $o$
is read following the, if $p(o|\text{the}) = c(\text{theo})/c(\text{the}) \neq 0$, this probability and the letter
will be sent to the arithmetic encoder for encoding, and relevant counts are updated
accordingly (e.g., $c(\text{theo})++$, $c(\text{heo})++$, $c(eo)++$, $c(o)++$); otherwise, look at the
context of next shorter length, that is, he, to see if $p(o|\text{he}) = 0$ or not; if still 0, keep
looking at the next shorter context, until the context of zero length – then, send $o$ and
its probability\textsuperscript{20} $p(o) = c(o)/|C|$ to the encoder for encoding.

**Dictionary-based text compression**

There are four major types of text compression scheme [14]. The three types other
than the dictionary-based methods are based, respectively, on adaptive frequency tech-
niques (e.g., Huffman coding [158]), context models [296] (e.g., the PMM algorithm
[73, 252, 330]) and hidden Markov models [14]. These three types are all statistic-based
approaches, in contrary to the dictionary-based ones.

The dictionary-based text compression uses a variable-length block coding scheme.
It compresses texts by constructing a dictionary of words (i.e., the blocks, also referred
to as phrases), each of which is referenced by a codeword (also known as an index or
pointer). The compression effect is achieved by assigning to a word an index significantly
shorter than the word and then replacing all occurrences of the word by its index. A
dictionary-based text compression algorithm usually employs a deterministic strategy

\textsuperscript{18}The abbreviation PPM stands for “Prediction by Partial Match”, according to [14].
\textsuperscript{19}A string of certain length preceding a symbol in the input that can be used to predict the symbol’s
probability of occurrence is known as the symbol’s context.
\textsuperscript{20}Here, $|C|$ denotes the length of the input corpus that has been read so far.
to parse the input into blocks by an incremental single pass, mainly for the purpose of maintaining a fast speed.

LZ77 [363] and LZ78 [364] are two pioneering methods for dictionary-based text compression by Ziv and Lempel, which have led to a large volume of literature of subsequent research – more than a dozen new methods invented were inspired by them [306]. They are based on a similar idea but with different implementation techniques. The basic idea is to parse the input into a sequence of words such that each word (or phrase) is encoded as the longest existing (i.e., seen) word matching the current string followed by a character. A seen word is represented by the position of its first occurrence in the input. The compression effect is gained from the fact that when more input has been read, a pair of a position number – the word’s index – and a character can represent longer and longer strings. Figure 3.1 gives an example of LZ78 coding, where 0 represents an empty string and the “Index” line is the output of encoding. The words induced in this way from natural language text input are observed to have a certain linguistic relevance: many of them are real words in a linguistic sense.

LZW is an elaborated version of LZ78 that saves the waste of using a following character in each code. It uses the index of a word in a dictionary, instead of the word’s position in the input. This dictionary does not need to be output, because both the encoding and decoding construct them in terms of the input in the same way\(^\text{21}\), with the same word having the same index. A trick in LZW to do without the character following an index to make the last character of a word in the dictionary to be the first character of next word. During the decoding, any time the decoder outputs a word for some index, it also outputs the first character of the next word in the dictionary.

LZMW algorithm [250] is a further elaboration of LZW, which concatenates two existing words, instead of an existing word and a character, into a new word. Thus, LZMW generates a lexical hierarchy similar to the one generated by de Marcken’s concatenative model [97] (see Section 4.5 for details), but with a lower degree of fitness to the data, mainly due to the fact that it uses a greedy on-line parsing strategy, in contrast to de Marcken’s model, which uses an optimisation process.

In general, the encoding in the dictionary-based text compression can be understood as a process of transforming fragments (i.e., the so-called words) of the input into shorter indices. However, all the approaches of coding and dictionary-based compression discussed above are “blind” methods, in that the coding of a symbol (or a block

\(^{21}\) That is, the encoder and the decoder work in lockstep – to use another term, they are synchronised.
of symbols) and the transformation of strings into indices are done before knowing how much compression effect can be gained from such a move being taken. These methods seem to have given up the possibility of optimising the encoding to achieve an optimal compression effect.

In contrast, our unsupervised lexical learning approach, which takes a learning-via-compression approach, inspired in part by Solomonoff’s coding method of inductive inference, computes the gain of compression before determining a word, and the words that are finally identified within an utterance are the chunks which have resulted from an optimisation process with respect to these chunks’ total compression effect (see Chapter 7 for details). This leaves a possibility for our learning approach, which was originally aimed at identifying linguistic words in natural texts, to be adapted as an effective approach to optimal dictionary-based encoding for text compression, because it provides not only an optimisation process but, more importantly, also a mechanism to evaluate how much good a piece of regularity (e.g., a chunk) can do to the compression.

Another point that is worth marking here is that these dictionary-based compression approaches are not aimed at finding the least-cost representation for the data, since the demand for speed allows only a one-pass processing of the input. This demand prevents the compression algorithms from carrying out any optimisation process that needs to go through the input more than one time, contrary to what a language learning algorithm usually does. In our learning-via-compression approach, compression is mainly a way of thinking about the formulation of a learning problem based on our basic assumption that the learners seeks for a minimal representation for the input through learning. It does not indicate that a compression algorithm of the above style has to be incorporated in the learning. This is why a learning program usually does not really carry out any compression, but just computes how much the data can be compressed by the expected learning results. However, when the learning is finished, the compression does take place,

---

22This problem is not as serious as before, due to technological advances. At least, it has been significantly alleviated.
because it results in an optimally shorter representation for the data, under certain
costants (e.g., allowable representation format) though.

3.5 Evaluation – How Well Does It Learn?

Once a language model is output from a language learning process as the learning result,
we need to have some proper means to evaluate it; otherwise we would have no idea
about how well the learner learns. In principle, there are two basic criteria for evaluation:
one is the theoretical criterion and the other is the empirical criterion.

3.5.1 The Theoretical Criterion

As discussed before, learning is aimed to search for a model that is as close as possible
to the true model that generated the data for learning. Thus, the distance of the learned
model from the true one is a proper measure for the evaluation. Unfortunately, we have
no way to calculate this distance directly from the models themselves.

One may think that we can seek help from statistic methods: the learned model,
say $M$, and the true model, say $m$, each determine a probability distribution over the
utterances of the same language, the distance of these two distributions can be used
to measure the distance of the two models, with one of the following formulae, just as
measuring how well a curve fits the given data points – here each utterance is thought
of as a data point.

$$D(M, m) = \sum_{u \in L} |p(u|M) - p(u|m)|$$

$$D(M, m) = \sum_{u \in L} (p(u|M) - p(u|m))^2$$

$$D(M, m) = \sqrt{\sum_{u \in L} (p(u|M) - p(u|m))^2}$$

Similarly, the entropy-based measure Kullback-Leibler distance (or relative entropy) be-
tween two distributions, which is defined as in (3.23), can also be used.

$$D(p(\cdot|M) \parallel p(\cdot|m)) = -\sum_{u \in L} p(u|M) \log \frac{p(u|m)}{p(u|M)}$$ \hspace{1cm} (3.23)

This is not a true distance measure, because it is asymmetric, i.e., $D(p \parallel q) \neq D(q \parallel p)$.
To remedy this “flaw”, one may use the divergence between two distributions, e.g., as
used in [39], which is defined as in (3.24).

$$D(p(\cdot|M) \parallel p(\cdot|m)) = D(p(\cdot|M)\|p(\cdot|m)) + D(p(\cdot|m)\|p(\cdot|M))$$ \hspace{1cm} (3.24)
Unfortunately, this line of thinking does not work in practice, simply because there is no way to know the true distribution \( p(\cdot|m) \); otherwise, we wouldn’t bother with language modelling for the purpose of achieving a better approximation of the true distribution.

Another choice is to resort to the theoretical criterion that is used to guide the learning (or training), to see how well the learner follows this pre-defined criterion. For example, for the learning-via-compression approach to lexical learning that we developed in previous sections in the MDL framework, the minimal description length (or compression effect) that the learner can achieve is no doubt a good criterion. However, a problem with this circular self-evaluation is that when a language learning algorithm is programmed to learn certain linguistic structures (e.g., words) with a pre-defined information-theoretical guiding criterion such as the MDL, the formulation of the learning algorithm comes from, and is also set to test, the basic assumptions. If the assumptions are incorrect, no matter how well the learner follows the criterion, it won’t be able to learn what it was expected to. Therefore, it is necessary to resort to empirical evaluation.

Another choice of theoretical measure for how well a learner learns is the fitness of a model \( M \) for a given set of data \( D \), quantified by the likelihood function \( p(D|M) \), which has been given in (3.22). The maximum likelihood (ML) estimator of probabilistic parameters has been routinely used for speech processing. (3.11) happens to realise the ML estimator, because it allocates no probability to any unseen utterances and maximises the probability of observed ones.

Another measure closely related to this is the perplexity of a model on a given set of data, which has been routinely used to judge the quality of a language model in speech processing community. It is defined as below:

\[
PP(D, M) = P(D|M)^{-\frac{1}{|D|}} \tag{3.25}
\]

where \(|D|\) denotes the number of linguistic elements or symbols in the data set \( D \). Thus, the logarithm of the perplexity

\[
\log_2 PP(D, M) = -\frac{1}{|D|} \log_2 P(D|M) \tag{3.26}
\]

is the average codeword length needed for encoding \( D \) with the aid of the language model \( M \). It is an entropy-based measure. A lower perplexity indicates a better language model for the given data (either training data or testing data).

There is a generalisation problem in language modelling. That is, given an observed data set \( D \) from a language \( L \), we want to find a language model that not only covers
the data set $D$ as well as possible, but also covers as many legal utterances as possible and as few illegal utterances as possible in the language. To test this, the data is divided into a training set and a test set. The test set is used to test how well a language model models unseen utterances. The perplexity of the test set gives an important indicator about how well a model can cover legal utterances outside the training data.

### 3.5.2 The Empirical Criterion

Interestingly, an empirical criterion that is perhaps more important than the perplexity in the evaluation of a language model for speech processing is the error rate in word recognition using the language model, because this measure directly measures how well the language model serves what it is supposed to serve.

Also, linguistically ill-formed structures learned in previous studies that we mentioned before indicates that theoretical criteria are not always reliable. Why? Consider this: you first have a hypothesis and then come up with a statistical, probabilistic or information-theoretical criterion from the hypothesis and use it to guide the learning, and finally you use the criterion itself to evaluate the learning results – such a circular self-evaluation is not trustworthy. It indicates only how close the learner follows the criterion. We are not sure if any of such criteria would bear any linguistic plausibility. There is also no proof that any of them can bridge the learner to what it is supposed to learn. The learning results are the only things that can verify or falsify the learning mechanism based on the hypothesis and the pre-defined criterion.

An evaluation criterion proposed by Solomonoff in [322] is that learning results must be finally judged by human intuition. For unsupervised language learning, this is actually an objective criterion. For example, for the performance of a lexical learner for learning linguistic words from a natural language corpus, what can be better than the percentage of correct words it really learns?
Chapter 4

Computational Models of Lexical Learning

4.1 Overview

This chapter provides a computational background for the lexical learning research of this thesis by reviewing a number of representative previous studies on computational lexical learning. Many computational models have been proposed for lexical learning since the late 60's, including statistical models and neural network models. As highlighted in [33], only in the past five years ten more new models were proposed and further implemented as computer programs by various researchers in the field. In this chapter, we will focus on the statistical models that have a direct relation to and a crucial influence on our work.

In Section 4.2, we will first provide a general discussion of computational models for language learning, and will then move on to reviewing the representative computational models specific for lexical learning: Section 4.3 on an early model by Olivier for lexical learning known as word grammar; Section 4.4 on Brent and Cartwright's distributional regularity (DR) model; Section 4.5 on de Marcken's concatenative model; and Section 4.6 on Brent's recent study, a probabilistically-sound model and its implementation – the MBDP-1 system, whose learning performance demonstrates the state of the art of computational studies specifically devoted to lexical learning. In the last section, Section 4.7, we will have a brief summary to conclude the chapter and summarise the major issues involved in computational lexical learning. We will present our solutions for these issues in latter chapters.
CHAPTER 4. COMPUTATIONAL MODELS OF LEXICAL LEARNING

4.2 Why Computational Models?

The main purpose of the studies of computational models for language learning at various linguistic levels is not to reconstruct in a computer, nor to simulate any exact details of, the actual cognitive process of how a child learns the linguistic units and structures and builds up the mental representation for the linguistic relations among them, although there is no doubt that the success of the computational learning can advance our understanding of the cognitive learning. The input, output and internal representation of linguistic units (e.g., sounds) and structures (e.g., words) in human brains and in computers are so different from each other that the computer simulation of language learning may not really reflect, to any meaningful extent, the details of cognitive processes of language learning that really take place in the human brain. Rather, the computer simulation of language learning is aimed more at testing the effectiveness of the plausible strategies that human learners may exploit to perform language learning tasks [30].

In general, the computational models for lexical learning mainly fall into two categories, namely, connectionist (or neural network) models and probabilistic models. The connectionist approach to language learning has become popular since the mid 80’s. Many researchers have done a lot of significant work on different aspects of language learning, e.g., Rumelhart and McClelland on learning the past tense of English verbs [301], MacWhinney and Leinbach on verb learning [221], Plunkett and Marchman on acquiring verb morphology [288], McClelland and Elman’s TRACE model [239] and Norris and co-workers’ Shortlist model [271, 270] for modelling the competition between activated candidate words in lexical processing, among many others. Some more discussion on neural network approach to language learning can be found in [286, 117]. Since the connectionist approaches are not directly related to the learning approach undertaken in this thesis, we will not discuss them. Instead, we will focus on probabilistic models of lexical learning that have direct influence on our work.

The probabilistic language models serve different research purposes. Many are aimed at enhancing language technology, e.g., language modelling for speech processing [164]. Others are cognitively oriented, aimed at exploring machine learning mechanisms for a better understanding of human language learning. Probabilistic models for lexical learning are focused on computational simulation of the cognitive process of human lexical learning via testing the effectiveness of learning strategies that children may
CHAPTER 4. COMPUTATIONAL MODELS OF LEXICAL LEARNING

exploit to learn a lexicon from a language, with an attempt to gain new insights into human language acquisition in general [273, 354, 356, 34, 32]. In some of the cognitively oriented models, the probabilities may appear in the guise of relative frequencies.

In order to test the effectiveness of plausible strategies that human learners may use in particular learning situations, we must first have a hypothesis about what strategy the learners may use for a particular learning task. A typical hypothesis for language acquisition is in the following form, as given in [30]:

**Hypothesis:** “Children’s ability to perform task \( t \) is partially explained by their use of the strategy \( s \)” (p.3)

From the cognitive perspective, we are interested in knowing how children perform a particular language learning task. Unfortunately, there is no way to directly read a human subject’s mind for the purpose of obtaining a good answer to this question. Fortunately, however, we can examine, through computational modelling at various abstraction levels, what children may compute in their minds in order to perform the learning tasks, and why they do so.

Two abstraction levels are particularly concerned here, according to Marr’s [233] and Brent’s [30] discussions. Originally in Marr’s terms [233], one is *computational*, about what is computed and why, the other is *algorithmic*, about how the input and output are represented and how the computation is carried out on the representation. Marr gives an arithmetic operation, namely, addition, as an example to illustrate these two levels of abstraction: addition theory is at the computational level, in that it provides an abstract and formal account for the addition operation; a mechanical cash register that computes the addition is at the algorithmic level, in that it deals with how numbers are represented and how the addition operation is carried out based on this representation. Another example of machine to compute arithmetical operations like addition is the oriental abacus. It follows the same addition theory as a cash register does but uses a different representation for numbers and, accordingly, its algorithm and operations are different. From this example, we can also see that a computational theory may have more than one algorithmic theory that can implement it. Conversely, an algorithmic theory may correspond to multiple computational theories that describe what it implements.

An important distinction between these two abstraction levels is that, for example, the addition theory is in principle applicable to any numbers with regard to its *competence*, whereas a machine to perform an algorithm for addition, be it a cash register
or an abacus, cannot work on numbers larger than it can represent, that is, its performance is limited by its representation capacity. The distinction between competence and performance is particularly important in linguistic studies, especially in the studies of language acquisition.

Why is testing the effectiveness of a learning strategy so important to the studies of language acquisition? As we mentioned above, a strategy $s$ for a learning task $t$ is mainly about what children compute in their minds in order to perform a learning task. It involves the question why they do so, for which Brent gives two main reasons in [30]: one is necessity, that is, without using strategy $s$ the learners cannot achieve a good performance on learning task $t$; and the other is efficiency (or effectiveness), that is, using $s$ can effectively improve the performance on $t$. The evidence of effectiveness is essential for supporting a hypothesis of the above form.

Computer simulation is a good way to test the effectiveness of a plausible strategy $s$ for language learning. Although we are unable to make a computer to conduct exactly the same learning task as human subjects may do, mostly due to the tremendous difference in the representation and manipulation of linguistic symbols (e.g., sounds) between a computer and a human brain, we can define a machine learning task $t'$ that is similar, in its nature, to a language acquisition task $t$, and then examine the performance of a machine undertaking the task $t'$ by virtue of using the strategy $s$. Certainly, testing effectiveness of a learning strategy by computer simulation in this way requires establishment of a novel algorithm to carry out the formally defined learning task $t'$ by using strategy $s$. It is worth noting that the details of the implementation of the strategy in question may not be psychologically plausible to human subjects. We allow for this for our research purposes, as many researchers in the field do, since what we intend to do by computer simulation is to test the effectiveness of the strategy, not the plausibility of children’s using the strategy exactly in the same way. For example, in a probabilistic language model for lexical learning, all strings involved are associated with a probability parameter, but this probability parameter may not be psychologically real. However, it is reasonable to assume that a parameter of this kind may have a psychological counterpart in the human mind in some other form than a real number, but this counterpart can be represented as a real number for research purposes. Also, the algorithm based on a probabilistic model may be off-line, non-incremental and involve a huge amount of working memory, in contrast to human language learning, which is on-line, incremental and uses a strictly limited working memory. However, the most
important point in computational studies of human cognition is that a computational model with such limitations and psychological implausibility can still bring us a better understanding of a human cognitive process, e.g., lexical learning, if the effectiveness of the strategy embedded in the model can be verified or falsified.

Although bearing a number of limitations, the computer simulation approach to the studies of language learning has several advantages over the empirical method of using behavioural experiments to test a computational theory about learning. Several most prominent advantages are discussed at length in [30]. First, the malleability of the simulation program is so valuable. In the studies of language learning, one of our most important concerns is what factors are likely to have an influence on the learning performance. In computer simulation, it is very simple to have the program re-run many times under different conditions. What we need to do is to set the involved parameters to new values, either automatically or manually, and replicate the experiments. In contrast, psychological experiments are not easy to replicate. Second, the transparency of the simulation program is useful to help us to look into the question of how the learning process is carried out step by step, by which we can gain a better understanding of the nature of the learning process. Although we cannot say that having complete information about a process is a complete understanding (e.g., listing the source codes of a program may not indicate a sound understanding of the program’s mechanism and behaviours), working with a simulation program that we can inspect and manipulate as we want is no doubt a great advance over working with human minds, which are black boxes that we have no way to look into but only can guess what may take place inside based on the experimental outcomes. Other advantages of computer simulation include, in particular, its very fast running speed and its capacity to handle a vast amount of data – this is also unimaginable for behavioural methods in psychological experiments.

In addition to the advantage that simulation experiments can provide a direct approach to testing the effectiveness of a learning strategy at the algorithmic level, the more important is that an algorithm implemented in a simulation program constrains its correspondent computational theories (or models) to those that can have effective implementations. That is, there is always a sound model behind an effective simulation program, or, in other words, a simulation program must simulate a model at the computational level of abstraction. A simulation without a formal model is meaningless, if not unimaginable. What we aim to do with a computer simulation algorithm for a cognitive learning process is to study the computational model behind it and to understand the
fundamental mechanisms underlying the model and, therefore, the cognitive process it models.

In lexical learning, researchers have been exploring the possibility of making use of language models to study how lexical items can be learned from a stream of spoken or written symbols without any a priori knowledge – a typical learning task that a human infant inevitably encounters in the initial stage of language learning. Human infants are assumed to be born with an innate language faculty with a universal grammar that is to be parameterised, through the infants’ language learning, towards the syntax of a particular language. So far, we are not sure if, and how, lexical acquisition is involved in this process of parameterisation. What we know for sure is that lexical items, basically, the words, are the basic building blocks that the target syntax is to operate on, and also that a lexicon for a particular language is not something that an infant can be born with. Thus, we are in a position to posit that lexical learning starts from scratch, initially with an empty lexicon and no prior knowledge about how to extract lexical forms from the up-coming language data, and that the lexical learning process for acquiring word forms is not realised by, nor associated with, the parameterisation of UG, although there may be some universal principles guiding or constraining the learning and there must be some general cognitive mechanisms for information input, output, storage and representation that support the learning. This is exactly the initial situation that a probabilistic lexical learning model starts to work in, with necessary information processing (e.g., string processing) mechanisms that a computer can provide as support.

In the following sections, we will review a number of representative probabilistic approaches to lexical learning that focus on learning word forms.

4.3 Olivier’s Word Grammar Model

Olivier attempted an early computational study on lexical acquisition mechanisms in the late 1960’s with a class of stochastic models known as word grammars [273]. He developed a computer program to derive a word grammar from a given text corpus. Specifically, a word grammar is a finite state grammar consisting of a finite set of words, each of which is associated with a probabilistic parameter. A word in the word grammar is a finite string of symbols, either characters or sounds – more precisely, written symbols representing sounds – from a finite alphabet. A word grammar assumes a language where each utterance is generated by stringing up a finite number of words that are randomly selected from the grammar according to the probability distribution over the
existing words in the grammar. When a word grammar is successfully learned from an appropriate volume of natural language data, its words are expected to be valid word forms in the language and the probabilities will truly characterise the right probability distribution of the words in the language data. No conceptual clarity would be lost if we think of a word grammar as a dictionary with each word associated with a probability.

As with other types of stochastic grammar, the probability of an utterance given a word grammar is the sum of all possible parses of the utterance in terms of the grammar. The probability of an individual parse is the product of the probabilities of all words involved in the parse. A parse of an utterance under a word grammar is actually a flat segmentation of the utterance into existing words in the grammar. This corresponds to how an utterance is generated by a word grammar. Thus, given an utterance $u$, its probability under a word grammar $G$ is

$$p_G(u) = \sum_{s \in \mathcal{P}_G(u)} \prod_{w \in s} p_G(w)$$

(4.1)

where $p_G(w)$ is the probability of an existing word $w$ in $G$ and $\mathcal{P}_G(u)$ denotes the set of all possible parses of $u$ under $G$. The parse with the highest probability within the set $\mathcal{P}_G(u)$ is referred to as the maximum-likelihood (ML) parse for the utterance $u$. Olivier provided an efficient algorithm for finding the ML parse for an input string of $n$ characters long, with a time complexity $O(nL)$, where $L$ is the maximal word length in the dictionary.

The ML segmentation algorithm works on an input utterance $u = c_1c_2 \cdots c_n$ in an iterative fashion as below. It finds the ML segmentation $s_i$ for the prefix $c_1c_2 \cdots c_i$ (for $i = 0, 1, 2, \cdots$) one by one in order. The ML segmentation was called a shortest path in [273] (p. 33). The length of a path, $l(s_i)$, is the negative logarithm of the probability of the correspondent ML segmentation, i.e., $l(s_i) = -\log \prod_{w \in s_i} p_G(w)$. Initially, $l(s_0) = 0$. Once $s_i$ is found, $s_{i+1}$ can be found as below, if it exists:

$$s_{i+1} = \left\lfloor \arg \min_{s_j} l(s_j) + t(c_j \cdots c_{i+1}) \right\rfloor \oplus c_j \cdots c_{i+1}$$

(4.2)

where

$$t(c_j \cdots c_{i+1}) = \begin{cases} -\log p_G(c_j \cdots c_{i+1}), & \text{if } c_j \cdots c_{i+1} \in G \\ \infty, & \text{otherwise} \end{cases}$$

and $\oplus$ indicates a word boundary. Accordingly, the length for the new path $s_{i+1}$ is

$$l(s_{i+1}) = \min_{s_j} l(s_j) + t(c_j \cdots c_{i+1})$$

(4.3)
CHAPTER 4. COMPUTATIONAL MODELS OF LEXICAL LEARNING

When this iteration keeps going on until \( i \) reaches \( n \), the ML segmentation for the given utterance \( u \) is found, if it is a well-formed utterance in terms of the grammar.

In Olivier’s word discovery procedure that employs the ML segmentation algorithm, the initial word grammar consists of the set of alphabetical symbols as starting “words”, each of which is assigned an equal probability. This initial state, of the “words” and the provisional uniform probability distribution over them, indicates that the learner has no \textit{a priori} knowledge about the language, except for the rudimentary ability to identify elemental symbols in the incoming data. This seems to be the real situation that preverbal infants confront before the lexical acquisition process starts: they can perceive speech sounds but have no idea about what sounds in the speech stream would make up a word.

The word discovery procedure works as below when the language data, a corpus of collapsed English written texts with punctuation marks and spaces removed, are fed to it section by section, each of some arbitrary length:

1. Parse each section into the ML segmentation of words.

2. Revise the current dictionary according to the parse of each section:

   (a) Adjust word probabilities according to the new relative frequencies of words;

   (b) Add each pair of consecutive words in the parse into the dictionary as a new word, together with its relative frequency;

   (c) Whenever a certain number of sections has been parsed, discard all words that have appeared only once so far.

Olivier’s experimental results from the corpus of the US presidential nomination acceptance speeches during the period of 1928-1960, of about 60,000 English words, demonstrated that this heuristic lexical inference process could enable the words in the dictionary to converge slowly to orthographic words in the original texts. But the appropriateness of using such data to study infants’ lexical acquisition mechanisms is obviously in doubt. According to the charts shown in [273] (pp.87-8), after parsing 550 sections, about 45% of the input was segmented into correct words and about 75% into words or clumps of words. A re-implementation of Olivier’s algorithm recently by Brent [32] is reported to have reached its peak of performance at 55% precision and 40% recall on a spontaneous child-directed corpus [222], with clumps of words not credited, according the charts provided.
CHAPTER 4. COMPUTATIONAL MODELS OF LEXICAL LEARNING

Olivier’s work seems to have excited many psycholinguists of that time. For example, as noted in a book by Brown [43] on human language acquisition, Olivier’s model was recognised as “a simple and testable prototype of the child’s segmentation problem” (p.396) – as it was originally aimed at being – that “lead to a better understanding of the child’s problem of segmentation and perhaps more generally of grammar discovery ...” (p.397). Even the mistakes it made, e.g., taking clumps of words such as *itis, thesis* and *letme* as single words, were also highly valued, simply because they demonstrated that “[it] also made mistakes which are like the mistakes children make” (p.396). This kind of appreciation seems to stem from the thought that similar mistakes may result from similar learning mechanisms: human infants may also make use of distributional information to facilitate their inference of new words in the very early stage of language learning.

However, as noted in [34], Olivier’s approach does not, in principle, reveal a general principle for lexical learning, because it employs an *ad hoc* heuristic for adding, blindly, in a sense, new words to the dictionary and removing low frequency old words periodically from the dictionary. Although it is psychologically plausible that human learners have a tendency, or a memory mechanism, to forget rarely used words for the sake of relieving the memory burden, it is hardly likely that taking pairs of consecutive words, of frequency one (at the time they are added), as new words would bear any cognitive significance to human’s lexical learning mechanism. A even worse thing in Olivier’s heuristics for new word addition and old word deletion is that these two strategies seem inconsistent with each other: some words of frequency one are added, but others of the same frequency are deleted. It is reasonable to expect that cognitively plausible learning strategies for the same learning task should be consistent with one another.

Nevertheless, Olivier’s work is a nice demonstration that a learning problem can be formulated as an optimisation problem. Most later research in the field was followed a similar framework to work on various language learning issues, although different researchers may employ different evaluation functions (or criteria) to formulate the optimisation problem for particular learning tasks and, accordingly, implement different search algorithms, to approach to a better learning performance.

4.4 Brent and Cartwright’s DR Model

Brent and Cartwright’s *distributional regularity* (DR) model [34] is an early version of the INCDROP (INCremental DR OPtimization) model [31]. Its aim is to study the use-
fulness of distributional regularity, together with phonotactic constraints, to children’s speech segmentation and word discovery inference. According to its mathematical formulation, it is a minimum description length (MDL) model [297, 344].

In the DR model, the segmentation of input utterances into sequences of lexical items (or word-like linguistic units) is also viewed as an optimisation problem. The objective (or evaluation) function that is to be minimised in the optimisation is defined as the minimal number of characters that are necessary to be used to represent the input data with the aid of a lexicon. In other words, the evaluation function is the length, in characters, of the representation of the input data by virtue of a lexicon. There are a great number of possible lexicons given a set of input data. To resolve the optimisation problem for lexical learning is to find the lexicon that can minimised the pre-defined objective function.

The example below in Table 4.1 is given in [34] to illustrate how the number of characters in the representation of an input corpus is calculated. The exemplar tiny corpus is of three utterances:

\begin{itemize}
  \item \texttt{doyouseethekitty}
  \item \texttt{seethekitty}
  \item \texttt{doyoulikethekitty}
\end{itemize}

The representation of the input consists of a lexicon and derivations for the utterances in the input. In the lexicon, all words (more precisely, word types) are indexed with integer indices, e.g., the first word is indexed by 1, the second by 2, etc. Accordingly, the derivation for an utterance is the sequence of the indices whose word forms can string up to form the utterance. The length of a lexicon is the sum all lexical items’ lengths plus the sum of all indices’ length in the lexicon. For example, the first lexicon in Table 4.1 is of 25 characters long, i.e., 5 indices $|\texttt{do}| + |\texttt{kitty}| + |\texttt{you}| + |\texttt{like}| + |\texttt{see}|$. The length of a derivation is the sum of the length, in digits, of all indices involved. For example, the first derivation for the corpus is of 10 characters long, i.e., $|1 3 5 2| + |5 2| + |1 3 4 2|$. In this example, all indices are of only one digit/character long.

In the illustration in Table 4.1, the italic letters and underlined indices are used to highlight the difference from the correct segmentation. One may ask why the segmentation with the shortest representation, that is, the first one, is not the correct one? Actually, this is determined by the nature of the corpus used here, where the words \texttt{the} and \texttt{kitty} always co-occur together, with no exception. This fact leads to treating them together as one single lexical unit, which results in a more economic representa-
CHAPTER 4. COMPUTATIONAL MODELS OF LEXICAL LEARNING

| Segmentation       | Representation | Length $(|\text{Der}| + |\text{Lex}|)$ |
|--------------------|---------------|----------------------------------------|
| do you see the kitty | 1 3 5 2 1 do 4 like | 10 + 25 = 35 |
| see the kitty      | 5 2 2 the kitty 5 see                      |
| do you like the kitty | 1 3 4 2 3 you                         |
| do you see the kitty | 1 3 5 2 6 1 do 4 like | 13 + 26 = 39 |
| see the kitty      | 5 2 6 2 the 5 see                      |
| do you like the kitty | 1 3 4 2 6 3 you 6 kitty                  |
| do you see the kitty | 1 7 2 6 1 do 5 see                      |
| see the kitty      | 5 2 6 2 the 6 kitty                    |
| do you like the kitty | 1 3 4 2 6 3 you 7 yousee 4 like        |

Table 4.1: Brent and Cartwright’s illustration of the calculation of the length, in characters, of representations for a tiny corpus with different lexicons.

tion for the entire corpus, a tiny fragment of a naturally-occurring corpus. If there are other occurrences of the and kitty separate from each other, the segmentation result, in terms of the DR objective function defined previously, will be different. For example, in a real conversation situation, the first occurrence of kitty is usually preceded by the determiner a, instead of the. If we change the corpus accordingly, we can see in Table 4.2 that treating the kitty as a single word is no longer as economic as before and that the correct segmentation leads to the shortest representation of 41 characters.

From these examples, we can also see what Brent and Cartwright call context variability effect and frequency effect. The former refers to the DR objective function’s preference for interpreting sub-strings that occur in a variety of contexts as single words. The latter indicates that the more frequently two sub-strings (e.g., the and kitty) co-occur adjacent to one another, the more likely they will be treated as a single word.

Formally, the objective function for the DR model is formulated as in (4.5).

$$f_{DR}(S) = |\text{TOKENS}(S)| + |\text{TYPES}(S)| + \sum_{w \in \text{TYPES}(S)} |w|$$

(4.4)

where $S$ is a segmentation, $\text{TOKENS}(S)$ is the set of word tokens in $S$, $\text{TYPES}(S)$ is the set of word types involved in $S$, and $|\cdot|$ is the cardinality of a set (i.e., the number of items in a set, e.g., $|\text{TYPES}(S)|$) or the length of a string (e.g., $|w|$). In the right-hand side of the formula, the first item is the number of indices in the derivation of $S$, the second item is the number of indices in the lexicon and the third one is the sum of all words’ lengths
| Segmentation                  | Derivation | Representation | Length ({|Der|} + |Lex|) |
|------------------------------|------------|----------------|-------------------|
| do you see *kitty*           | 1 3 5 6    | 1 do           | 4 like            |
| see *the* *kitty* do you like *the* *kitty* | 5 2 1 3 4 2 | 2 *the* *kitty* | 5 see             |
|                              |            | 3 you          | *6 *kitty*        |
|                              |            |                | 10 + 32 = 42      |
| do you see a *kitty*         | 1 3 5 7 6  | 1 do           | 4 like            |
| see *the* *kitty* do you like *the* *kitty* | 5 2 1 3 4 2 | 2 *the* *kitty* | 5 see             |
|                              |            | 3 you          | *6 *kitty* *7 *kitty* |
|                              |            |                | 11 + 33 = 44      |
| do you see *kitty*           | 1 3 5 6    | 1 do           | 4 like            |
| see *the* *kitty* do you like *the* *kitty* | 5 2 1 3 4 2 | 2 *the* *kitty* | 5 see             |
|                              |            | 3 you          | *6 *kitty*        |
|                              |            |                | 12 + 33 = 45      |
| do you see *kitty*           | 1 3 5 7 6  | 1 do           | 4 like            |
| see *the* *kitty* do you like *the* *kitty* | 5 2 1 3 4 2 | 2 *the* *kitty* | 5 see             |
|                              |            | 3 you          | *6 *kitty* *7 *kitty* |
|                              |            |                | 13 + 28 = 41      |

Table 4.2: Illustration of the calculation of the length, in characters, of representations for a more natural corpus. Italic letters are used to highlight the difference among the representations.

in the lexicon. In a real implementation of the DR model, it is possible to calculate this objective function for any segmentation of an utterance without a real construction of the representation. This property of the model saves a lot of computational effort and enables the simulation program to run much faster, because the construction of a representation is a very expensive operation, in terms of both time and memory space. This is particularly valuable while running a machine learning system on a large-scale natural language corpus.

Given the objective function in (4.5), the optimal segmentation that the DR model selects for an input corpus $C$ can be expressed by the following:

$$S_{op} = \arg\min_{S \in \text{SEGS}(C)} f_{DR}(S)$$  \hspace{1cm} (4.5)$$

where $\text{SEGS}(C)$ refers to the set of all possible segmentations over the input corpus $C$. This expression states that the optimal segmentation is the one, among all possible ones, with the minimal value in terms of the DR model’s objective function.

The representation schema for the DR model, as illustrated in Figure 4.1 and 4.2 above, may have variations. For example, we can use binary digits, instead of decimal
CHAPTER 4. COMPUTATIONAL MODELS OF LEXICAL LEARNING

digits, to represent word indices. Or, we can also use fixed-length numbers, be they binary or decimal, for word indices. For any variation of the representation schema, what the DR model looks for remains the same – it looks for a segmentation over the input corpus that minimises the following:

1. The number of word types in the segmentation
2. The number of word tokens in the segmentation
3. The sum of all word types’ lengths in the lexicon

These three items correspond, respectively, to the middle, the first and the last items in the left-hand side of the equation in (4.5).

The value of the DR model’s objective function is measured by the number of characters (or phonemic symbols) used in a representation for the input corpus. This may not be unbiased, since, following information theory [312], the more frequent items, for example, the indices of the more frequent words, can be represented by fewer binary digits than the rarer ones. The total number of bits needed to describe a representation for a segmentation can be substantially decreased to minimum, on average, if we encode the characters and indices involved in the representation and the lexicon in terms of their frequency distribution in the segmentation. According to Huffman coding [158], for a word $w$ with a relative frequency $\hat{p}(w)$ in the input corpus, we only need to use $\log_2 \hat{p}(w)$ bits to encode its index. Here, we have

$$\hat{p}(w) = \frac{c(w)}{|\text{TOKENS}(S)|} \quad (4.6)$$

If we do so, the minimum average codeword length, in bits, for each word index in the derivation part of the representation for a segmentation $S$ of the input corpus is the empirical entropy of the segmentation, $H(S)$, which is defined in (4.8).

$$H(S) = - \sum_{w \in \text{TYPES}(S)} \hat{p}(w) \log_2 \hat{p}(w) \quad (4.7)$$

In order to minimise the cost for describing the word indices in the lexicon for a segmentation, it is necessary to also minimise the following:

4. The entropy of the relative frequencies of the words in the segmentation.

It is straightforward that a segmentation algorithm that tries to minimise the entropy will tend to choose a segmentation that has as few words as possible to account for most
of the frequency, instead of a segmentation in which the frequency is divided by words more evenly. Brent and Cartwright call any function that has the above four properties a \textit{distributional regularity} (DR) function.

For the implementation of the simulation program for the DR model, the objective function is further elaborated to calculate the number of bits that are necessary to describe the representation for a segmentation of the input corpus. Recall that a representation consists of two parts: a lexicon and a derivation. The total cost for the representation of a segmentation is the sum of the costs of the two parts.

The lexicon is a list of lexical entries, each of which consists of an index and a word form. As discussed above, in order to minimise the cost of describing the derivation, the indices in the lexicon need to be encoded according to their correspondent words’ frequency in the segmentation, that is, using \( \log_2 \hat{p}(w) \) to encode the word \( w \)'s index, following Huffman coding \cite{158}. Thus, the average codeword length for an index is \( H(S) \), and the total number of bits used to describe the derivation is estimated as

\[
|\text{TOKENS}(S)| \times H(S)
\]

While encoding the indices this way, we need to represent the shape of the Huffman tree, which has \(|\text{TYPES}(S)|\) leaves. Brent and Cartwright followed Quinlan and Rivest’s method \cite{290} to encode a binary tree with one bit per node. The total number of nodes in the Huffman tree in this case is \( 2|\text{TYPES}(S)| - 1 \).

Brent and Cartwright take a shortcut to deal with the encoding of the word forms in the lexicon. First, they use a fixed-length code to encode each letter involved in the words, be it a phonemic symbol or a character. Assuming there are \( P \) letters, each code consists of \( \log_2 P \) bits, on average. The total number of bits to represent the words in the lexicon is

\[
(\log_2 P) \sum_{w \in \text{TYPES}(S)} |w|
\]

Second, they concatenate all words together with a delimiter “.” inserted in between any two words, e.g.,

\[
[t] [x] [a] - [x] [i] [t] [x] [y] - [i] [i] [x] [e] - [d] [o] - [y] [a] [a] - [a] [a] [e]
\]

where each pair of square brackets with a letter inside stands for the binary code for the letter. However, it appears that they make a mistake in counting each delimiter letter as one bit\(^1\): “... there is one ‘.’ for each word in the lexicon, so we must \(|\text{TYPES}(S)|\) for

\(^1\) A more appropriate treatment of these delimiter characters is to encode them each as an individual
CHAPTER 4. COMPUTATIONAL MODELS OF LEXICAL LEARNING 107

these characters.”(p. 37) So the total cost for the lexicon is the sum of the costs for the
Huffman tree, for the words and for the delimiter characters, i.e.,

$$3|\text{TYPES}(S)| - 1 + (\log_2 P) \left( \sum_{w \in \text{TYPES}(S)} |w| \right) \quad (4.12)$$

The total length, in bits, of the representation for a segmentation is the sum of the
length of the lexicon, as in (4.13), and the length of the derivation, as in (4.9). Since a
constant has no effect on the solution of the optimisation problem expressed by (4.6),
-1 in (4.13) is omitted. The DR function is finalised as below:

$$f_{LR}(S) = 3|\text{TYPES}(S)| + (\log_2 P) \left( \sum_{w \in \text{TYPES}(S)} |w| \right) \quad (4.13)$$

The main idea of the search algorithm given in [34] for the optimal segmentation for
an input, in regard with the DR function, can be summarised as the following:

1. Apply the DR function to the original input, and select it (as the best segmentation
   at the beginning);

2. Try all ways of inserting one and two word boundaries into the last selected seg-
   mentation, and apply the DR function to all new segmentations so resulting;

3. Select the best one among all new segmentations, and repeat step (2) on it, until
   no more word boundaries can be inserted into the last selected segmentation;

4. Output the best one among all selected segmentations as the final result.

Obviously, this is a brute-force best-first search, which may not return the truly best
one as the final result. As reported in [34], if the above step (2) is restricted to insert
one boundary only, it results in a worse segmentation performance, whereas the same
search algorithm using two other DR functions outputs similar segmentation results as
using the DR function in (4.14).

letter involved in the lexicon. Accordingly, the fixed number of bits to encode each letter becomes
$$\log_2 (P + 1),$$ and the cost for the words in the lexicon is adjusted as

$$\sum_{w \in \text{TYPES}(S)} |w|), \quad (4.10)$$

instead of

$$|\text{TYPES}(S)| + (\log_2 P) \left( \sum_{w \in \text{TYPES}(S)} |w| \right). \quad (4.11)$$
Several experiments were conducted with the DR optimisation algorithm. The baseline for comparison was a random segmentation of the input into the same number of words as the original corpus. As noted in [34], this is not a truly random segmentation, because the number of words is known - so, it is called a word count algorithm. The input was the phonetic transcripts of nine mothers’ spontaneous speech to their children from the CHILDES collection [222]. The phonetic transcripts were transcribed from the orthographic transcripts made by Bernstein-Ratner [18]. Onomatopoeia and interjections were eliminated, and the word boundaries were removed, but the utterance boundaries in the original corpus were kept. The orthographic segmentation in the original corpus is used as the standard to score each experiment result. The segmentation performance is evaluated by two measures\(^2\), namely, precision and recall.

The experiments were conducted on the baseline word count algorithm (WC), the DR optimisation algorithm (DR), and these algorithms integrated with language-specific phonotactic constraints, like the vowel constraint (DR + V) and boundary clusters constraints (WC/DR + V + BC), aiming at testing the usefulness of the DR optimisation and phonotactic constraints in word discovery. The vowel constraint is that every word in English must contain at least one vowel. The phonotactic boundary constraints are comprised of consonant clusters that are not allowed to appear in any word-initial or word-final position in English. These are language-specific constraints. Two types of phonotactic boundary constraints were tested, one was the hand-coded (HAND), another was utterance boundaries (UB) collected from the corpus data. The experimental results, in terms of precision and recall, are summarised in Table (4.3) below. We can see from the last two rows in the table that the two types of phonotactic boundary constraints are equally useful to DR segmentation when used with the vowel constraint, although the UB leads to a lightly better precision and recall than the HAND; but the difference appears to be of very little statistical significance.

Although the unconstrained DR segmentation achieve a performance not too sur-

\(^2\)These two measures are also called accuracy and completeness, respectively, and defined as below in a number of Brent’s papers prior to [32]:

\[
\text{accuracy} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \\
\text{completeness} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}
\]  

(4.14)  

(4.15)

where the true positives are correct words in the segmentation, false positives are erroneous words in the segmentation and false negatives are words in the standard that do not line up with words in the segmentation. Actually, the denominator in (4.15) is the number of words in a segmentation output and that in (4.16) the number of words in the standard segmentation.
Table 4.3: Performance of Brent and Cartwright’s baseline WC segmentation and DR optimisation segmentation integrated with various types of phonotactic constraint, averaged over nine mother-child dyads. Standard deviations are shown in parentheses.

<table>
<thead>
<tr>
<th>Constraint Type</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WC</td>
<td>DR</td>
</tr>
<tr>
<td>Unconstrained</td>
<td>13.4 (1.9)</td>
<td>41.3 (12.7)</td>
</tr>
<tr>
<td>+ V</td>
<td>-</td>
<td>68.2 (5.1)</td>
</tr>
<tr>
<td>+ V + BC (HAND)</td>
<td>39.9 (2.5)</td>
<td>75.7 (4.4)</td>
</tr>
<tr>
<td>+ V + BC (UB)</td>
<td>-</td>
<td>76.3 (5.0)</td>
</tr>
</tbody>
</table>

prising, the effectiveness of DR optimisation is clearly evidenced by the improvements that it makes on the precision and recall over those of the unconstrained and constrained WC segmentations, as highlighted in the upper part of Table (4.4). It is worth noting that the DR segmentation starts from a state with no prior linguistic knowledge, except for the input corpus as a sequence of phonetic symbols transcribed from spoken speech. The effectiveness of phonotactic constraints is shown in the lower part of Table (4.4). We can see that using each type of phonotactic constraint leads to substantial improvement in both precision and recall. From the data, we can also see that the DR optimisation and the phonotactic constraints are largely independent and additive.

Table 4.4: The effectiveness of DR optimisation and phonotactic constraints, shown by how much increment of precision and recall they have caused in Brent and Cartwright’s experiments.

<table>
<thead>
<tr>
<th></th>
<th>WC → DR</th>
<th>Incr.</th>
<th>WC +V +BC → DR +V +BC</th>
<th>Incr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision (%)</td>
<td>13.4</td>
<td>+27.9</td>
<td>208%</td>
<td>90%</td>
</tr>
<tr>
<td>Recall (%)</td>
<td>13.4</td>
<td>+33.9</td>
<td>253%</td>
<td>71%</td>
</tr>
<tr>
<td></td>
<td>WC → +V +BC</td>
<td>Incr.</td>
<td>DR → DR +V +BC</td>
<td>Incr.</td>
</tr>
<tr>
<td>Precision (%)</td>
<td>13.4</td>
<td>+26.5</td>
<td>198%</td>
<td>83%</td>
</tr>
<tr>
<td>Recall (%)</td>
<td>13.4</td>
<td>+26.5</td>
<td>198%</td>
<td>44%</td>
</tr>
</tbody>
</table>

Although the search used in the DR segmentation apparently bears no psychological plausibility, the experimental results from various versions of the DR segmentation, as reviewed above, suggest that the DR strategy can be a useful strategy that infants may
use to segment utterances into words in the initial state of knowing no words. These experimental results are also supportive to the hypotheses proposed in [34]:

1. Children use multiple strategies, including the DR strategy, to discover lexical units.

2. Children prefer segmentation not violating phonotactic constraints, including the vowel constraint and the boundary clusters constraint.

However, how pre-lexical infants learn such phonotactic constraints becomes an interesting and critical question for lexical learning. Brent and Cartwright hypothesise that children learn these constraints from unsegmented utterances, independently of the success of early attempts at segmentation. It is possible that children can learn boundary clusters from utterance boundaries [5].

However, the question seems to be left open whether phonotactic constraints, and other types of cues for lexical discovery, are a by-product of lexical learning, an accompanying happening with lexical learning, or a starting point of lexical learning, as assumed by Brent and Cartwright, because we do not know how many lexical units an infant has learned at the age of mastering such constraints and cues. Brent and Cartwright’s hypothesis would be validated if we can find evidence that children have mastered a significant number of constraints at the age of being able to comprehend very few words.

4.5 De Marcken’s Concatenative Model

De Marcken proposed the concatenative model as a computational theory for unsupervised language acquisition [95, 97], with a focus on learning lexicons from unsegmented (i.e., spaceless) texts and continuous speech signals. Language acquisition can be conceptually viewed, at an abstract level, as a mapping from a certain amount of evidence of a language, to which a learner has been exposed, to the knowledge of the language at various linguistic levels in the form of a grammar, in a general sense, that can be used to comprehend and produce new utterances. It is not extraordinary that in this theory language acquisition is computationally formulated as a process of fitting a stochastic, generative language model to the evidence available to the learner, through iteratively refining the structure and stochastic parameters in the model. For quite some time researchers have been trying to apply this idea to induce language models from language data in the fields of language learning, speech processing and natural language
processing. For example, Stolcke [328] formulated a best-first learning algorithm in a similar fashion in the Bayesian framework to induce context-free grammars from various sets of artificial language data. Algorithms following a general strategy of this kind for searching for the optimal grammar, in particular, those algorithms that only tune the stochastic parameters toward given language data without adjusting the model structure, usually get stuck at local optima. However, de Marcken’s algorithm appears not to suffer too much from the local optimum problem, because it goes through a stage in each iteration specifically designed for adjusting the structure of the grammar. Each iteration in de Marcken’s algorithm undergoes two stages: first, optimise the stochastic parameters of the grammar while the structure is unchanged; second, change the mode structure if such changes are predicted to lead to a better stochastic model. The representation for utterances as flat sequences of characters in the concatenative model seems to have facilitated the structure adjustment to a great extent.

In the concatenative model, a sentence is represented as a concatenation of a sequence of words, and a word can be further decomposed into shorter sequences of characters, very similar to the multigram model [20, 104, 21, 106]. Such decomposition leads to a hierarchical segmentation for an input utterance. For example, below is a small fragment of the output from de Marcken’s learning algorithm, as given in [97].

```
for the purpose of maintaining international peace and promoting
the advancement of all people the United States of America joined in
forming the United Nations
```

A possible representation for the utterance the cat sat in that hat, for instance, in the concatenative model may look like that in Table 4.5, where a lexical item has a code (e.g., a Huffman code [158]) and may, in turn, have a representation and a corresponding encoding if it is composed of other lexical items. The lexicon in this example may not be optimal. The task for the learning algorithm is to search for the optimal lexicon that has a minimum encoding cost for both the input utterances and the lexicon, following the minimum description length (MDL) principle [294, 297, 299]. That is, given the input \( U \) as a set of utterances, we look for the grammar (i.e., lexicon) \( G \) in the space \( \mathcal{G} \) of possible grammars that minimises the sum of the description length of \( U \) under \( G \).
and the description length of $G$ itself, as below:

$$
G = \arg \min_{G \in G} |G| + |U|_G \\
\approx \arg \min_{G \in G} \sum_{w \in G} |w|_G + \sum_{u \in U} |u|_G 
$$

(4.16)

where $|\cdot|_G$ is the description length of a character sequence (or string) under the grammar $G$. The approximation shows that some minor coding costs, such as for coding the number of parameters in the representation of an utterance or another linguistic parameter (i.e., a word), can be ignored. The description length for a string $u$ is defined as the negative logarithm of the probability of the utterance (under the grammar in question)$^3$, following Shannon’s source coding theorem [312]:

$$
|u| = -\log p(u) 
$$

(4.18)

and the probability of a string is defined as below, similar to that in the multigram

$^3$If necessary, to highlight the grammar in question, we can write (4.19) as

$$
|u|_G = -\log p(u|G) 
$$

We will follow the convention as in (4.19) hereafter, as the grammar in question is obvious.
CHAPTER 4. COMPUTATIONAL MODELS OF LEXICAL LEARNING

language model:

\[ p(u) = \sum_{n} p(n) \sum_{w_1 \cdots w_n \text{ s.t.}} \prod_{i=1}^{n} p(w_i) \] (4.19)

where \( n \) is the number of parameters involved in a representation for \( u \). If we assume a uniform probability distribution on positive integers (or the cost for encoding integers is insignificant in the learning process), we can overlook \( p(n) \) (or \(- \log p(n)\)). Accordingly, we have the following approximation, as was used in the original unigram model, for computing probability for a given utterance\(^4\):

\[ p(u) = \sum_{w_1 \cdots w_n \text{ s.t.}} \prod_{i=1}^{n} p(w_i) \] (4.21)

As in Olivier’s algorithm, the initial grammar in de Marcken’s learning algorithm starts with a simplest lexicon, namely, the set of the alphabetical (or terminal) symbols, with a uniform probability distribution over the symbols. Then the algorithm iterates on the refinement of the stochastic parameters and the structure (i.e., linguistic parameters, in de Marcken’s terms) of the model, using the Baum-Welch (or forward-backward) algorithms [9], as in Figure 4.1. The learning algorithm only considers the pairs of existing words as linguistic parameters to be added to the lexicon, and all existing words are considered as candidates to be removed from the current lexicon.

The forward-backward algorithm is an incarnation of the expectation-maximisation (EM) procedure of Dempster et al. [11, 107] for parameter optimisation for hidden Markov models (HMMs) involving hidden representation. De Marcken gives the forward-backward algorithm in a simpler form for unigrams, for the purpose of estimating not only the probability of a given string but also the counts of a lexical item in a given \( U + G \).

Given an utterance \( u = c_1 \cdots c_l \) as a character sequence, let the forward probability \( \alpha_2(u) \) denote the probability of the word sequence \( w_1 \cdots w_n \) being generated (over

\( \text{Consequently, the codeword length for an utterance is, again, following Shannon’s source coding theorem, approximated by} \)

\[ -\log p(u) = -\log \sum_{w_1 \cdots w_n \text{ s.t.}} \prod_{i=1}^{n} p(w_i) \] (4.20)
1. Initialisation: Let $G = \sigma$ (the set of terminals) with a uniform distribution.

2. Iterate until convergence (or up to a given number of iterations):
   (a) Stage 1:
   i. Optimise the stochastic parameters of $G$ over $U + G$ via the forward-backward algorithm, and record Viterbi representations and the co-occurrence counts of existing lexical items;
   ii. Add new lexical items to $G$ that are pairs of existing ones, if the addition of each item is predicted to decrease $|U + G|$.
   (b) Stage 2:
   i. Optimise the stochastic parameters of $G$ over $U + G$ via the forward-backward algorithm;
   ii. Delete linguistic parameters (i.e., lexical items) from $G$, if the deletion is predicted to decrease $|U + G|$.

Figure 4.1: De Marcken’s learning algorithm for the concatenative model.

u) such that $w_1 \circ \cdots \circ w_0 = c_1 \cdots c_i$. We then have\(^5\)

$$\alpha_i(u) = \sum_{j=0}^{i} \alpha_j(u) \sum_{w-c_j+1 \cdots c_i \in G, w \in U} p_G(w)$$ (4.22)

with $\alpha_0(u) \equiv 1$. Here, the probability of a lexical item $w$ is its optimal probability

$$p_G(w) = \frac{c_G(w)}{\sum_{w'} c_G(w')}$$ (4.23)

where $c_G(\cdot)$ is the expected count of a word throughout $U' = U + G$. Letting the backward probability denote the probability of the word sequence $w_1 \cdots w_0 \cdots w_n$ being generated over $u$, given that $w_1 \circ \cdots \circ w_0 = c_1 \cdots c_i$, we have

$$\beta_i(u) = \sum_{j=i}^{l} \beta_j(u) \sum_{w-c_j+1 \cdots c_i \in G, w \in U} p_G(w)$$ (4.24)

with $\beta_i(u) \equiv 1$. Consequently, we have $p_G(u) \equiv \alpha_1(u) \equiv \beta_0(u) \equiv \alpha_i(u)\beta_i(u)$ (for $i = 1, \cdots, l$). Following the independence of parameter generation in the multigram model,

\(^5\)In both (4.23) and (4.25), the condition $w \in U$ in the subscript denotes that $w$ is a “sub-string of” $u$. This condition prevents the possibility that an utterance is the representation for itself, as a single component!
the conditional probability of a lexical item \( w \) spanning \( c_{a+1} \cdots c_b \) (i.e., \( w = c_{a+1} \cdots c_b \)) in an utterance \( u = c_1 \cdots c_l \) is, as given by de Marcken, as below:

\[
p_G(a \xrightarrow{w} b|u) = \frac{c_G(u) p_G(w) \beta_b(u)}{p_G(u)}
\]  

(4.25)

Obviously, \( p_G(a \xrightarrow{w} b|u) \equiv 0 \) if \( w \neq c_a \cdots c_b \). Accordingly, the count of \( w \) under a lexicon \( G \) through a given corpus \( U \) and the lexicon can be computed as

\[
c_G(w) = \sum_{u \in U} \sum_{a=0}^{l} \sum_{b=a}^{l} p_G(a \xrightarrow{w} b|u)
\]  

(4.26)

Similarly, the count of two words, say, \( w_1 \) and \( w_2 \), that co-occur adjacently in \( U + G \) is estimated as

\[
c_G(w_1, w_2) = \sum_{u \in U'} \sum_{a=0}^{l} \sum_{b=a}^{l} p_G(a \xrightarrow{w_1,w_2} b|u)
\]  

(4.27)

where the probability of \( w_1 \circ w_2 = c_a \cdots c_b \) in an utterance \( u \) is given as below, similar to the probability of a single word in (4.26):

\[
p_G(a \xrightarrow{w_1,w_2} b|u) = \frac{c_G(u) p_G(w_1) p_G(w_2) \beta_b(u)}{p_G(u)}
\]  

(4.28)

The co-occurrence count of two words estimated by (4.28) is used in de Marcken's learning algorithm to predict whether a new word comprised of two existing words should be added to the current lexicon and whether an existing word consisting of two other words should be deleted from the current lexicon, in order to arrive at a better lexicon, i.e., a new lexicon that can shorten \( |U'| \), the combined description length of the input data and the lexicon itself. The description length\(^6 \) \( |U'| \) under a lexicon \( G \) is as

\[
|U'|_G = \sum_{u \in U'} -\log p_G(u)
\]

\[
= \sum_{u \in U'} -\log \sum_{w_1 \cdots w_n, \alpha \in \Sigma} \frac{\prod_{i=1}^{n} p_G(w_i)}{p_G(u)}
\]

\[
\approx \sum_{u \in U'} -\log \sum_{w_1 \cdots w_n, \alpha \in \Sigma} -\log \prod_{i=1}^{n} p_G(w_i)
\]

\[
= \sum_{u \in U'} -\log \sum_{w_1 \cdots w_n, \alpha \in \Sigma} \prod_{i=1}^{n} p_G(w_i)
\]

\[
= \sum_{u \in U'} -c_G(w) \log p_G(w)
\]

\[\text{De Marcken gives a lengthy and circuitous derivation as below for (4.30), where the second step uses (4.22), and the approximation results from moving the logarithm inside the summation, based on the fact that the majority of the probability of an utterance comes from the Viterbi representation of the utterance [97] (p. 83), and the last step comes from the fact that the summation is over each occurrence of each word in } U'.\]
straightforward as in (4.30), which states that the description length of the corpus is the sum of all its individual words’ description length.

\[ |U'|_G = \sum_{w \in G} -c_G(w) \log p_G(w) \]  

(4.29)

Thus, the benefit of adding (or deleting) a word \( w \) as a linguistic parameter to (or from) the lexicon is estimated as below in bits\(^7\):

\[ \Delta = |U'|_G - |U'|_{G^*} \approx \sum_{w \in G} -c_G(w) \log p_G(w) - \sum_{w \in G^*} -c_{G^*}(w) \log p_{G^*}(w) \]  

(4.30)

where \( G \) is the current lexicon, \( G^* \) is the new lexicon, and \( p_{G^*}(w) \) is computed using (4.24) in terms of the word count throughout \( U + G^* \). If a change leads to a positive \( \Delta \), the change is estimated to decrease the combined description length of the evidence and the lexicon. In de Marcken’s learning algorithm, all positive changes with \( \Delta > 0 \) are carried out simultaneously. For the sake of efficiency, the learning algorithm only considers the parameter pairs that appear in the Viterbi representations as candidates for new words, and existing words with occurrences fewer than 2 in the Viterbi representations will be deleted.

When a new word \( W = w_1 \circ w_2 \) is added, it means that all instances of \( w_1 \circ w_2 \) are replaced by \( W \) throughout \( U' \), and the updated counts of \( w_1 \) and \( w_2 \) can be estimated as

\[ c_{G^*}(w) \approx c_G(w) + c_{G^*}(w \in W) - c_{G^*}(W)c_G(w \in W) \]  

(4.31)

where the second term on the right-hand side is the count of \( w \) in the newly added \( W \) and the third term is the number of \( w \)'s reduced from \( U' \) by the replacement of \( w_1 \circ w_2 \) by \( W \). Here, the count of a word in the representation of an utterance \( u = c_1 \cdots c_l \) is defined as

\[ c_G(w \in u) = \sum_{a=0}^{l} \sum_{b \rightarrow a} p_G(a \rightarrow b|c_1 \cdots c_l) \]  

(4.32)

In practice, it suffices to use the following approximations, for the sake of simplicity and efficiency: \( c_{G^*}(W) \approx c_G(w_1, w_2) \) and \( c_{G^*}(w \in W) \approx c_G(w \in W) \). In this way,

\(^7\)Originally, in [97] \( \Delta \) represents the estimate of approximate change in description length, defined as

\[ \Delta = |U'|_{G^*} - |U'|_G \]

Accordingly, any change with \( \Delta < 0 \) is considered a positive change. De Marcken provides a more efficient way to compute (4.31) in [97] (p. 83), which deals only with the words and counts under change. We skip the technical details here.
to compute (4.32), only one pass of the forward-backward algorithm is needed for the estimation of \( c_G(w) \).

When a word \( W = w_1 \cdots w_k \) is deleted from the current lexicon, it means that all its occurrences in \( U' \) must be replaced by its representation, and accordingly \( c_{G'}(W) = 0 \) and the new count of each word in its representation is

\[
c_{G'}(w) \approx c_G(w) - c_G(w \in W) + c_G(W)c_G(w \in W)
\]  

(4.33)

where the second term on the right-hand side is the number of occurrences of \( w \) reduced by deleting \( W \) and the third term is the number of occurrences of \( w \) added by replacing \( W \) with \( w_1 \cdots w_k \) throughout \( U' \).

The total computational complexity of the learning algorithm for de Marcken’s concatenative model is \( O(ip^2e + l) \), as given in [97] (p. 86) (we will not go into the details of the analysis), where \( i, p, e \) and \( l \) are, respectively, the number of iterations during the learning, the length of the longest word in the lexicon during the learning, the size of the largest set of candidate changes to the lexicon and the length of the input corpus in characters. This complexity is linear in the length of the corpus.

The performance of de Marcken’s learning algorithm looks rather impressive on hierarchical segmentation (i.e., chunking) of both English and Chinese utterances as character sequences. When the algorithm was run on the Brown corpus [199, 126], it produced a lexicon of 26,026 words and reached a recall of 90.5% and a bracket-crossing rate as low as 1.7%. When it was run on a corpus of Chinese news texts of 4 million Chinese characters from China’s official Xinhua news agency, it produced a lexicon of 57,885 words and reached a recall of 96.9% and a bracket-crossing rate of 1.3%. The bracket-crossing rate is defined as the proportion of the true words (i.e., the orthographic words) that are crossed by any learned word. A low bracket-crossing rate indicates that the learning results are highly consistent with native speakers’ intuition about words, as reflected in the orthography.

However, as was pointed out by Brent in [32], de Marcken’s learning model does not specifically address the lexical learning problem, because it outputs a hierarchical representation for utterances without telling which chunks in the hierarchy are words and which are not, and the measures for its performance are questionable: when all chunks in the learned hierarchical representations for utterances from the Brown corpus are considered, the precision of the output is as low as 16%, i.e., only about two out of a dozen are real words, rather close to the recall of the random baseline used in Brent’s
work [34, 32], which is 13-14%.

4.6 Brent’s Probabilistically Sound Model and MBDP-1 Algorithm

Brent’s efficient, probabilistically-sound model and its implementation in the MBDP-1 algorithm [32] for segmentation and word discovery can be viewed as a successor of the previous INCDROP model – the DR model [34]. It achieves the best performance in unsupervised lexical learning from a naturally-occurring children-directed speech corpus [222], in comparison with other approaches, including transitional probability (TP) segmentation, mutual information (MI) segmentation, Elman’s simple recurrent nets [117], Olivier’s word grammars [273], LZW segmentation [346], and a pseudo-random baseline. In this sense, it reflects the state-of-the-art of computational studies on lexical learning focusing on word discovery from spontaneous speech data.

The idea of determining word boundaries based on the transitional probability can be traced back to Harris’s empirical method for segmenting phoneme strings in phonetic transcription into morphemes for an unfamiliar language [148, 149]. Harris proposed that given an utterance, the boundaries should be placed after the prefix phoneme sequences whose successor counts are not less than those of its immediate neighbours. The successor count of a prefix, following Harris’s definition, is the number of distinct phonemes that can follow that prefix. Formally put, given an available corpus C and a particular utterance \( U = u_0u_1 \cdots u_n \), the successor count of a prefix \( u_0u_1 \cdots u_i \) (for \( i = 0, 1, \cdots, n \)) is

\[
sc(u_0u_1 \cdots u_i) = \left| \{ x \mid u_0u_1 \cdots u_ix \in C \} \right| \tag{4.34}
\]

where \( \epsilon \) denotes the appearance of a string in a corpus and \( | \cdot | \) the number of distinct elements in a set. Accordingly, Harris’s idea can be formulated as this: a morpheme boundary should be put after \( u_i \) (for \( i = 0, 1, \cdots, n \)) if

\[
sc(u_0u_1 \cdots u_{i-1}) \leq sc(u_0u_1 \cdots u_i) \geq sc(u_0u_1 \cdots u_{i+1}) \tag{4.35}
\]

Harris’s example to illustrate this idea is he’s quicker, whose correspondent phoneme sequence in phonemic transcript is as below. The numbers in the second line are the successor counts after the correspondent phonemes and the dots indicate the morpheme boundaries inserted according to the counts.

\[
\begin{array}{cccccccccc}
9 & 14 & 29 & 29 & 11 & 6 & 10 & 28 & 14 & 28
\end{array}
\]
CHAPTER 4. COMPUTATIONAL MODELS OF LEXICAL LEARNING

Why put a boundary at a position where the successor count is no less than those of its immediate neighbour positions? The essence of this idea appears to be highly relevant to assigning a boundary to a position where the conditional (or transitional) probability is lower than its neighbours. It is assumed in linguistics that the dependency between elements in a word is greater than that between elements in different words. This dependency can be measured by transitional probability, which is defined as

\[ p(x|u_0u_1 \cdots u_i) = \frac{p(u_0u_1 \cdots u_ix)}{p(u_0u_1 \cdots u_i)} \frac{c(u_0u_1 \cdots u_ix)}{c(u_0u_1 \cdots u_i)} \]  \hspace{1cm} (4.36)

where \( p(\cdot) \) is the probability of a sequence and \( c(\cdot) \) the frequency of a sequence in a given corpus.

Since this probability causes many problems in real applications when the preceding sequence gets too long, e.g., the sparse data problem (i.e., many transitional probabilities are zero, because the sequence \( u_0u_1 \cdots u_ix \) has never seen before for many \( x \)'s.), it is usually approximated by (4.38), the conditional probability of \( x \) given one preceding symbol,

\[ p(x|u_i) = \frac{p(u_ix)}{p(u_i)} = \frac{f(u_ix)}{f(u_i)} \]  \hspace{1cm} (4.37)

It is suggested in [304] that language learners may hypothesise word boundaries at lower points of transitional probabilities between adjacent speech sound units (e.g., syllables). A lower transitional probability indicates a weak connection or dependency, or a higher surprisingness given its predecessor.

Another measure for the surprisingness of adjacent symbols (e.g., sound units) in a sequence (e.g., in a speech stream) is their mutual information, which is defined as

\[ \text{MI}(x,y) = \log_2 \frac{p(xy)}{p(x)p(y)} \]  \hspace{1cm} (4.38)

This is a more natural and symmetric measure for surprisingness that can be applied to predict (or detect) word boundaries. Segmentation with mutual information is straightforward, “although it has never been proposed in the language acquisition literature” [32] (p. 73). It is as simple as this: low points of mutual information of adjacent symbols are taken as word boundaries. It follows from that belief that the dependency or association of two adjacent symbols across two words is weaker than that of two adjacent symbols in the same word. This dependency is assumed to be statistically measurable, and the mutual information of two adjacent symbols is one of the measures for this kind of dependency.
CHAPTER 4. COMPUTATIONAL MODELS OF LEXICAL LEARNING

LZW [346] is a dictionary-based text compression algorithm, originally from LZ78 [364] – see Section 3.4.3 for a brief introduction.

Brent’s probabilistically sound model behind the MBDP-1 algorithm\(^8\) is based on a generation process that is proposed to have four non-deterministic steps and one deterministic step, as exemplified below:

1. Pick \( n \) – the size of a lexicon. E.g., \( n = 4 \).

2. Pick a set \( L \) of \( n \) distinct strings, \( \{W_1, W_2, \ldots, W_n\} \), from \( \sum^+ \# \), where \( \sum \) is the input alphabet and \# is word “delimiter”; and let the utterance-boundary marker \( W_0 = $ \). E.g., \( L = \{\#a, \#cat, \#saw, \#dog\} \), and

\[
\begin{array}{cccc}
W_1 & W_2 & W_3 & W_4 \\
\#a & \#cat & \#saw & \#dog \\
\end{array}
\]

3. Pick a function \( f : \{0, \ldots, n\} \rightarrow \{1, 2, \ldots\} \), where \( f(i) \) is \( W_i \)'s occurrence frequency. E.g.,

\[
\begin{array}{ccccc}
f(0) & f(1) & f(2) & f(3) & f(4) \\
1 & 4 & 2 & 2 & 2 \\
\end{array}
\]

4. Pick corpus length \( m \) and a function \( s : \{1, \ldots, m\} \rightarrow \{1, \ldots, n\} \) to map a position in the corpus to a lexical index. E.g., \( m = \sum_{i=0}^{n} f(i) = 1 + 4 + 2 + 2 + 2 = 11 \), and

\[
\begin{array}{cccccccccccc}
s(1) & s(2) & s(3) & s(4) & s(5) & s(6) & s(7) & s(8) & s(9) & s(10) & s(11) \\
1 & 2 & 3 & 4 & 1 & 4 & 3 & 1 & 1 & 2 \\
\end{array}
\]

The result of this mapping is that the word \( w_{s(i)} \) (for \( i = 1, 2, \ldots, m \)) in the output \( \tilde{w} = w_1, w_2, \ldots, w_m \) is mapped to \( W_i \) in the lexicon. E.g.,

\[
\begin{array}{cccccccccccc}
w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & w_7 & w_8 & w_9 & w_{10} & w_{11} \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
W_s(1) & W_s(2) & W_s(3) & W_s(4) & W_s(5) & W_s(6) & W_s(7) & W_s(8) & W_s(9) & W_s(10) & W_s(11) \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
W_1 & W_2 & W_3 & W_4 & W_5 & W_6 & W_7 & W_8 & W_9 & W_{10} & W_{11} \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
a# & cat# & saw# & a# & dog# & $ & a# & dog# & saw# & a# & cat# \\
\end{array}
\]

5. Concatenate \( w_1, w_2, \ldots, w_m \) together, delete #’s and output the result. E.g.,

\(\text{MBDP stands for model-based dynamic programming, according to [32].}\)
a cat saw a dog $\$ a dog saw a cat

This output is called the yield of $w_1, w_2, \cdots, w_m$.

According to these steps, given a segmentation hypothesis $\bar{\omega}_m = w_1, w_2, \cdots, w_m$ (with ’s deleted), its probability $\Pr(\bar{\omega}_m)$ is the sum over all possible outcomes from steps 1-4. That is,

$$\Pr(\bar{\omega}_m) = \sum_n \sum_L \sum_f \sum_s \Pr(\bar{\omega}|n, L, f, s) \Pr(n, L, f, s)$$

Since once a particular $\bar{\omega}_m$ is given, only the combination of $n$, $L$, $f$ and $s$ that matches $\bar{\omega}_m$, denoted as $n_m$, $L_m$, $f_m$ and $s_m$, respectively, is possible, and $\bar{\omega}_m$ is completely determined by this combination, i.e., $\Pr(\bar{\omega}_m|n_m, L_m, f_m, s_m) = 1$, thus (4.40) can be rewritten as:

$$\Pr(\bar{\omega}_m) = \Pr(\bar{\omega}_m|n_m, L_m, f_m, s_m) \Pr(n_m, L_m, f_m, s_m)$$

$$= \Pr(n_m, L_m, f_m, s_m)$$

$$= \Pr(s_m|f_m, L_m, n_m) \Pr(f_m|L_m, n_m) \Pr(L_m|n_m) \Pr(n_m)$$ (4.40)

where the last step uses the chain rule. Moreover, several linguistic assumptions allow this formulae to be further simplified:

1. The function $s$ for ordering words in the output is independent of the lexical forms in the lexicon, therefore $\Pr(s_m|f_m, L_m, n_m) = \Pr(s_m|f_m, n_m)$.

2. The length of the output $n_m$ is entirely determined by the frequency function $f_m$, so $\Pr(s_m|f_m, n_m) = \Pr(s_m|f_m)$.

3. Word frequencies are independent of the lexical forms and of the lexicon size, but dependent on the probability distribution on the frequencies of words, thus

$$\Pr(f_m|L_m, n_m) = \sum_{i=0}^{n_m} \Pr(f(i)|W_i)$$

where $\Pr_f$ is the probability distribution on the frequencies of individual words.

Accordingly, (4.41) can be rewritten into

$$\Pr(\bar{\omega}_m) = \Pr(s_m|f_m) \left[ \sum_{i=0}^{n_m} \Pr(f(i)|W_i) \right] \Pr(L_m|n_m) \Pr(n_m)$$ (4.41)
For the purpose of computing this probability, a relative probability\(^{3}\) \(R\) is defined as

\[ R(\tilde{a}_k) \equiv \frac{\Pr(\tilde{a}_k)}{\Pr(\tilde{a}_{k-1})}, \text{ with } R(\tilde{a}_0) \equiv 1 \]  

(4.42)

Following this definition, we have

\[ \Pr(\tilde{a}_k) = R(\tilde{a}_k)\Pr(\tilde{a}_{k-1}) = R(\tilde{a}_k)R(\tilde{a}_{k-1})\Pr(\tilde{a}_{k-2}) = \cdots = \prod_{i=1}^{k} R(\tilde{a}_k) \]  

(4.43)

The critical point here of making use of this formula to compute the probability for a segmentation over a given string is the estimation of the \(R(\tilde{a}_k)\). While applying (4.42) to the definition in (4.43), we have

\[ R(\tilde{a}_k) = \frac{\Pr(s_k|f_k)}{\Pr(s_{k-1}|f_{k-1})} \sum_{i=0}^{n_k} \Pr_f(f_k(i)|W_i) \frac{\Pr(L_k|n_k)}{\Pr(L_{k-1}|n_{k-1})} \frac{\Pr(n_k)}{\Pr(n_{k-1})} \]  

(4.44)

This formula can be further differentiated into two cases: \(w_k\) is a familiar word in \(\tilde{a}_{k-1}\) (i.e., \(w_k \in L_{k-1}\)), or a novel word. In the case of a familiar word, \(L_k = L_{k-1}, n_k = n_{k-1}\) and \(f_k(\hat{k}) = f_{k-1}(\hat{k}) + 1\), where \(\hat{k}\) is introduced here as a shorthand denotation for \(s(w_k)\), the type index of \(w_k\). Consequently, the nominator and the denominator in both the third and fourth term in (4.45) are identical, and all terms but the last in the nominator and the denominator in the second term are also identical. Therefore, we have (4.46) for this case.

\[ R(\tilde{a}_k|w_k \in L_{k-1}) = \frac{\Pr(s_k|f_k)}{\Pr(s_{k-1}|f_{k-1})} \frac{\Pr_f(f_k(\hat{k})|W_k)}{\Pr_f(f_k(\hat{k}) - 1|W_k)} \]  

(4.45)

In the case of \(w_k\) being a novel word (i.e., \(w_k \not\in L_{k-1}\)), we have \(n_k = n_{k-1} + 1\) (i.e., the lexical size increases by 1), \(f_k(i) = f_{k-1}(i)\) for \(i < n_k\) (i.e., the frequencies of familiar words are unchanged) and \(f_k(n_k) = 1\). In this case, (4.45) becomes

\[ R(\tilde{a}_k|w_k \not\in L_{k-1}) = \frac{\Pr(s_k|f_k)}{\Pr(s_{k-1}|f_{k-1})} \Pr_f(1|W_{n_k}) \frac{\Pr(L_k|n_k)}{\Pr(L_{k-1}|n_{k-1})} \frac{\Pr(n_k)}{\Pr(n_k - 1)} \]  

(4.46)

Then, Brent endeavoured to simplify these formulae by invoking a number of assumptions, including

\(^{3}\)It appears to be for the sake of some conceptual nicety (or subtlety) that Brent did not recognise this “conditional” probability, although he admitted that thinking of this probability ratio as a conditional probability is convenient and algebraically proper, as noted in Note 4 in [32]. Thus, Brent’s main point is this: in Brent’s model, the sequences \(w_1, w_2, \ldots, w_{k-1}\) and \(w_1, w_2, \ldots, w_k\) are two mutually exclusive events, each determined by the joint outcomes of the steps 1-4. The question is, why would the probability space, where each string has a chance to be generated by the 4 steps, conceptually disallow this conditional probability – the probability of \(w_1, w_2, \ldots, w_k\) given its prefix \(w_1, w_2, \ldots, w_{k-1}\)? Anyway, apart from this subtlety, we can simply read it off as a conditional probability, for convenience of thinking and understanding.
1. A uniform distribution on ordering functions given frequencies, which leads to:
\[
\Pr_U(s_k|f_k) = \frac{\prod_{i=0}^{n_k} f_k(i)}{k!}
\]
where \( k \) is the sum of the frequencies of all words; accordingly,
\[
\frac{\Pr_U(s_k|f_k)}{\Pr_U(s_{k-1}|f_{k-1})} = \left\{ \begin{array}{ll}
\frac{f_k(k)}{k}, & \text{for a familiar word } w_k \in L_{k-1} \\
\frac{f_k(k)}{k}, & \text{for a novel word } w_k \not\in L_{k-1}
\end{array} \right.
\]

2. A distribution on word frequencies independent of word pronunciations, i.e.,
\[
\Pr_f(f_k(\hat{k})|W_k) = \Pr_f(f_k(\hat{k}))
\]

3. A distribution on sets of pronunciations to derive an approximation for the ratio
\[
\frac{\Pr(L_k|n_k)}{\Pr(L_{k-1}|n_k-1)} \approx \frac{n_k \Pr_{\Sigma}(W_{n_k})}{1 - \frac{n_k-1}{n_k} \sum_{j=1}^{q} \Pr_{\Sigma}(W_j)}
\]

4. A distribution on individual pronunciations\(^{10}\)
\[
\Pr_{\Sigma}(a_1, \ldots, a_q) = \frac{1}{1 - \Pr_{\Sigma}(\#)} \prod_{i=1}^{q} \Pr_{\Sigma}(a_i)
\]
where \( \Pr_{\Sigma} \) is a probability distribution on \( \Sigma \cup \{\#\} \), to be estimated on-line from the relative frequencies of phonemes in the lexicon.

5. A distribution on positive integers \( \Pr(i) \equiv \frac{d}{i^2} \cdot \frac{1}{\pi^2} \), which is picked for algebraic and computational simplicity.

After applying these assumptions to (4.46) and (4.47), we have\(^{11}\)
\[
\begin{align*}
R(\bar{w}_k|w_k & \in L_{k-1}) = \frac{f_k(\hat{k})}{k} \cdot (f_k(\hat{k}) - \frac{1}{f_k(\hat{k})})^2 \\
R(\bar{w}_k|w_k & \not\in L_{k-1}) = \frac{6}{\pi^2} \cdot \frac{n_k}{k} \cdot \frac{\Pr_{\Sigma}(W_{n_k})}{1 - \frac{n_k-1}{n_k} \sum_{j=1}^{q} \Pr_{\Sigma}(W_j)} \cdot (\frac{n_k - 1}{n_k})^2
\end{align*}
(4.48) (4.49)
\]

The terms in these two formulae have some interpretations that are worth noting.
In (4.49), the first term is the relative frequency of the new word \( w_k \), which appears to

\(^{10}\)Alternatively, this distribution can be represented as
\[
\Pr_{\Sigma}(W) = \frac{1}{1 - \Pr_{\Sigma}(\#)} \prod_{a \in W} \Pr_{\Sigma}(a)
\]
(4.47)

\(^{11}\)We do not present the lengthy derivation for this result step by step in this short review. The details can be found in [32]. Notice that in (4.50), \( w_k \) and \( W_{n_k} \) appear to be identical strings.
have the add-one smoothing effect [129, 352] (i.e., the effect of adding one to both the numerator and the denominator in the formula \( p(x_{j+1}|x_{i,j}) = c(x_{i,j}x_{j+1})/c(x_{i,j}) \) for probability estimation), and the second term approaches 1 rapidly when the frequency of the word in question increases, starting from one fourth for a word with frequency 1 so far. In (4.50), the first term is a constant for normalisation, the second term can be thought of as a type-token ratio, the third is the probability for a particular novel word to be randomly chosen during the generation of the lexicon, which tends to be lower when the novel word gets longer, and the fourth approaches 1 rapidly when the number of word types in the lexicon increases, starting from one fourth for the first word type.

Based on (4.49) and (4.50), which provides the means for computing the prior probability for a segmentation of an utterance, Brent proposed two search algorithms for searching the optimal segmentation of an input corpus, namely, an incremental search and a Viterbi search with dynamic programming – an approximation for the incremental search.

### 4.6.1 Incremental Search

Given that the optimal segmentation of the previous utterances in the input is \( \tilde{w}_m \), and the current utterance is \( w_{m+1} \cdots w_{m+p} \), the incremental search evaluates the probabilities of all word sequences \( \tilde{w}_{m+p} \) using the following formula for each sequence, which follows from (4.44):

\[
Pr(\tilde{w}_{m+p}) = Pr(\tilde{w}_m) \prod_{i=1}^{p} R(\tilde{w}_{m+i})
\]

The search chooses the word sequence with the highest probability as the segmentation for the current utterance. Then, we have the optimal segmentation \( \tilde{w}_{m+p} \), and move on to work on the next utterance, until the whole corpus is entirely segmented.

The incremental search works on one utterance at a time. An utterance of length \( l \) has \( 2^{l-1} \) possible word sequences. Thus, the complexity for segmenting a corpus of \( n \) utterances of lengths \( l_1, \cdots, l_n \) is \( O(\sum_{i=1}^{n} 2^{l_i-1}) \). This complexity is perhaps tractable by current computers if each \( l_i \) in the input is not too large, but a more efficient algorithm is available using the dynamic programming technique.

### 4.6.2 Viterbi Search with Dynamic Programming

The Viterbi algorithm is an approximation to the incremental search, aimed at finding the optimal segmentation for an upcoming utterance without evaluating all possible
CHAPTER 4. COMPUTATIONAL MODELS OF LEXICAL LEARNING

segmentations. Given the optimal segmentation for previous utterances of the input corpus $\bar{\sigma}_m$, the following formula\(^\text{12}\) is used to approximate (4.51):

$$\Pr(\bar{\sigma}_{m+p}) \approx \Pr(\bar{\sigma}_m) \prod_{i=1}^{p} R(\bar{\sigma}_m w_{m+i})$$

(4.51)

where $R(\bar{\sigma}_m w_{m+i})$ substitutes for $R(\bar{\sigma}_m w_{m+1} \cdots w_{m+i})$. This approximation is reasonable when $m \gg p$ – in this case, $m+i \approx m$, $L_{m+i} \approx L_m$, $n_{m+i} \approx n_m$ and $f_{m+i} \approx f_m$ (for $1 \leq i \leq p$). It leads to an efficient Viterbi search algorithm with the dynamic programming technique as the following: Given the current utterance as a symbol sequence $s_0s_1 \cdots s_i$,

1. Starting from $i = 0$, the best segmentation for $s_0 \cdots s_i$, denoted as $\text{seg}(s_0 \cdots s_i)$, is found;

2. For $s_0 \cdots s_{i+1}$, select the best one among $\text{seg}(s_0 \cdots s_j)[s_{j+1} \cdots s_{i+1}]$ (for $j = 0, 1, \cdots, i$)\(^\text{13}\) as the optimal segmentation, using the criterion set in (4.52); that is\(^\text{14}\):

$$\text{seg}(s_0 \cdots s_{i+1}) = \arg \max_{j \in \{0, \cdots, i\}} \left[ \prod_{w \in \text{seg}(s_0 \cdots s_j)} R(\bar{\sigma}_m w) \cdot R(\bar{\sigma}_m [s_{j+1} \cdots s_{i+1}]) \right]$$

(4.52)

3. Repeat these steps until the best segmentation for the entire $s_0s_1 \cdots s_i$ is found.

This algorithm takes at most $l^2$ steps to work through an utterance of length $l$. For a corpus of $n$ utterances of lengths $l_1, \cdots, l_q$, its complexity is $O(\sum_{i=1}^{n} l_i^2)$. Brent called his implementation of this Viterbi algorithm MBDP-1, and viewed it as an implementation of the INCDROP proposed in [31, 93].

Tested on the Bernstein-Ratner corpus [18] of 9790 utterances and 33387 words taken from the CHILDES collection [222], MBDP-1 reached its peak of performance at a precision slightly higher than 80% and a recall slightly lower than 80% after 9500 utterances.

\(^{12}\) With regard to Brent’s implementation of this Viterbi search, $\bar{\sigma}_m$ only plays the role as a background (i.e., providing the values of $L_m$, $k$, $n_k$ and $f_k$) for inferring words $w_{m+i}$ (for $i = 1, \cdots, p$). It would be more natural to put it in the following form:

$$\Pr(\bar{\sigma}_{m+p}) \approx \Pr(\bar{\sigma}_m) \prod_{i=1}^{p} R(w_{m+i} | \bar{\sigma}_m)$$

\(^{13}\) Here $\text{seg}(s_0 \cdots s_j)$ denotes a word sequence over $s_0 \cdots s_j$ and $[s_{j+1} \cdots s_{i+1}]$ denotes the hypothesis for $s_{j+1} \cdots s_{i+1}$ to be a word.

\(^{14}\) A more readable form for this formula could be:

$$\text{seg}(s_0 \cdots s_{i+1}) = \arg \max_{j \in \{0, \cdots, i\}} \left[ \prod_{w \in \text{seg}(s_0 \cdots s_j)} R(w | \bar{\sigma}_m) \cdot R([s_{j+1} \cdots s_{i+1}] | \bar{\sigma}_m) \right]$$
CHAPTER 4. COMPUTATIONAL MODELS OF LEXICAL LEARNING

was input, and the average precision and recall were both about 71\((\pm 2)\)%, estimated according to the performance charts provided in [32]. The orthographic transcripts of the corpus were transcribed into phonemic transcripts, with every occurrence of the same word transcribed into the same phonemic form and with onomatopoeia (e.g., bang) and interjections (e.g., oh and huh) removed.

MBDP-1 appeared to outperform all other algorithms re-implemented by Brent and coworkers for comparisons, including the mutual information (MI) algorithm, Olivier’s algorithm, the transitional probability (TP) algorithm, Elman’s connectionist algorithm, the LZW algorithm, and a random baseline, with regard to the traditional precision and recall criteria. The performance of all these algorithms at the end of the input – most algorithms reach their top performance at this point – is given in Table 4.6, where the numbers in brackets indicate the ranking order of these algorithms. Most algorithms, except the last two in the precision ranking, have a segmentation precision going up from the start (i.e., the first block of 500 utterances) to the end of the input; whereas the first three algorithms in the recall ranking have a segmentation recall going up from 44%, with the rest going down.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>MBDP-1</th>
<th>MI</th>
<th>Olivier</th>
<th>TP</th>
<th>Elman</th>
<th>LZW</th>
<th>Rand</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision (%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start</td>
<td>62</td>
<td>53</td>
<td>38</td>
<td>34</td>
<td>30</td>
<td>27</td>
<td>13</td>
</tr>
<tr>
<td>End</td>
<td>80 (1)</td>
<td>60 (2)</td>
<td>53 (3)</td>
<td>50 (4)</td>
<td>42 (5)</td>
<td>24 (6)</td>
<td>14 (7)</td>
</tr>
<tr>
<td>Median</td>
<td>71</td>
<td>56.5</td>
<td>45.5</td>
<td>42</td>
<td>36</td>
<td>25.5</td>
<td>13.5</td>
</tr>
<tr>
<td>Recall (%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>33</td>
<td>13</td>
</tr>
<tr>
<td>End</td>
<td>80 (1)</td>
<td>63 (2)</td>
<td>38 (5)</td>
<td>55 (3)</td>
<td>43 (4)</td>
<td>17 (6)</td>
<td>14 (7)</td>
</tr>
<tr>
<td>Median</td>
<td>72</td>
<td>53.5</td>
<td>41</td>
<td>49.5</td>
<td>43.5</td>
<td>25</td>
<td>13.4</td>
</tr>
<tr>
<td>Lexical Precision (%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start</td>
<td>37</td>
<td>27</td>
<td>33</td>
<td>27</td>
<td>27</td>
<td>18</td>
<td>15</td>
</tr>
<tr>
<td>End</td>
<td>53 (1)</td>
<td>18 (3)</td>
<td>23 (2)</td>
<td>15 (4)</td>
<td>13 (5)</td>
<td>8 (6)</td>
<td>6 (7)</td>
</tr>
<tr>
<td>Incr.</td>
<td>+43%</td>
<td>-33%</td>
<td>-30%</td>
<td>-44%</td>
<td>-52%</td>
<td>-55%</td>
<td>-60%</td>
</tr>
<tr>
<td>Median</td>
<td>45</td>
<td>25.5</td>
<td>25</td>
<td>21</td>
<td>20</td>
<td>13</td>
<td>10.5</td>
</tr>
</tbody>
</table>

Table 4.6: Segmentation precision/recall and lexical precision of MBDP-1 and the other six algorithms re-implemented by Brent and coworkers for comparison. All data in the “Start” and “End” rows in this table are read by eye from the performance charts in [32], with about ±5% accuracy.

The lexical precision is computed in terms of word types in the learned lexicon instead of word tokens in the segmentation output. We can see from the table that MBDP-1 is the only algorithm that has a lexical precision going up from the start to the end of the input, by 43%; whereas all other algorithms have a lexical precision going down, by
CHAPTER 4. COMPUTATIONAL MODELS OF LEXICAL LEARNING

various percentages. The lexical recalls of these algorithms, including MBDP-1, were not reported in [32].

In contrast to Olivier’s algorithm, where possible segmentations compete with each other in terms of the product of the relative frequencies of words in a segmentation and, therefore, the segmentations with fewer words, thus longer words, are strongly favoured without any penalty, MBDP-1 tends to balance the two factors, namely, the product of probabilities of words and the number of words in a segmentation, because in Brent’s model, “... the probability that a given word will be generated in the lexicon (Prσ) decreases rapidly with the length of the word” (p.98). As is shown in (4.48), the length of a word lowers its probability in Brent’s probabilistic (or probabilistically-sound) model. This is claimed to be the main factor to account for the outstanding performance of Brent’s algorithm.

Brent’s model demonstrates the state of the art of computational studies on lexical learning. Brent and his co-workers’ psycholinguistic exploration [93] also shows that the model can predict adult’s behaviours in learning words from an artificial language, e.g., an utterance with no familiar words in it will be treated as a single novel word. However, whether Brent’s model can predict pre-linguistic infants’ behaviours in lexical learning from natural language data is still an interesting issue to be explored. It looks to be a fascinating edge of research to further investigate into the psycholinguistic plausibility of the model.

It is observed from Brent’s model that the formulae for estimating the probability ratios for the familiar and novel word, in (4.49) and (4.50), respectively, look a bit divergent from each other, in the sense that the latter rests on the distribution Prσ (which is to be estimated by the distribution PrΣ over alphabetical symbols), whereas the former rests entirely only on the word’s occurrence frequencies in the past. This appears to result from the few assumptions that Brent made to derive the formulae. Consequently, the balancing factor, which Brent pointed out as an advantage of his model over Olivier’s model, to balance the product of relative frequencies and the word lengths in a segmentation have effect only on novel words, but not on familiar words. So, a question we have here is: is this kind of divergence the main factor that leads Brent’s model to the success? If we could get a more consistent estimation of the probability ratios for both familiar and novel words, would that be more psycholinguistically plausible in terms of human learners?
CHAPTER 4. COMPUTATIONAL MODELS OF LEXICAL LEARNING

A rather noticeable aspect of the MBDP-1 algorithm is that in comparison with its INCDROP predecessor (e.g., the DR model), MBDP-1 does not make use of any phonotactic constraints but achieves better performance. This fact may lead us to rethink (1) the origin of the phonotactic heuristics for word identification – is it possible that such heuristics are the result (more precisely, the “by-product”) of the statistical induction for linguistic patterns (e.g. words) during lexical learning? – and (2) the role that the phonotactics may play in lexical learning – must a learner necessarily be equipped with certain phonotactic knowledge at the starting point of the learning in order to succeed in lexical learning? Brent’s algorithm seems to have shown us that even without any phonotactic constraints, the machine learner which relies entirely on distributional statistics can carry out the learning task extraordinarily well.

Finally, what we also have learned from the MBDP-1 algorithm, in particular, when we compare it with Olivier’s algorithm, is that their search algorithms are highly similar: both are Viterbi algorithms with similar dynamic programming techniques, but guided by different criteria. This tells us that a better guiding (or goodness) criterion for the search plays a fundamental role in the lexical learning, which is formulated as an optimisation process in terms of a predefined (or selected) goodness criterion (or objective function). The most significant implication of this is the possibility that looking for a better goodness criterion may lead to a better learning performance and would, therefore, get closer to the true mechanism underlying human lexical learning.

4.7 Summary

In all the representative computational studies on lexical learning that we have reviewed in this chapter, the learning problem is formulated as an optimisation problem with respect to some objective function (or goodness criterion). The optimal segmentation of the input utterances as the result of the optimisation process turns out to comprise chunks that have a very nice correspondence, in a restricted sense, to linguistic lexical units, as demonstrated, particularly, in the outputs from de Marcken’s and Brent’s learning algorithms. The Viterbi algorithm is commonly used for searching for a sound answer to an optimisation problem. Although very few researchers explicitly spell out why the criteria they chose for the optimisation can lead to chunks that correspond so well to linguistic structures (in particular, words), a general mechanism underlying most researchers’ lexical learning models is observed to be that the learning is actually a process to search for the most economic (or least-effort) representation for the input
data. One indication of this is that seeking for the least-effort representation for the
input is the common principle underlying both machine and human lexical learning.
Another indication, as evidenced by so many machine learning models, is that any
learner capable of looking for the least-effort representation for the data is capable of
digging out the lexical units embedded in the data – we dub this digging process *lexical
learning*.

The performance of a learning algorithm appears to be determined mainly by three
factors. The first one is a *representation* scheme that determines the learner’s search
space – if this space does not include the true answer, the learner, no matter what
searching algorithm it is equipped with, would not be able to get the true answer (if
exists), but might reach the one closest to it (if the goodness criterion can guide it to
the closest one). The second factor is a *search* algorithm – this determines whether or
not the best answer in the search space can be found in a reasonable time. The last
factor is a *goodness criterion*, which guides the learner to the best answer in the search
space in terms of this particular criterion. A criterion is said to be better if it leads the
learning output to match more standard answers (i.e., true words in lexical learning) in
empirical evaluation, when all other factors are equal.

In the next chapter, we will have a thorough discussion on the number of n-grams
in a given corpus, which is closely related to the first two issues we have highlighted
above. Since each n-gram in the data is a possible word, we need to have a clear idea,
before we start off to deal with the learning algorithm, about how many n-grams there
are for the learning algorithm to manipulate for the purpose of inducing an optimal set
of n-grams as the learning result – a lexicon that gives the least-effort representation
for the input. In Chapter 6, we will formulate a new goodness measure specifically for
lexical learning as searching for the least-effort representation for the data. In Chapter
7, we will present the learning algorithms using this goodness measure.
Chapter 5

N-grams and the Virtual Corpus

5.1 Overview

This chapter presents an efficient technique, based on the suffix array data structure, for deriving and retrieving n-grams of arbitrary length from a large-scale corpus. It is the operational basis for the implementation of the unsupervised lexical learning algorithm that will be formulated in Chapter 7. Since every n-gram item in a given corpus as the input for lexical learning is possibly a word, the learner needs to be equipped with an efficient technique to retrieve, and then examine, all n-gram items in the input, to see which ones should be put into the lexicon. The goodness criterion for selecting word candidates will be formulated in the next chapter. In this chapter, we focus on the derivation of n-gram items and their counts from a large-scale natural language corpus.

The popular n-gram language models are not deeply involved in our lexical learning. But some recent research on multigram models has been highly relevant to our work. Our unsupervised lexical learning is not concerned with any profound issues in n-gram models, e.g., the smoothing techniques to deal with the sparse data problem. It only has to do with a rather “shallow” and specific technique – we need a very efficient approach to counting and retrieving n-gram items of any length from a large-scale natural language corpus for the purpose of selecting lexical candidates during the learning process.

Counting fixed-length n-grams appears to be a trivial task. But counting and retrieving all n-grams of interest of arbitrary length from a large-scale corpus are not, especially when the counting and retrieval are expected to be adequately efficient to facilitate unsupervised language learning, which is known to be computationally expensive. For the purpose of acquiring a lexicon as a language model for a given corpus, it is necessary to examine all fragments (i.e., all n-gram items) of each utterance as a
sequence of some linguistic symbols (e.g., characters, phonemes) in the input, in order to decide which fragments are more likely to be words. There is a strong demand for a very fast approach for accessing all n-grams.

We adopt a position indexing technique, known as suffix array, to implement a system, called Virtual Corpus (VC), for fast accessing of n-grams in a large-scale corpus. An efficient implementation of this technique is essential to our work on unsupervised lexical learning. Nagao and Mori’s implementation, as reported in [260], had some inefficient factors— it takes an hour to derive n-grams of various lengths from a Japanese text of 3.7M bytes (i.e., 1.85M characters). Stolcke and Segal [329, 328] present an approach to deriving n-grams from a stochastic CFG which takes 9 hours on a Sun SPARCstation to finish the job of computing bi-gram probabilities from an SCFG with 133 non-terminals. Since accessing n-gram items is an essential stage of the lexical learning, it deserves all our efforts to look for a very fast approach to counting and retrieving n-grams of any length to facilitate the learning.

It is also reasonable to expect that making all n-grams efficiently accessible can bring certain benefits to other people’s work on n-gram models as well. At least, there can be more choices available for constructing various n-gram models. For example, one can try different n-gram models, for example, mixed-order n-gram models and long distance bi-gram models, and then select the best one for particular applications.

Section 5.2 have a brief discussion of n-gram models, with a focus on an introduction to multigram models. Section 5.3 analyses the number of n-grams in a large-scale corpus that need to be dealt with in n-gram models and in lexical learning. Section 5.4 presents the implementation details of the VC system, based on [193, 196], and the statistical facilities it provides. Section 5.5 concludes the chapter.

5.2 N-gram Language Models

It is known that n-gram models are the simplest, most durable and successful class of statistical models for NLP and speech processing applications. Many high performance part-of-speech (POS) taggers are based on an n-gram model using n-gram statistics in a training corpus, e.g., Leech et al. [216], Church [69] and Charniak et al. [57], among many others.

The general assumption when using an n-gram model is that the chance for the current symbol (or token, e.g., word, character) to appear depends only on its $n - 1$ preceding symbols (or tokens). This is exactly the assumption for a Markov chain, if the
sequence of the \( n - 1 \) preceding symbols is taken as a state in the chain. In other words, n-gram models assume that the token sequences (or other linguistic element sequences) in the utterances of a natural language forms a Markov chain. Straightforwardly, n-gram models are special cases of Markov models. Hidden Markov models, the most popular language models for speech processing, are an extension of Markov models. In the field of NLP, Kupiec [204] demonstrates a successful HMM approach to POS tagging, where the model is trained on an untagged corpus.

In this section, we will give a brief introduction to n-gram models, with an emphasis on the multigram model, which is highly relevant to our lexical learning.

### 5.2.1 Fixed-order n-gram Models

A fixed-order n-gram model of order \( n \) is defined as \( (5.1) \), to assign a probability to an utterance \( u = w_1w_2 \cdots w_N \) (also denoted as \( w_{1..N} \) for simplicity) as a sequence of words or linguistic elements of some other type (e.g., characters).

\[
p(u) = \prod_{i=1}^{N} p(w_i|w_{i-n+1} \cdots w_{i-1})
\]

(5.1)

where \( p(w_i|w_{i-n+1} \cdots w_{i-1}) \) (also denoted as \( p(w_i|w_{i-n+1..i-1}) \)) is the probability of \( w_i \) given its \( n - 1 \) preceding words (or tokens), which is to be estimated, basically, in terms of the counts of \( w_{i-n+1..i} \) and \( w_{i-n+1..i-1} \) in a training corpus, as given in \( (3.13) \). Here, \( w_{i-n+1..i-1} \) is defined to refer to \( w_{i..i-1} \) if the subscription \( i - n + 1 < 1 \) (i.e., \( i < n \)).

The most popular n-gram models are trigram and bigram models, as defined in \( (3.16) \) and \( (3.17) \), repeated below for convenience.

\[
p(w_{1..N}) = \prod_{i=1}^{N} p(w_i|w_{i-2\ldots i-1})
\]

\[
p(w_{1..N}) = \prod_{i=1}^{N} p(w_i|w_{i-1})
\]

To use an n-gram model, one needs to estimate the probabilistic parameters in the model, i.e., all possibilities \( p(w_i|w_{i-n+1..i-1}) \) for all strings \( w_{i-n+1..i} \) involved. This process of parameter estimation is also known as training in language modelling. Given a vocabulary \( V \) of all words involved, there are \( |V|^n \) such parameters for a given order \( n \), because there are \( |V|^n \) possible strings of length \( n \) over \( V \).

For n-gram models, the first step of training is relatively simple: count all n-gram items from a given corpus of representative data. If the training corpus consists of an
adequate amount of data and the data are typical for the application that the model is
designed for, an n-gram model trained on it can be expected to have a rather good per-
formance, e.g., Church’s POS tagger [69], which has popularised the tagging technique
with n-gram models in the field of NLP in the past decade.

In the simplest training for an n-gram model, the probability \( p(w_i|w_{i-n+1:i-1}) \) is
estimated as the relative frequency \( f(w_i|w_{i-n+1:i-1}) \) in the training corpus, which is
defined as

\[
f(w_i|w_{i-n+1:i-1}) = \frac{c(w_{i-n+1:i})}{c(w_{i-n+1:i-1})}
\]  

(5.2)

where \( c(\cdot) \) denotes the count of an n-gram in the training corpus. This method of
parameter estimation is known as the maximum likelihood estimator (MLE).

There are some other sophisticated training methods to estimate the probabilistic
parameters for a language model, e.g., the EM algorithms that we have had some dis-
cussion of in previous chapters. There are also many sophisticated techniques to soothe the
notorious sparse data problem that n-gram models unavoidably encounter [164]. Brown
et al. [41] presents a class-based n-gram model that also can alleviate the sparse data
problem to a significant extent. In Section 5.3, we will have some formal analysis of why
the sparse data problem is so severe in language modelling.

### 5.2.2 Mixed-order n-gram Models

One of the main purposes of using variable order n-grams in one language model is to
alleviate the sparse data problem. As discussed before, the central point of language
modelling is that we estimate the probability of a long string with the probabilities of its
sub-strings that are reliable. The unreliable, especially the unavailable, long n-grams’
probabilities can be adjusted using those of the lower order n-grams. In this sense, the
interpolated trigram model for linear smoothing defined as (3.19) in [163] can be thought
of as a mixed-order n-gram model. The *polygram model* proposed by Kuhn et al. [203]
also interpolates the shorter n-grams’ probabilities for the estimation of the probability
for a long n-grams, as an attempt to account for the statistical dependency between
variable-length n-grams. But a mixed-order n-gram model makes a different assumption
from that of a fixed-order n-gram model speculating the statistical dependency between
symbols within a fixed length \( n \).

Here, we introduce the *multigram model*, a mixed-order n-gram model proposed by
Bimbot and his colleagues for speech processing [20, 21, 104, 106], which assumes that an
utterance as a string of linguistic symbols (e.g., characters, words) can be decomposed
into variable-length sub-strings (e.g., words, phrases) that are assumed to be independent of each other, and accordingly, the probability of the whole utterance is the sum of the probabilities of all possible decompositions (i.e., segmentations). The probability of each segmentation is the product of the probabilities of the “independent” sub-strings within the segmentation. Given an utterance \( u = w_1 w_2 \cdots w_n \), its probability is defined as below in a multigram model:

\[
p(u) = \sum_{S \in S(u)} \prod_{s_i \in S} p(s_i)
\]  

(5.3)

where \( S(\cdot) \) denotes the set of all possible segmentations (i.e., decompositions) of an utterance, and \( s_i \)'s are individual sub-strings in each segmentation. Here, and hereafter, we use \( \in \) as the denotation for both the element-set relation (as in \( S \in S(u) \)) and the substring-string relation (as in \( s_i \in S \)).

A multigram also has an order \( n \) – the length of the longest n-gram used in the model. An n-multigram model defines the most likely segmentation as below

\[
S_{ML}(u) = \arg \max_{S \in S(u)} \prod_{s_i \in S, |s_i| \leq n} p(s_i)
\]  

(5.4)

where \( p(s_i) = 0 \) for any \( s_i \) with \( |s_i| > n \). When this formula is recursively applied to each individual \( s_i \) until \( s_i \) becomes an individual symbol, we have a hierarchical segmentation – the research that de Marcken [97] pursued on lexical learning, with arbitrary length sub-strings allowable in his model. In order to estimate the set of probabilistic parameters for each \( s_i \) with a maximal likelihood on the training data, Binbot and colleagues provide a forward-backward algorithm in [104, 106]. De Marcken also formulates a forward-backward algorithm for multigram models (see Section 4.5 for a brief introduction).

It is reported that multigram models outperform the conventional n-gram models in terms of the perplexity measure. More interestingly, when an n-multigram model (with \( n = 5 \)) is trained with an appropriate pruning factor (a way to give penalties to low frequency sequences in multigram modelling) on the ATIS data set [223], a set of spontaneous utterances about flight ticket booking and purchasing, it groups words into syntactically and/or semantically meaningful word sequences [104]. For example, it acquires patterns like depart from <city> and from <city> to <city>. Recently, it was reported in [105] that a 2-multigram model as a statistical language model on the ATIS data set allows a reduction of 10% of the word error rate in the speech recognition
on the ATIS data with respect to a usual trigram model, using 25% fewer probabilistic parameters than those in the trigram model. Also, a 5-multigram model can detect many words and meaningful character sequences in Old Testament texts with word delimiters (namely, the spaces) skipped [21]. For example, it infers the following sub-strings

therefore the un godly shall not stand in the judgment nor
sin n ers in the co ngreg a tion of the right eous
from the spaceless input

therefore the un godly shall not stand in the judgment nor sinners in the congregation of the righteous

However, such results appear to be a significant distance from those that are output from a good lexical learner, according to what we have reviewed in Chapter 4. We can also see that imposing 5 as an order upon a multigram model also prevents it from detecting any word longer than 5 characters.

5.3 Number of N-grams

Up to now only the low order n-grams such as unigram, bigrams and trigrams, rather than any higher order n-grams, were used in fixed-order n-gram models for NLP and speech processing applications. There are many reasons for this. One is that these low order n-grams can serve many applications quite well to a certain extent. Another one is that using higher order n-grams is computationally expensive, because there are too many probabilistic parameters that need to be estimated. Given a model order \( n \) and a vocabulary \( V \), the number of the parameters in the model, i.e., the number of all possible n-grams over \( V \), is \( |V|^n \). When \( n \) increases, this number goes up exponentially.

How many n-grams do we need to deal with in our unsupervised lexical learning based on n-gram statistics? Notice that to an unsupervised learner, a word can be of any length. A simple but true answer is, we need to deal with all n-grams of any length, especially with a frequency \( \geq 2 \), in the input corpus. If the number of all these n-grams in a corpus also increases exponentially with the corpus size, our lexical learning that needs to examine all n-grams would have a severe problem in terms of its computational complexity.

In this section, we will take a close look at the number of n-grams in a given corpus, in order to get a clearer idea about how many n-grams we need to deal with for the lexical learning. The number of n-grams involved reflects the most crucial aspect of the complexity of the lexical learning algorithm. We are also particularly concerned with
the issue of how to derive and retrieve all n-grams adequately fast. But if the number of n-grams is really exponentially large, there would be no way to do it in polynomial time.

Actually, counting n-grams is not as difficult as one is often said: there are $|V|^n$ n-gram items to count in a corpus for a given $n$. This is a misleading paradox. It is only true if $n$’s and $V$ are small and the given corpus is large enough to cover all $|V|^n$ n-gram items. This rarely happens in practical situations.

The number of all n-grams in a given corpus $C$ (of length $|C|$) must be at most linear to $|C|$, since an n-gram item, no matter what the $n$ is, must start from a position in the given corpus. There are in total only $|C| - (n - 1)$ such positions for n-gram items of order $n$, since the $n - 1$ positions with a shorter distance than $n$ to the end of the corpus should not be counted. Thus, the total number of n-gram items of any $n$ in the corpus should be of the order $O(|C| - n)$. This number is not exponential but only linear to $|C|$. This observation can be expressed as the following:

**Observation 5.1** Given a corpus $C$ with a vocabulary $V$ and any $n > 0$,

$$\{|x \mid x \in V^n \land x \in C\}| \leq |C| - n + 1. \quad (5.5)$$

Following this straightforward observation, we have a number of corollaries below about the number of n-grams in a given corpus.

**Corollary 5.1.1** Given a corpus $C$ with a vocabulary $V$, for any $n > 0$,

$$\sum_{k=1}^{n} |\{x \mid x \in V^k \land x \in C\}| \leq n|C|. \quad (5.6)$$

**Proof** For any $n > 0$,

$$\sum_{k=1}^{n} |\{x \mid x \in V^k \land x \in C\}| \leq n|C| - \sum_{k=1}^{n} k + n$$

$$\leq n|C|$$

where the first step follows directly from (5.5).

**Corollary 5.1.2** Given a corpus $C$ with a vocabulary $V$ and $n > 0$,

$$|\{x \mid x \in V^n \land x \not\in C\}| \geq |V|^n - |C|. \quad (5.7)$$
Proof For any $n > 0$,

$$\begin{align*}
\{|x \mid x \in V^n \land x \notin C\} &= |V|^n - \{|x \mid x \in V^n \land x \in C\} \\
&\geq |V|^n - |C| + n - 1 \\
&\geq |V|^n - |C|
\end{align*}$$

where the second step follows from (5.5).

Corollary 5.1.1 states that there are fewer than $n|C|$ n-grams of length up to $n$ in the corpus $C$; Corollary 5.1.2 states that there are at least $|V|^n - |C|$ n-gram items of length $n$ not in the corpus $C$. It is rather straightforward that these two corollaries simply follow from Observation 5.1. Next, we have another corollary to state, that there are more than $|V|^n - n|C|$ n-gram items of length up to $n$ not in the corpus $C$ on the vocabulary $V$.

Corollary 5.1.3 Given a corpus $C$ with a vocabulary $V$ and $n > 0$,

$$\sum_{k=1}^{n} \{|x \mid x \in V^k \land x \notin C\} > |V|^n - n|C|. \quad (5.8)$$

Proof Since

$$\sum_{k=1}^{n} |\{|x \mid x \in V^k\}| = \sum_{k=1}^{n} |V|^k > |V|^n,$$

we have

$$\sum_{k=1}^{n} \{|x \mid x \in V^k \land x \notin C\} > |V|^n - \sum_{k=1}^{n} \{|x \mid x \in V^n \land x \in C\}$$

$$> |V|^n - n|C| + \sum_{k=1}^{n} k - n$$

$$> |V|^n - n|C|$$

where the second step follows from (5.5).

These corollaries look plain and perhaps even superficial, in a sense. Although little attention has been drawn to them so far, they are by no means trivial. Their significance is multi-fold. First, they show how severe the sparse data problem will be when $n$ gets larger. With reference to Corollary 5.1.3, we can see that dealing with small counts, especially zero counts, by smoothing is so costly in higher order n-gram models, since the number of zero count n-grams (i.e., n-grams not in the training corpus) increases
CHAPTER 5. N-GRAMS AND THE VIRTUAL CORPUS

exponentially. Enlarging the training corpus appears not to be a valid approach to alleviating this problem, since the number of n-grams in any corpus, no matter how large it is, can only grow linearly to the corpus size. This is a simple but exact and formal explanation for why the higher order n-gram models are unfavourable in language modelling practice.

More interestingly, the above viewpoint from these plain corollaries is also in agreement with several researchers’ empirical observations on low order n-grams when the increase of vocabulary size along with the corpus size is taken into account – it is natural that the vocabulary size \(|V|\) goes up when \(|C|\) increases, because new words come in. For example, Gale et al. [130] observe that a bigram language model has a worse sparse data problem when trained with a larger data set because \(|V| > O(\sqrt{|C|})\). Sampson [306] also has a similar observation on the variety of noun phrases in a corpus. Dunning [111] points out the unreliability of small counts on the majority of word types and proposes to apply the likelihood ratio test to remedy the problem. All these insights confirm the view in the corollaries that enlarging the training corpus cannot really resolve the sparse data problem. Zipf’s law [361, 362], which states that the frequency is approximately proportional to the inverse rank, also leads to a similar conclusion: no matter how large a corpus is, there is a large portion of words (or word sequences) which appear rarely in the corpus.

Furthermore, Corollary 5.1.3 implicates a right way to make use of long n-grams: use the available long n-grams, especially the significantly frequent ones from which your language model can benefit; don’t bother too much with the rare ones, there are too many. This implication is in accordance with the recent development of n-gram models in the multigram direction. Smoothing in language modelling is a difficult issue, which attempts to achieve reasonable adjustment of the low frequencies (e.g., zero frequencies) of short n-gram items (e.g., bi-grams and tri-grams) in the incomplete data. But our corollaries seem to forecast a serious unconquerable difficulty. Progress can be made in smoothing, to improve a language model to a certain extent, but the problem can never be completely solved. A better direction for exploring a good solution might be to compute the probability of longer strings with those of the shorter strings in them – that is, we need to return to the starting point of language modelling to explore a solution.

What is more interesting to us in these corollaries in relation to lexical learning is that they help estimate the amount of work on counting and deriving n-grams from
a large-scale natural language corpus. This amount is estimated as \( O(n|C|) \), which is linear to the corpus size and to the maximal length of the n-grams involved. This is specifically relevant to the task of lexical learning based on n-gram statistics. If the amount of work of counting n-grams were at the level of \( O(|V|^n) \), there would be no appeal to using n-gram statistics.

More importantly, these corollaries can further help estimate the complexity of lexical learning using n-gram statistics: assuming that learning one lexical item requires searching through all n-grams once (or even several times) in the given corpus and each search-through consumes a similar time, the complexity of learning \( k \) words from the corpus would be \( O(kn|C|) \). Although how many words could be learned from a corpus is unknown yet – it depends both on the regularities in the corpus that are captureable by the model formalism adopted for the learning task and on how capable the learning algorithm is of capturing these regularities – the above estimation shows, nevertheless, that the learning algorithm for unsupervised lexical learning, to be presented in later chapters, can run in a polynomial time. This is exactly the positive result we expect from the studies of the number of n-grams about the computational plausibility of lexical learning with n-gram statistics.

However, dealing with \( n|C| \) n-grams on a huge corpus, for example of millions or tens of millions of characters or words, is still an arduous job. A time complexity of \( O(n|C|) \) of a program looks really ideal, but a space complexity at this level is still dreadful, especially to those who have an interest in using n-grams of a large \( n \), say, up to 12 or 20, for their NLP applications. This is critical to our lexical learning task, since it is quite common to have a word of 12 characters or longer. It is wise for us to look for an efficient approach to overcoming this obstacle.

### 5.4 The Virtual Corpus

The Virtual Corpus (VC) approach [193, 196] to deriving and retrieving n-grams of any length from a large-scale corpus provides an efficient way to handle the issue that was raised at the end of last section. Our statistical approach to learning lexical items from large-scale corpora requires this technique, since the learning is obligated to examine all possible n-gram items in the corpus in order to discover good candidates for words.

The implementation of the VC system relies on position indexing of all suffix strings in a given corpus, each starting from some position in the corpus and getting through to the end of the corpus. This position indexing technique appears to have been popularised
by Nagao and Mori [260] in the domain of corpus-based NLP. The underlying data structure for this position indexing technique is known as suffix arrays, proposed by Manber and Myers [228]. It is also known as a \textit{PAT tree} in [139]. Church demonstrates a few simple programs for an implementation in [70].

We call it a virtual corpus with respect to the fact that it relies on a virtually existing sorted corpus for n-gram counting - see below for more details. Its functionality is not limited to merely counting n-grams. Interestingly, it has a remarkable potential for storing and retrieving n-grams and also provides other utilities concerning n-gram statistics.

In the following sections, we will first introduce the position indexing technique, and then report our implementation of an algorithm for the VC approach to counting n-grams of any length based on this technique. Notice that a pre-processing process is employed to convert corpus tokens into fixed length digit codes (if necessary) before constructing the VC, aiming at saving effort and memory space for string processing. The code length depends on the number and the type of the corpus tokens: an \texttt{int} type (in C++) is used for the codes only if the number cannot be accommodated by a \texttt{short} \texttt{int} type; if the tokens are ASCII characters, the pre-processing will be disabled.

A brief analysis of the computational complexity of the algorithm is also presented. The most important part of the job of constructing a VC is to sort the indices in the VC in terms of the order of the strings they represent. Experiments show that we implemented a highly efficient program, in that it can finish sorting the VC for the entire tagged PTB-II WSJ corpus of 1.3M tags in 1.5 minutes on a Sun SPARCstation 4 and in 0.5 minutes on a Sun SPARCstation 20, and sort the VC for the 6-million-character Brown corpus in 1.5 minutes on a Sun Ultra 2.

5.4.1 Position Indexing

A VC is realised by a set of pointers (or indices), called \textit{VC pointers}, each of which points to a corpus token (e.g., a word, a tag or a character) at some position in the corpus and stands, virtually, for the sequence of tokens from that token to the end of the corpus. The pointers are as many as the number of tokens in the corpus, i.e., the corpus size $|C|$. With these pointers we have a virtual corpus consisting of $|C|$ token sequences. After sorting these sequences by sorting their VC pointers, the count of an n-gram item consisting of any $n$ tokens in the corpus is simply the count of the number of adjacent pointers that take these $n$ tokens as prefix.
CHAPTER 5. N-GRAMS AND THE VIRTUAL CORPUS

This approach can be exemplified by the following three stages on a very simple artificial corpus.

Stage 1: Construct a virtual corpus with VC pointers to each token’s position, and let each pointer denote the token sequence from that position to the end of corpus:

Corpus:  b a b a a c b a a b .... [ ]
        ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
VC pointers: p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 .... [ ]
Virtual corpus:
  p1 -> b a b a a c b a a b .... [ ]
p2 -> a b a a c b a a b .... [ ]
p3 -> b a a c b a a b .... [ ]
p4 -> a a c b a a b .... [ ]
p5 -> a c b a a b .... [ ]
p6 -> c b a a b .... [ ]
p7 -> b a a b .... [ ]
p8 -> a a b .... [ ]
p9 -> a b .... [ ]
p10 -> b .... [ ]
:          ([ ]: end of corpus)

Stage 2: Sort the VC pointers and then count the length of each VC pointer’s common prefix with the next one in the sorted VC.

<table>
<thead>
<tr>
<th>CmmprefLgth</th>
<th>VCptr</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>p8    -&gt; a a b .... [ ]</td>
</tr>
<tr>
<td>1</td>
<td>p4    -&gt; a a c b a a b .... [ ]</td>
</tr>
<tr>
<td>2</td>
<td>p2    -&gt; a b a a c b a a b .... [ ]</td>
</tr>
<tr>
<td>1</td>
<td>p9    -&gt; a b .... [ ]</td>
</tr>
<tr>
<td>0</td>
<td>p5    -&gt; a c b a a b .... [ ]</td>
</tr>
<tr>
<td>3</td>
<td>p7    -&gt; b a a b .... [ ]</td>
</tr>
<tr>
<td>2</td>
<td>p3    -&gt; b a a c b a a b .... [ ]</td>
</tr>
<tr>
<td>1</td>
<td>p1    -&gt; b a b a a c b a a b .... [ ]</td>
</tr>
<tr>
<td>0</td>
<td>p10   -&gt; b .... [ ]</td>
</tr>
</tbody>
</table>
:          : p6    -> c b a a b .... [ ] |
:          :

After the sorting, all VC pointers with an identical prefix of any length, e.g., [b a] or [b a a] above, must be adjacent to one another.

Stage 3: Count the occurrences of n-gram items of any length n. This is simply to count the number of adjacent VC pointers with a common prefix length $\geq n$. It is very straightforward to derive the following n-grams and their counts from the above sorted VC.
CHAPTER 5.  N-GRAMS AND THE VIRTUAL CORPUS

<table>
<thead>
<tr>
<th>uni-gram</th>
<th>bi-gram</th>
<th>tri-gram</th>
<th>quad-gram</th>
</tr>
</thead>
<tbody>
<tr>
<td>[a]: 5</td>
<td>[a a]: 2</td>
<td>[a a b]: 1</td>
<td>....</td>
</tr>
<tr>
<td>[b]: 4</td>
<td>[a b]: 2</td>
<td>[a a c]: 1</td>
<td>[a a c b]: 1</td>
</tr>
<tr>
<td>[c]: 1</td>
<td>[a c]: 1</td>
<td>....</td>
<td>....</td>
</tr>
<tr>
<td>....</td>
<td>[b a]: 3</td>
<td>[b a a]: 2</td>
<td>[b a a b]: 1</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
<td>[b a b]: 1</td>
<td>[b a a c]: 1</td>
</tr>
</tbody>
</table>

After a VC is sorted, one may either go ahead and carry out the n-gram counting and output the results, or simply keep (or output) the VC as an n-gram resource such that an n-gram frequency is counted only when it is needed.

5.4.2 Sorting

The central task in the implementation of a virtual corpus based on the suffix array data structure is to sort the VC pointers, as illustrated in the preceding section. A number of technical details to enhance the efficiency of the sorting for the construction of a virtual corpus are particularly worth noting.

First, the corpus tokens are converted into digit codes before a VC is constructed. There are two major reasons for this. One is that it can make efficient use of the working space if it can determine the length of a code in terms of the vocabulary size (i.e., the number of distinct tokens in the corpus). Saving working space is a critical issue when dealing with large-scale corpora of millions or tens of millions of words. For example, for a PTB tagged corpus, since the tag number (which is 48) is small, we can use a char for a PTB tag, which is usually of 2-3 letters. In this way, to retrieve a tag in the corpus is to read a char, instead of a string. This treatment saves both space and time. It also leads to another favourite consequence: it can speed up the VC pointer comparison in the sorting. It is obvious that the comparison of two char codes is much faster than that of two corpus tokens as strings of several characters (e.g., English words). The token-code conversion is rather simple and fast: read through the corpus, once a new token shows up, assign a new code to it.

Second, a bucket-radixsort algorithm is employed for sorting VC pointers. Other sorting algorithms, e.g., the comb sort and disc mergesort that are reported in [260], were assumed to have the time complexity $O(N \log N)$ in comparisons of the VC pointers. As we see it, however, this complexity measure seems inappropriate in the context of sorting VC pointers, because comparing two VC pointers is to compare the two sequences of corpus tokens they represent. For example, to compare two VC pointers $p_i$ and $p_j$ as
follows, where \( ti_k = crps[pi+k] \) (assuming the corpus is in the array \( crps[] \)), the comparison procedure can be characterised as the pseudo-C++ code below, which is rather straightforward and highly similar to string comparison.

\[
\begin{align*}
pi \to & \; t_0 \; t_1 \; t_2 \; t_3 \; \ldots \; t_i \; \ldots \; [] \\
pj \to & \; t'_0 \; t'_1 \; t'_2 \; t'_3 \; \ldots \; t_j \; \ldots \; []
\end{align*}
\]

```c
int VPtr_cmp(int pi, int pj) // comparison of two VC pointers
{
    if (pi == pj) return 0;
    for (int rslt=0, k=0; ; k++)
        if (rslt = tok_cmp(crps[pi + k], crps[pj + k]))
            return rslt;
}

int tok_cmp(int ti, int tj) { return ti - tj; }
int tok_cmp(str ti, str tj) { return strtok(ti, tj); }
```

The first \( \text{tok cmp}() \) is for token code comparison and the second one for token string comparison - which one is used depends on the type of \( \text{crps[]} \). Also, notice that there is no need to break the \( \text{for} \) loop, because either \( \text{crps}[\text{pi+k}] \) or \( \text{crps}[\text{pj+k}] \) must hit the end-of-corpus \( [] \) (whose code is 0, uniquely) as \( k \) increases, and then \( \text{tok cmp}() \) returns a non-zero value and causes the result \( \text{rslt} \) to be returned.

The number of times of token reading and token comparison in VC pointer comparison as above actually depends on the length of the common prefix of the two pointers in question. This raises two problems for Nagao and Mori’s estimation of the complexity of their sorting algorithms [260]. First, it appears to be problematic to characterise the complexity of combsort and mergesort on a VC as \( O(N \log N) \) in VC pointer comparisons, because the time for comparing two VC pointers is not a constant. Consider an extreme corpus such as \([a \; a \ldots a \; b]\), of \( N \) \( a \)’s preceding \( b \) at the end. Sorting its VC will take \( O(N \log N) \) time in VC pointer comparisons, and any two pointers in the VC have a common prefix of length \( N/2 \) on average, i.e., comparing two VC pointers takes \( O(N) \) time in token comparisons. Consequently, the combsort and mergesort for VC sorting have a complexity of \( O(N^2 \log N) \) in token comparisons.

Second, the \( \text{VPtr cmp}() \) procedure above (or its equivalent) always starts comparing two VC pointers from the first token, then the second, the third, and so on, regardless of how many common prefix tokens have been found to be identical in the two VC pointers by previous comparisons. It is quite common that many comparisons of identical token pairs at the beginning parts of two VC pointers are unnecessarily repeated.
CHAPTER 5. N-GRAMS AND THE VIRTUAL CORPUS

However, this deficiency can be remedied by a bucket-radixsort. The pseudo-C++
code given below is the recursive version of the algorithm for sorting a VC with a
vocabulary size \( v \), where tokens are assumed to be in integer codes. Since the outputting
takes place during the recursion, there is no need to have a merge operation; otherwise
the running time could last significantly longer.

```cpp
void VC::brsort(int lvl) // bucket-radixsort a VC at level 'lvl'
{
    if (is_singleton()) output and return;
    VC *subVCs = new VC[v + 1]; // buckets
    for each ptr in *this VC into subVCs[crps(ptr + lvl)]; // division
    for (int i = 0; i < v; i++)
        if (!subVCs[i].empty()) subVCs[i].brsort(lvl + 1); // recursion
    delete [] subVCs;
}
```

Each corpus token in each VC pointer is read only once during the sorting, because
\( lvl \) increases incrementally and never repeats when the recursion goes deeper. The
recurrence for this algorithm is \( C_N = vC_{N/v} + N \), where \( N \) is the length of the input
corpus. Consequently, its complexity can be appropriately characterised as \( O(N \log_e N) \)
in token comparisons.

The bucket-radixsort we use in the VC program is an iterative version of the above
algorithm, where a stack for sub-VCs is used as working space in place of \( \text{subVCs[]} \),
so as to avoid frequent memory allocation in each recursion as the above, and, more
importantly, to avoid scanning through the \( v \) sub-VCs when most of them are empty.
This can speed up the program to a great extent, particularly when a corpus with a
large vocabulary-size is processed.

We have also tried an alternative sorting algorithm, namely the \( \text{qsort()} \) in standard
C library, as suggested in [70]. What we need to do is to provide an appropriate function
\int VCptr_cmp(const void* p1, const void* p2) \) for VC pointer comparison.

This approach is also very fast. It takes about 1.5 minutes, on a Sun SPARCstation
4, to construct a sorted Virtual Corpus for the tagged PTB-II WSJ corpus of 1.29M tags.
Roughly, the sorting takes only about an half minute; the rest of time is consumed by the
pre-sorting operations, like file loading and memory allocation for the initial unsorted
virtual corpus, and by the post-sorting operations, like calculating the common prefix
length between every pair of adjacent VC pointers in the sorted VC.

Experiments show that the \( \text{qsort()} \) works basically as well as the \( \text{brsort()} \). It runs
slower than \( \text{brsort()} \) on a small-vocabulary corpus (e.g., POS tag or character corpus)
but faster on a large-vocabulary corpus (e.g., word corpus). In the latter case, the above recursive version of the \texttt{bssort()} runs much slower than its iterative version, because when \( n \) is large, the \texttt{for} loop takes a long time to go through all \( v \) \texttt{subVCs}. In particular, when the recursion goes deeper, the time wasted on this loop is greater, because most of the \texttt{subVCs} are empty. In comparison with Nagao and Mori’s system, which takes an hour to carry out the sorting on a Japanese corpus of 3.7M bytes (i.e., 1.85M characters), the above two implementations (with \texttt{bssort()} and \texttt{qsort()} appears remarkably simpler and more efficient.

A number of experiments have been conducted on large-scale text corpora like the PTB-II WSJ corpus [232] and the Brown corpus [126], to derive n-gram statistics for n-grams of various lengths and token types using the VC system. The experiments were carried out on different models of the Sun station, in order to examine the program’s performance objectively. Table 5.1 is the experimental results on a Sun Sparc 20 and an Ultra 2. All running time data were obtained in this way: we ran the program on each corpus several times, and took the average of the two fastest running times as the final result.

One may ask why there are only 48 POS tags in PTB-II but we have 89 tags in the PTB-II Brown POS tag corpus and 85 tags in the PTB-II WSJ POS tag corpus. The reason is that there are a number of multiple tags used in the PTB corpus [232], e.g.,

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Size</th>
<th>Token Type</th>
<th>Vocabulary Size</th>
<th>\texttt{bssort} (sec.)</th>
<th>\texttt{qsort} (sec.)</th>
<th>Output VC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brown</td>
<td>1.17M</td>
<td>POS Tag</td>
<td>89</td>
<td>32.5</td>
<td>45.0</td>
<td>25.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Word</td>
<td>47705</td>
<td>50.0</td>
<td>34.0</td>
<td>24.0</td>
</tr>
<tr>
<td>WSJ</td>
<td>1.29M</td>
<td>POS Tag</td>
<td>85</td>
<td>33.0</td>
<td>52.0</td>
<td>27.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Word</td>
<td>45649</td>
<td>48.0</td>
<td>40.0</td>
<td>27.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Size</th>
<th>Token Type</th>
<th>Vocabulary Size</th>
<th>\texttt{bssort} (sec.)</th>
<th>\texttt{qsort} (sec.)</th>
<th>Output VC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brown</td>
<td>6.13M</td>
<td>POS Tag</td>
<td>58</td>
<td>11.0</td>
<td>15.0</td>
<td>6.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Word</td>
<td>47705</td>
<td>16.0</td>
<td>12.0</td>
<td>6.0</td>
</tr>
<tr>
<td>WSJ</td>
<td>1.29M</td>
<td>POS Tag</td>
<td>85</td>
<td>86.0</td>
<td>106.5</td>
<td>36.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Word</td>
<td>45649</td>
<td>13.0</td>
<td>17.0</td>
<td>6.5</td>
</tr>
<tr>
<td></td>
<td>5.73M</td>
<td>Char</td>
<td>57</td>
<td>86.5</td>
<td>96.0</td>
<td>32.5</td>
</tr>
</tbody>
</table>

Table 5.1: Execution time of \texttt{bssort()} and \texttt{qsort()} for VC construction for the PTB-II WSJ corpus and the Brown corpus.
JJN, JJVBG, to avoid forcing the annotators to make decisions they are unsure of. These multiple tags are treated as different tokens from other individual POS tags by the VC system.

From the experimental results, we can see that the system takes about 11-13 seconds on a Sun Ultra 2 and roughly half a minute on a Sun Sparc 20 to prepare a sorted VC for a corpus of 1.17 to 1.29 million POS tags (or words) for n-gram counting. On an Ultra 2, it takes less than 1.5 minutes to do the sorting for a corpus of 5.73 to 6.13 million characters. Furthermore, we also observe that, on an Ultra 2, the brsort() is about 30-36% faster than the qsort() on a medium-sized large-scale corpus with a small-sized vocabulary (e.g., the WSJ and Brown POS tag and character corpora), but about 25-33% slower on a corpus of similar size with a large-sized vocabulary (e.g., the WSJ and Brown word corpora). On a very large-scale corpus with a small-sized vocabulary, e.g., the WSJ and Brown character corpora, the brsort() is about 11-23% faster than the qsort(). Nevertheless, both of them appear to be adequately fast for the purpose of practical use.

Time Complexity

As analysed above, the complexity of the brsort() is $O(|C| \log |C|)$, when sorting the VC for a corpus $C$ with a vocabulary of size $v$.

The complexity of the qsort() is known to be $O(N \log N)$ - in our case, it is $O(|C| \log |C|)$ in VC pointer comparisons. On average, the number of token comparisons in comparing two VC pointers is the average length of the common prefix of the token sequences that the pointers stand for in the corpus. So, denoting the average common prefix length as $P$, the time complexity of sorting a virtual corpus using the qsort() can be characterised as $O(P|C| \log |C|)$.

Counting the common prefix length between a pair of adjacent VC pointers is similar to VC pointer comparison, and there are $|C|$ such pairs. So the time complexity for this task is only $O(P|C|)$ in token comparisons, on average.

Space Complexity

To construct a virtual corpus, we only need adequate space to accommodate the whole corpus of size $|C|$, and two integer arrays of the same size for the VC pointers and for the common prefix lengths of adjacent pointers. The stack size for the qsort() is, according to C library document, $O(1)$. So, the space complexity for the virtual corpus is $O(|C|)$,
which is linear to corpus size but has nothing to do with the maximal n-gram length $n$.

More precisely, if the corpus tokens and the VC pointers are also represented as integers, the space needed for the virtual corpus for a corpus $C$ is $3|C|$ integers (i.e., $12|C|$ bytes on a Sun station where an integer takes 4 bytes), ignoring the dynamic space allocation by the $qsort()$.

With respect to both the time and space complexity, the Virtual Corpus system demonstrates a very efficient approach to accessing n-grams of any length.

5.4.3 Counting

Although counting and retrieving n-grams in a sorted VC is rather simple, it still need to find the right way to do it, for the purpose of achieving a better efficiency - recall that the number of n-grams in a large-scale corpus is huge.

Counting all n-grams

It is trivial to output n-grams with a count of 1 in a sorted VC, with the aid of the array $cp1[]$ storing the common prefix lengths between adjacent VC pointers: for every VC pointer indexed by $i$ in the sorted VC, its first $\text{MAX}(cp1[i-1], cp1[i]) + 1$ tokens are an n-gram item that occurs only once in the corpus, and all longer n-grams taking this n-gram as prefix in the corpus are also of count 1. What we need to endeavour to deal with in n-gram counting is the n-grams with a frequency $\geq 2$ in the input.

There is a simple and efficient approach to calculating the frequency for all these n-grams in a batch mode in a sorted virtual corpus. The main idea is to accumulate the count for an n-gram item from all longer items that take it as prefix. This is similar to the calculation of the number of leaves in a tree by depth-first traversing a tree and summing up what every sub-tree has.

In Table 5.2, Part A gives the pseudo-C++ code for counting n-grams with a frequency $\geq 2$, with the aid of a set of accumulators, and outputting n-grams of specified minimal count and length, where $cp1[]$ and $VCptrs[]$ are data members and $\text{maxCPL}()$ and $\text{mySize}()$ are member functions in the VC class; Part B illustrates how the frequency accumulation works.

This program searches through each VC pointer and each n-gram item only once in the corpus. The accumulation is remarkably simple. The time complexity of the program is $O(|C|)$. It is also straightforward to customise it for outputting all n-grams up to a given length $k$, simply by skipping n-grams longer than $k$. 

\[ \text{\textit{CHAPTER 5. N-GRAMS AND THE VIRTUAL CORPUS}} \]
CHAPTER 5. N-GRAMS AND THE VIRTUAL CORPUS

Part A. Pseudo-C++ code for n-gram counting with frequency accumulation

```cpp
void VC::output_ngrams (int minC, int minL) const
    // output ngrams whose count >= minC (>= 2) & length >= minL
{
    int i, j, curcpl, tmpcnt;
    int *cnts = new int[maxCPL() + 1] & init: cnts[i] = 0; // cnts[0] not used
    for (i = 0, curcpl = cpl[i]; i < mySize(); i++) { // scan through VC
        if (cpl[i] == curcpl) cnts[cpl[i]]++;
    else { // cpl[i] < curcpl
        for (tmpcnt = 1, j = curcpl; j > cpl[i]; j--) {
            tmpcnt += cnts[j]; // count accumulation
            cnts[j] = 0; // reset
        if (tmpcnt >= minC && j >= minL)
            output: ngram = crps[VCptrs[i] .. VCptrs[i]+j-1] &
                count = tmpcnt
        }
        cnts[cpl[i]] += tmpcnt; // count accumulation
        cnts[curcpl] = 0; // reset
    }
    curcpl = cpl[i];
}
    delete [] cnts;
}
```

Part B. Illustration of frequency accumulation on an artificial virtual corpus


<table>
<thead>
<tr>
<th>CPL</th>
<th>VCptrs</th>
<th>Count</th>
<th>Output</th>
<th>Accumulate</th>
<th>Reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>p8 -&gt; a a b ..</td>
<td>c[2]++</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>p2 -&gt; a b a ..</td>
<td>c[2]++</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>0</td>
<td>p6 -&gt; a c b ..</td>
<td>c[1]++</td>
<td>c(a c) = c[1] = 3)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>3</td>
<td>p7 -&gt; b a a b ..</td>
<td>c[3]++</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>:</td>
<td>p6 -&gt; c b ..</td>
<td>..</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
</tbody>
</table>

Table 5.2: Pseudo-C++ code for n-gram counting on a sorted VC with frequency accumulation, and an illustration of the frequency accumulation on an artificial corpus
CHAPTER 5. N-GRAMS AND THE VIRTUAL CORPUS

Counting fixed-length n-grams

The pseudo-C++ code for a program to serve the purpose of counting n-grams of a fixed length is given as the following:

```cpp
void VC::ngrams(int lgth) const
{
    while (int cnt = 0, ix = 0; ix < mySize(); ix += cnt) {
        cnt = ngram_count(ix, lgth);
        output_ngram(crcs[VCptrs[ix]..VCptrs[ix]+lgth-1], cnt);
    }
}

int VC::ngram_count(int ix, int lgth) const
    // count the n-gram crcs[VCptrs[ix]..VCptrs[ix]+lgth-1]
{
    for (int i = 0; ; i++)
        if (cpl[ix+i] < lgth) return i + 1; // the last one
}

5.4.4 Retrieval

When a virtual corpus is used as storage for all n-grams in an input corpus, we need to have an efficient approach to retrieving individual n-gram items and their counts. There are two subtasks in the retrieval, namely, locate an n-gram item in the VC and calculate its count.

Locating an n-gram

Since the VC pointers are sorted, it is rather straightforward to locate an n-gram item using the binary search, of \(O(\log n)\) computation. The simplest way is using `bssearch()` in the standard C Library together with the `VCptr_cmp()` defined above.

Counting an n-gram

When a VC pointer taking the target n-gram, of length \(lgth\), as its prefix is located in the VC as `VCptr[ix]`, the counting of the n-gram is as simple as the following:

```cpp
int i, cnt = 1; // the one in VCptr[ix]
for (i = ix-1; cpl[i] >= lgth; i--) cnt++; // count backward
for (i = ix; cpl[i] >= lgth; i++) cnt++; // count forward
return ++cnt
```
CHAPTER 5. N-GRAMS AND THE VIRTUAL CORPUS

5.4.5 Other Utilities

In addition to counting and retrieving n-grams and their frequencies, a virtual corpus can also provide many other utilities for collecting statistical data from large-scale text corpora for corpus-based NLP, e.g., long distance bi-grams and mutual information between linguistic elements. Since they are not directly related to the theme of lexical learning, we are not going to go into the details here.

5.5 Summary

In this chapter, we have studied n-gram items of variable-length in large-scale natural language corpora, including the number of n-grams and the efficient way of counting n-grams.

We first introduced the multigram models relevant to lexical learning, which use variable-length n-grams to compute the probability for a long string, in contrast to fixed-order n-gram models. The multigram model is highly relevant to lexical learning in that both the multigram model and lexical learning involve a similar task to decompose an utterance into variable-length n-gram items. Lexical learning is more specific in matching the variable-length n-grams to linguistic words.

Then, we analysed the number of n-grams of various lengths in a corpus. Our studies lead to two important conclusions. First, the number of n-grams not in a given corpus is \(|V|^n - n|C|\), suggesting that using higher order n-gram models is unhelpful in language modelling and that enlarging the training data set is not a valid solution for the notorious sparse data problem – it only makes the problem even worse, in that \(|V|^n\) increases must fast than \(n|C|\) (because \(|V| > O(\sqrt{|C|})\) as the corpus enlarges [130]). Second, the number of n-grams of any length in a given corpus is linear to the corpus size – this result confirms the plausibility of doing lexical learning based on n-gram statistics. Lexical learning is different from language modelling in that it only needs to deal with n-grams that appear in the input corpus.

The focus of this chapter was the presentation of our implementation of an efficient Virtual Corpus system for very fast access to n-grams of any length. It takes about 1.5 minutes to construct a sorted VC for a corpus of 5-6 million characters for n-gram counting. In the VC system, we also provide an efficient approach for counting n-grams of various lengths with frequency accumulation. Our counting program goes through the sorted VC once and outputs all n-grams of all lengths. The implementation of the
VC system has laid a very good foundation for our unsupervised lexical learning based on n-gram statistics, which will be formulated in later chapters. In the next chapter, we will see the role that an n-gram count will play in the calculation of the description length gain – a goodness criterion for the selection of lexical candidates in unsupervised lexical learning.
Chapter 6

A Goodness Measure for Lexical Learning

6.1 Overview

In this chapter we formulate a goodness measure for unsupervised lexical learning from naturally-occurring corpora within the learning-via-compression paradigm following the MDL principle [294, 297, 344, 345]. The MDL principle is theoretically rooted in Solomonoff’s idea of inductive inference [322] and in Kolmogorov complexity theory [322, 198, 55, 219]. It is a computable approximation for the non-computable Kolmogorov complexity in tackling practical learning problems.

The specific task in this chapter is to develop a suitable criterion for unsupervised lexical learning from a natural language corpus. Its focus is to formulate an empirical information-theoretical measure, known as description length gain (DLG), for the goodness of inferring a linguistic structure such as an n-gram item of characters (or POS tags/words) as a lexical item (or phrase) from a given corpus. The formulation is based on classic information theory [312, 78], and the calculation of the description length gain for an n-gram item can be performed very fast based on its count and the alphabetical symbols’ counts in the given corpus. This goodness measure is suitable for unsupervised language learning including lexical learning within the MDL paradigm.

To examine the validity of this goodness measure, a best-first learning algorithm is designed to make use of this goodness measure to detect linguistic patterns in text corpora. Two small-scale experiments have been conducted on two unsupervised language learning tasks to learn, respectively, phrase patterns from a POS tag corpus and words from a character corpus. The experiments on phrase and lexical learning from sub-corpora of the Brown corpus [199] and the WSJ corpus in PTB-II give promising re-
CHAPTER 6. A GOODNESS MEASURE FOR LEXICAL LEARNING

sults. These positive results encourage us to move on to the design of more sophisticated algorithms for unsupervised lexical learning in Chapter 7.

The rest of this chapter is organised as follows. Section 6.2 reviews relevant previous research on language learning, focusing on the goodness criteria that various researchers have developed to infer different types of linguistic structures and/or language models. Section 6.3 re-visits the approach of language learning via compression, and develops the idea of doing the two-part code within the conventional MDL paradigm with one ideal universal coding scheme (that is, we actually encode the two parts as one), which appears to bring up some theoretical elegance in inductive inference. Section 6.4 formulates the calculation of description length in bits for a given corpus, adhering to Rissanen's insights about the objectiveness of code-word length [294, 299]; based on this calculation, we move on to develop the DLG measure, also called the compression effect, to evaluate the goodness of extracting a sub-string of characters (or POS tags/words) as a lexical item (or a phrase, respectively), in terms of the number of bits that the extraction of the sub-string as a pattern from the corpus can compress the entire corpus. Section 6.5 formulates a best-first learning algorithm guided by the DLG goodness measure, and Section 6.6 reports the preliminary results of two best-first learning experiments to test the validity of this goodness measure, one on phrase learning from a POS tag corpus and the other on lexical learning from a character corpus. The majority of the output from the experiments is comprised of linguistically sound phrases and words, indicating that the DLG criterion points to linguistic structures. Section 6.7 gives some concluding remarks on the goodness measure with regard to the experimental results.

Since many relevant topics have been discussed above, we have to, inevitably, re-highlight again some key points and references for the convenience of developing the discussion in this chapter. Some formulae are also repeated, but not numbered.

6.2 Criteria for Language Learning

Given no prior knowledge but only a corpus of a sequence of linguistic symbols of some type (e.g., characters, words or POS tags) as data, unsupervised language learning to infer linguistic structures needs to determine which segments of the sequence are target linguistic structures, such as words and phrases. The learning has to be guided by some reliable information-theoretical criterion for judging the goodness of a fragment of the input sequence as a linguistic structure. The learning needs to consider all fragments, i.e., all n-gram items, in the corpus. Although n-grams of arbitrary lengths in a large-
scale corpus are known to be huge in number, we have developed the Virtual Corpus (VC) system [193, 196], based on the suffix array data structure [228], as a fairly efficient approach to handling them, including counting, storing and retrieval. It provides an ideal operational basis for inducing lexical items and phrases, respectively, from character and POS tag corpus based on n-gram statistics. However, a more crucial issue in inducing linguistic structures from natural language corpora is a sound criterion to guide the learning towards linguistically sound structures.

As revealed by most previous practice, inferring a language model (e.g., a probabilistic regular grammar (PRG) like a HMM or a probabilistic context-free grammar (PCFG)) from natural language data involves, in general, two essential sub-tasks. One is to infer the model structure, such as a set of phrase structure rules (or productions) in a PRG or a PCFG and a set of states and transitions between states in a HMM. The other is to estimate a correspondent set of probabilistic parameters (e.g., production probabilities in a PCFG and transition probabilities in a HMM) that optimise the fit of the model to the data. The inference can be viewed as a search process for the best model in a predefined hypothesis space. If a set of permissible rules or rule formats (e.g., Chomsky normal form (CNF) for PCFG) are given, it is popular to apply, respectively, the forward-backward (or Baum-Welch) algorithm [10, 9] and its extension, the inside-outside algorithm [6, 208], to estimate the probabilistic parameters for regular grammars (e.g., HMMs) and PCFGs. Both the forward-backward and inside-outside algorithms are instantiations of the expectation-maximisation estimation [107]. There are also many other sophisticated algorithms, e.g., the genetic algorithm [156] and the simulated annealing algorithm [192], that can facilitate the searching.

However, no matter how sophisticated the search method used, the goodness criterion for guiding the searching process towards the target model remains a critical issue. It is this criterion that tells a search algorithm which model is better. To an unsupervised language learner (e.g., a lexical learner for learning words from a corpus of characters or phonemes, a phrase learner for learning phrases from a corpus of words or POS tags), every n-gram item in the corpus is possibly a candidate for the target linguistic structure. Thus, the first essential task of learning is to determine which n-grams are more likely to be the target structures and which are less. Towards this end, the learner needs to be equipped with a suitable criterion for differentiating between n-grams that are good candidates for the target structures and those ones that are not. Also, the criterion must indicate which among the good ones are better.
CHAPTER 6. A GOODNESS MEASURE FOR LEXICAL LEARNING

Many criteria have been explored in previous research. Cook et al. [75] explore a hill-climbing search for a grammar with a smaller weighted sum of grammar complexity and the discrepancy between grammar and corpus. Lari and Young [208] and Carroll and Charniak [51, 52] induce various types of probabilistic context-free grammar with the inside-outside algorithm [6] for probabilistic parameter re-estimation. Brill et al. [38] derive phrase structures from a tagged corpus with the measure generalised mutual information derived from mutual information. Brill and Marcus [39] attempt to induce binary branching phrases with distribution analysis using the information-theoretical measure divergence, derived from relative entropy. These researchers each have achieved significant progress, and also encountered some problems that they seem unable resolve given their approaches. For instance, Brill et al. have to redefine distitudes as a brute force to eliminate some unlikely phrases, e.g., [Noun Prep] and [Prep DT], that are favoured by their criteria. De Marcken [94] has an in-depth discussion on the kind of issues involved in pure distribution analysis and on the disadvantages of exploiting the inside-outside algorithm to select CFG rules in grammar induction.

Recent work in language modelling by Stolcke [328] and Chen [58, 59] is in the theoretical framework of Bayesian modelling. Basically, Stolcke’s work follows Cook et al.’s [75] paradigm of searching by hill-climbing, but guided by maximum likelihood. Chen follows Solomonoff’s basic idea of inductive learning [319, 321, 322], and uses the universal prior probability $p(G) = 2^{-|G|}$ for grammar induction. Their learning systems are reported to work well on small- to medium-sized artificial corpora, in terms of the theoretical measures like entropy, perplexity or likelihood. However, more concrete evaluations are expected to be based on the performance of learning from a large-scale naturally-occurring corpus like the Brown [126] or PTB corpus [232].

6.3 One-part versus Two-part Code

In this section, we will discuss the possibility of using one-part code instead of two-part code in MDL learning, and in the next section we will formulate the DLG measure for language learning on the basis of the one-part code.

The one-part code idea follows from the idea of learning via compression that we have discussed at length in the previous chapters. Recall Solomonoff’s notable discussion on the dual relationship between learning (i.e., detecting regularities in data) and compression in [322]. In the discussion, Solomonoff states that any regularity can be used to compress the data and what can be used to compress the data is a piece of
regularity. Compression is to derive a shorter (or more economic) representation for the
given data. Solomonoff also suggests a definition (or specification) for (unsupervised)
learning: learning is to detect regularities embedded in the data. We have followed this
line of thinking to develop a language learning theory in Chapter 3.

Compression, or a procedure equivalent to compression, has been widely exploited
for unsupervised language learning at various linguistic levels, e.g., as exemplified by the
work of Wolff [355, 356], Brent and co-workers [53, 54, 34, 35], Stolcke [328], Chen [58, 59]
and de Marcken [95, 97], among many others. Recently, the MDL principle is extensively
applied as a guidance to language learning, e.g., as demonstrated in [328, 95, 97, 59].
The central idea in MDL is that a model $M$ (e.g., a grammar $G$) for a set of data $D$
(e.g., a corpus $C$ as a sequence of characters or words) with the shortest description
length of both the model and the data (given the model), i.e., $|M| + |D_{given M}|$ in bits,
is the best model among all possible alternatives.

The difference among various learning-via-compression approaches lies in how the
compression is carried out. It is unnecessary to carry out a real compression procedure
for the learning, but a calculation of the probability of a model, $p(M)$, in terms of
its structure (e.g., grammar rules) and parameters, is inevitably needed. With $p(M)$
properly estimated, a learning procedure can focus on searching for the most likely
model, i.e., the model that maximises the posterior probability $p(M|D)$. By Bayes’ rule,
we have the Bayesian framework for unsupervised learning in (3.7), repeated below for
convenience, where $p(D)$ is a constant in relation to the variable $M$,

$$M_{\text{best}} = \arg \max_{M \in M} \frac{p(D|M) p(M)}{p(D)} = \arg \max_{M \in M} p(D|M) p(M)$$

How to estimate $p(M)$, the a priori probability of a language model, remains one
of the most critical issues in language learning. Stolcke and Chen both use the prior
$p(M) = c^{-l(M)}$ to calculate $p(M)$, where $l(M)$ is the description length of $M$ encoded
by a coding scheme that is carefully chosen or designed, and $c = 2$ if $l(M)$ is in bits.
Different ad hoc coding schemes are used by different researchers to compress $M$ for the
purpose of estimating $l(M)$ for the calculation of the prior $p(M)$.

Theoretically, searching for the most likely model for a given data set in a Bayesian
framework of modelling is equivalent to searching for a model with minimum description
length, because when the negative logarithm is applied, the above becomes the MDL
CHAPTER 6. A GOODNESS MEASURE FOR LEXICAL LEARNING

157

criterion expressed in (3.4) and (3.9), repeated as below.

\[ M_{\text{MDL}} = \arg\min_{M \in \mathcal{M}} - \log_2 p(D|M) - \log_2 p(M) \]

\[ = \arg\min_{M \in \mathcal{M}} |D_{\text{given } M}| + |M| \]  

(6.1)

This derivation straightforwardly follows from information theory [312, 78]. A Bayesian interpretation of MDL principle like this can also be found in [290, 219]. The MDL framework differs from the Bayesian framework only in that it can based on the code-word length in bytes in stead of probabilities. What else is also represented in (6.1) is the “two-part code” in the MDL literature, which indicates that the two parts on the right-hand side can be encoded independently of each other. That is, they can be encoded by different coding schemes.

However, there is another option that allows us to think of calculating the number of bits in the two parts in a unified way, if we assume, theoretically, there is an ideal universal coding scheme which can compress any sequence of data as well as any other coding scheme. Under this assumption, the model \( M \) and the data \( X \) (generated by \( M \)) in the MDL inference can be encoded as one-part code using this superior universal coding, instead of as two separate parts. That is,

\[ M_{\text{MDL}_1} = \arg\min_{M} |D_{\text{given } M} + M| \]

(6.2)

Notice, however, that this ideal universal coding scheme is not available in practice. For the purpose of language learning, a coding scheme such as Shannon-Fano coding [312], Huffman coding [158, 78] or arithmetic coding [293, 297] would suffice for the calculation of the description length for a model and for a data set given the model.

Consequently, a general criterion for selecting candidates for linguistic structures such as words or phrases from n-gram items within a given corpus \( C \) can be established in our study as follows: an n-gram item is a better candidate for the structure (either word or phrase) if incorporating it as a rule into our model \( M \) can lead to a shorter \(|M + C_{\text{given } M}|\). That is, only n-grams with a positive compression effect on the input corpus are considered as candidates for words or phrases. Accordingly, the unsupervised language learning becomes a search for a subset \( M \) of the set of all candidates such that \(|M + C_{\text{given } M}|\) is minimal.
6.4 Description Length Gain

In this section, we formulate the goodness measure description length gain (DLG) and its calculation following classic information theory [312, 78]. Then, we use it to evaluate how good a rule derived from linguistic data in large-scale corpora is. A rule represents a linguistic structure or pattern.

6.4.1 How good is a Model?

When one gets a model for a set of data, one asks how good the model is for the data. The answer depends largely on the purpose of constructing the model.

If we don’t care about the size of the model, or think of all possible models as having an equal prior probability, we have the ML criterion, as below.

\[ M_{ML} = \arg \max_{M \in \mathcal{M}} p(D|M) \]

The ML criterion prefers the model that gives the highest probability to the data, regardless of the size or complexity of the model. It is straightforward that the ML criterion is a special case of the Bayesian formulation of the learning problem. This criterion is popular in the speech processing community, where the greatest concern is the fit of the model to the data, instead of the model size or complexity.

A popular goodness measure for a model in speech processing is perplexity\(^1\), defined

\[ PP = 2^{LP} \]

\[ LP = \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} \log Q(w_i|w_1 \cdots w_{i-1}) \]

\(Q(\cdot|\cdot)\) is the probability under the language model in question, and the \(LP\) is theoretically defined as an asymptotic value. In practice, \(n\) is the size of some test data. Using the base 2 logarithm in (6.5), we have the \(LP\) for some test data \(D\) under a language model \(M\) as below (where \(n\) is replaced by \(|D|\)).

\[ LP(D|M) = \frac{1}{|D|} \log_2 \prod_{i=1}^{|D|} Q(w_i|w_1 \cdots w_{i-1}) = \frac{1}{|D|} \log_2 p(D|M) \]

\[ = \log_2 p(D|M)^{- \frac{1}{|D|}} \]

Applying this \(LP\) to (6.4), we have \(PP(D|M) = 2^{LP(D|M)} = p(D|M)^{- \frac{1}{|D|}}\).
as below.

\[ PP(D|M) = p(D|M)^{-\frac{1}{|M|}} \]

A lower perplexity indicates a better model for the data. Accordingly, the logarithm of the perplexity tells how many more bits, on average, are still needed to encode a symbol in the data with the aid of the given model.

\[ \log_2 PP_M(D) = -\frac{1}{|D|} \log_2 p(D|M) \]

Following the ML criterion, the fit of a model to the data is assessed by the probability it assigns to the data. That is, the higher the probability, the better the model. However, as it is well known, this criterion leads to the problem of data over-fitting or over-training. The over-fitting is an inherent problem of the ML criterion: when the training data is used to tune the parameters – either probabilistic or structural parameters, or both – in the model towards the data iteration by iteration, the training will not know when to stop, because any time the model is updated, the probability of the data given the model just keeps increasing. That is, the model is tuned more and more specifically to the data, in that it assigns an unreasonably high probability to some strings, in particular the frequent ones, in the data, but too low a probability to some others. An extreme case of over-training is that the entire set of data is taken over as the model and consequently such a model would assign a probability 1 to itself but zero to anything else.

Therefore, to avoid the over-training problem, there needs to be a balance between the complexity of the model and the probability it assigns to the data. The MDL principle offers a reasonable solution to this problem. It selects among all possible models the one that minimises the sum of the description length of the model and the description length of data given the model, as represented in (6.1) and (6.2) above. It is immune from the problem of the gain in increasing \( p(D|M) \) being not enough to pay off the cost of increasing the complexity of the model.

### 6.4.2 How Good is a Rule?

A model consists of a certain number of rules, selected from a large set (or space) of candidate rules. A different set of rules constitutes a different hypothesis in unsupervised learning. While selecting a set of rules as a model, we are concerned with how much goodness a candidate rule can contribute to the resultant model, because we aim to select the best set of rules.
CHAPTER 6. A GOODNESS MEASURE FOR LEXICAL LEARNING

In a probabilistic language model we usually do not think about how good an individual rule is. Also, we have not had any measure of how good a rule (e.g., a transition in a HMM) is. Probabilistic modelling does not answer this question on single rule basis. In a probabilistic model, a rule is associated with a probability, which indicates the chance for the rule to be used to generate the data, instead of how good it is. This probability is not an indication of how important the rule is in the model.

In the learning-via-compression approach, it is possible to evaluate the goodness of extracting a pattern (or a chunk) from the data as a rule – more specifically, as the right-hand-side of a rule. The goodness can be measured based on how much good it can do for the compression of the data by extracting the chunk as a rule, i.e., its compression effect. For example, if a chunk \( x_i x_{i+1} \cdots x_j \) is extracted from the corpus, we get a rule in the form \( r \rightarrow x_i x_{i+1} \cdots x_j \), and all occurrences of the chunk in the corpus can be replaced by the left-hand side \( r \). This replacement may lead to a positive compression effect, if the right-hand side is a good chunk.

This compression effect can be measured by the difference of the description lengths (or code-word lengths) of the data before and after the extraction of the rule. A bigger gain in the description length indicates that a stronger piece of regularity has been captured by the rule. This is why we prefer to call this goodness measure the description length gain (DLG). (Henceforth, we will allow both terms, namely, compression effect and description length gain, to be used interchangeably.)

Following this goodness measure, we can see that many possible rules extracted from a given corpus may bear a positive compression effect, others a negative effect. Correspondingly, we refer to the former as positive rules and the latter negative rules. The negative rules are thought of as capturing no regularity in the given data.

However, we have to bear in mind that the positive rules at the top of all possible rules, with regard to their compression effect, do not necessarily constitute an optimal model for the data, simply because such a model may not have the greatest sum of compression effect among all possible rule sets. Why? Because the compression effect of a model is not necessarily the sum of all its rules’ compression effects. There can be interference and competition among rules for compression effect. A possible effect of a competition is this: when a rule bearing a strong regularity wins its way into the model over some weak rules, the weak rules may be further weakened, in that their compression effects are accordingly reduced, because their counts in the input corpus may be reduced by the extraction of the strong rule.
CHAPTER 6. A GOODNESS MEASURE FOR LEXICAL LEARNING

Following the above goodness measure, the \textit{optimal model} for a set of data should be the set of rules that has the greatest total compression effect on the given data. Some rules in this set may not be at the top of the positive rules. Clearly, this further helps us to identify the central task of unsupervised language learning as to induce (or derive) the optimal set of rules from the given data. Or, in more general terms, it is to approach to a model, among all those reachable by the search method in use, that is closest to the true model, in terms of it compression effect. Recall that in many cases the true model may not be reachable by the search method in use.

6.4.3 Calculation of Description Length

Following the insights in Rissanen [297] and Quinlan and Rivest [290], we know that the application of the MDL principle is independent of the encoding scheme used. To resolve the critical issue of calculating the description length \( |M + D_{\text{given}}| \), as proposed in Section 6.3, what we need is a proper encoding method as an approximation for the ideal universal coding scheme that we assumed for the calculation, rather than a real compression program to carry out the compression of the data and its model.

Once a proper encoding method (e.g., Huffman coding [158]) is selected, the description length \( DL(X) \) on a given finite set of data \( X = x_1 x_2 \cdots x_N \), a corpus as a sequence of some linguistic symbols on a vocabulary \( V \), can be calculated as below in terms of the \textit{empirical entropy}\(^2\) \( \hat{H}(X) \), following classic information theory [312, 78].

\[
DL(X) = n \hat{H}(X) = -n \sum_{x \in V} \hat{p}(x) \log_2 \hat{p}(x) = - \sum_{x \in V} c(x) \log_2 \frac{c(x)}{|X|} \tag{6.5}
\]

Here \( c(x) \) is the token \( x \)'s count in \( X \) and the relative frequency \( \hat{p}(x) = c(x)/|X| \).

The empirical entropy \( \hat{H}(X) \) is the minimum expected average code-word length per symbol in \( X \), dependent on the probability (or frequency) distribution over all tokens in \( X \). Notice, however, that the above \( DL(X) \) is not identical to the Huffman code-word length for \( X \), because Huffman coding uses \( \lceil \log_2 \hat{p}(x) \rceil \) bits for each \( x \), whereas in (6.8) above we have \( \log_2 \hat{p}(x) \) for \( x \) for the purpose of calculation.

\(^2\)Notice its difference from Wright's notion of the same term [358], which is derived from all sentences' probability in a language \( L \), as

\[
H_e(L) = - \frac{\sum_{k=1}^{K} \log_2 P(S_k)}{\sum_{k=1}^{K} |S_k|}
\]
CHAPTER 6. A GOODNESS MEASURE FOR LEXICAL LEARNING

The relation between this description length and the perplexity under a uni-gram model is as straightforward as below in (6.9).

\[
\log_2 PP(X) = -\frac{1}{|X|} \log_2 \hat{p}(x_1 x_2 \cdots x_N) \\
= -\frac{1}{|X|} \log_2 [\hat{p}(x_1) \hat{p}(x_2) \cdots \hat{p}(x_N)] \\
= -\frac{1}{|X|} \sum_{i=1}^{N} \log_2 \hat{p}(x_i) \\
= -\frac{1}{|X|} \sum_{x \in V} c(x) \log_2 \hat{p}(x) \\
= \frac{1}{|X|} DL(X) \tag{6.6}
\]

As discussed above, perplexity is an indication of the quality of a language model – a model with a lower perplexity for the data is considered to be of a better fit to the data [163]. The description length of the data given a model and its average play similar roles in evaluating the quality of a language model.

6.4.4 Calculation of Description Length Gain

Following the description length calculation for a corpus as formulated above in (6.8), the description length gain from extracting a chunk of input symbols \( x_i x_{i+1} \cdots x_j \) (also denoted as \( x_{i,j} \) \( (i < j) \) from a given corpus \( X \) as a rule in the form \( r \rightarrow x_{i,j} \) can be defined as

\[
DLG(x_{i,j} \in X) = DL(X) - DL(X[r \rightarrow x_{i,j}] \oplus x_{i,j}) \tag{6.7}
\]

where \( X[r \rightarrow x_{i,j}] \) represents the resultant corpus from the operation of replacing all instances of \( x_{i,j} \) with the new symbol \( r \) throughout \( X \), and \( \oplus \) denotes a string concatenation operation with a delimiter\(^3\) inserted in between its two operands, i.e., the updated corpus \( X \) and the chunk \( x_{i,j} \) – the right-hand-side of the new rule.

What these few operations virtually do includes (1) extracting the string pattern \( x_{i,j} \) from the input corpus (and replacing all its occurrences with \( r \)), (2) putting it into the model, and then (3) putting the updated corpus and the model together for the calculation of the new DL \( |X_{\text{given } M + M}| \) as a one-part code. These virtual actions are merely for the purpose of calculating the DL. They need not be carried out during learning. A learning algorithm relying on the DLG can run very fast, because what it

\(^3\)For the sake of simplicity, sometimes the sign \( \oplus \) is also used to denote the delimiter inserted by the concatenation.
needs to do is simply the calculation, not the transformation of the old corpus into the
new one through these operations and the calculation.

The cost of the above transformation, virtual though it is, is the introduction of the
new symbol $r$ and a delimiter $\oplus$; the gain is that all instances of $x_{i,j}$, a sequence of
symbols, in the corpus are replaced by one symbol, namely, $r$. In general, $r \rightarrow x_{i,j}$ is a
good rule if the gain is greater than the cost.

In (6.10), we need to compute the DL for the new corpus $X' = X[r \rightarrow x_{i,j}] \oplus x_{i,j}$.
Following (6.8), it can be formulated straightforwardly as below in (6.11), using the new
count $c'(x)$ in the new corpus $X'$.

$$DL(X') = - \sum_{x \in V \cup \{r, \oplus\}} c'(x) \log_2 \frac{c'(x)}{X'}$$

(6.8)

where $c'(x)$ is the count of $x$ in $X'$. The focus of this computation is thus on how to
derive $c'(x)$ in the new corpus $X'$ without $X$. Notice that in order to have $X'$ we must
carry out the costly transformation by the extraction, replacement and concatenation.
This is what we do not want. Instead, we prefer to derive $c'(x)$ directly from the counts
$c(x)$ and $c(x_{i,j})$ in the original corpus $X$. Recall that we have had the efficient VC
system for deriving these counts for strings of any length in the corpus. Actually, this
derivation is quite straightforward, as given below in (6.12), where $c(\cdot)$ and $c(\cdot \in x_{i,j})$
denote an n-gram count in $X$ and in $x_{i,j}$, respectively.

$$c'(x) = \begin{cases} 
  c(x_{i,j}) & \text{if } x = r; \\
  c(\oplus) + 1 & \text{if } x = \oplus; \\
  c(x) & \text{if } x \notin x_{i,j}; \\
  c(x) - c(x_{i,j})c(x \in x_{i,j}) + c(x \in x_{i,j}) & \text{if } x \in x_{i,j}.
\end{cases}$$

(6.9)

The first three cases in (6.12) are trivially simple. In the fourth one, the general
case, the second item is the number of $x$'s reduced by the extraction of $x_{i,j}$ and the
third item is the number of $x$'s remaining in the only instance of $x_{i,j}$ in the model part.
The length of the updated corpus after the transformation for extracting $x_{i,j}$ is

$$|X'| = |X| - c(x_{i,j})|x_{i,j}| + |x_{i,j}| + 1$$

(6.10)

where the second item on the right-hand side is the length reduced by the extraction,
the third item is the length of the extracted pattern $x_{i,j}$, which is concatenated to the
updated corpus, and $1$ is for the delimiter introduced by the concatenation.

Accordingly, the average DLG of $x_{i,j}$, i.e., the compression effect of extracting each
individual instance of $x_{i,j}$, is
\[ avDLG(x_{i,j} \in X) = \frac{DLG(x_{i,j} \in X)}{e(x_{i,j})} \] (6.11)

The best-first learning algorithm in the next section will use the DLG in (6.10); and the optimal segmentation algorithm in the next chapter will use the average DLG in (6.14), because it needs to evaluate the DLGs of individual segments in a possible segmentation for an input utterance.

### 6.5 A Best-First Learning Algorithm

A best-first learning algorithm using the DLG as goodness measure for learning linguistic patterns from a language corpus is put forward as below, for the purpose of testing the effectiveness of the goodness measure and its calculation in unsupervised language learning. The algorithm reaches a lower DL when more patterns are acquired as rules in the language model.

1. Input: a corpus \( X_0 = x_1 x_2 \cdots x_N \) on a vocabulary \( V_0 \);
   Initialisation: set \( k = 0 \) and \( M_0 = \phi \) (i.e., empty).

2. Examine all n-gram items \( x_{i,j} \in X_k \) \((0 < i < j)\),
   (a) If no more \( x_{i,j} \in X_k \) such that \( DLG(x_{i,j} \in X_k) > 0 \),
   output the learning result in the model \( M_{k-1} \) and exit;
   (b) Otherwise, select
   \[
   s = \arg \max_{x_{i,j} \in X_k} DLG(x_{i,j} \in X_k)
   \]
   as the new pattern.

3. Update the corpus, vocabulary and language model accordingly:
   \[
   X_{k+1} = X_k[r^k \rightarrow s] \oplus s, \\
   V_{k+1} = V_k \cup \{r_k\}, \\
   M_{k+1} = M_k \cup \{s\}
   \]

4. Let \( k = k + 1 \), and goto 2.

It is worth noting that only the right-hand-side of a learned rule, instead of its index \( r^i \) \((0 \leq i < k)\), needs to be put in the language model and concatenated to the
updated corpus $X_i[r^i \rightarrow x_{i,j}]$ in Step 3. This eliminates all $r^i$'s, which are redundant indices, in the model part. Which pattern in the model belongs to which index $r^i$ can be straightforwardly recovered as follows: the pattern that immediately follows the $i$-th delimiter $@$ in the last updated corpus $X_{k-1}$ is the right-hand side for $r^i$.

One may see that this learning algorithm may not reach the shortest description length, since it is a best-first strategy that may, rather likely, stop at a local minimum. To get around this problem, a better optimisation process must be incorporated into the learning process – we postpone this task to Chapter 7. Our main purpose here is to test the effectiveness of the DLG criterion for unsupervised language learning, in particular lexical learning, before we go on to the design stage for a more advanced learning algorithm using this criterion. For the time being, a best-first learning algorithm suffices to serve this purpose.

For the purpose of testing, it is unsuitable to impose upon the learning algorithm any brute-force constraints like the artificial *distributes* \cite{38, 39} on the target linguistic structures, although that the learning has to observe some natural constraints, for example, a phrase does not cross a sentence boundary.

Since the left-hand side of an existing rule is allowed to go into the right-hand side of a new rule, the learning result from the best-first learning algorithm is a set of deterministic CFG (DCFG) rules. The DCFG rules guarantee the recovery of the original corpus by a reversed transformation.

After the learning, the original input corpus is turned into two parts. The first part is a compressed corpus where the regularities in the corpus have been squeezed out by the DCFG rules. The second part is the language model consisting of the DCFG rules representing the regularities. For the purpose of text compression, all tokens in these two parts need to be further encoded by a coding scheme such as Huffman coding \cite{158} or arithmetic coding \cite{293, 297}. Beyond this coding, the decompression process to recover the original corpus from the compressed corpus is relatively simple and can be carried out very fast with the aid of a stack, as follows.

1. If the stack is empty, read the next token from the compressed corpus; otherwise, pop a token from the stack.

2. If the current token is a terminal in $V_0$, output it; otherwise, it is a non-terminal (i.e., rule index),

(a) Retrieve its right-hand side from the model, and
(b) Push this right-hand side into the stack.

3. Go to Step 1, until both the stack and the entire compressed corpus are exhausted.

### 6.6 Preliminary Experiments

A number of preliminary experiments on unsupervised phrase and lexical learning have been conducted on parts of the PTB-II corpus, a revised and extended version of the PTB corpus [232]. This section presents the learning results. These experiments show that the learning approach with the DLG measure is effective. The learning algorithm has illustrated a very good capacity to capture the linguistic regularities in natural language data.

In the coming sub-sections, we will first introduce what the input and output look like with a simple illustration of the learning procedure, and then present the result of phrase learning from a POS tag sequence in a sub-corpus of the PTB-II Brown corpus and the result of lexical learning from a character sequence in a sub-corpus of the PTB-II WSJ corpus.

#### 6.6.1 Input and Representation

The input to the language learning is a text corpus as a list of naturally-occurring utterances. That is, a corpus $C$ with a vocabulary $V$,

$$ C = \{u_1, u_2, \cdots, u_N\}, $$

where each $u_i \in V^*$ for $i = 1, 2, \cdots, N$. The entire corpus can also be viewed as a single string, that is, $C \in V^N$, where $N = |C|$, which is the size of the corpus.

The vocabulary is a set of linguistic tokens. Each utterance, and even the entire corpus, is plainly a sequence of linguistic tokens. By the term *token*, we mean the atomic linguistic units at the particular linguistic levels of our interest, especially characters and words (or POS tags). Part A in Figure 6.1 below gives a small size example of a naturally-occurring corpus transcribed from a mother talking to a pre-linguistic baby, borrowed from [50]. In this version, the entire corpus is a sequence of characters (the lower- and uppercase letters are not differentiated), including delimiters like spaces and line breaks. A line break can be simply treated as a special case of space. Also, there is an end-of-corpus symbol $\square$ to mark the end of the corpus.
Part A. Original text corpus

oh, what a nice little smile!
yes, isn’t it that nice?
there. There’s a nice little smile.
what a nice wind as well!
yes, that’s better, isn’t it?
yes. yes. yes! that’s a nice noise.[]

Part B. Tokenised corpus

[oh][,][what][a][nice][little][smile][!]
[yes][,][isn’t][it][that][nice][?]
[there][.][there’s][a][nice][little][smile][.]
[what][a][nice][wind][as][well][!]
[yes][,][that’s][better][,][isn’t][it][?]
[yes][.][yes][.][yes][!][that’s][a][nice][noise][.][!]

Part C. A fragment of the tagged PTB WSJ corpus

[ Pierre/NNP Vinken/NNP ]
.,
[ 6i/CD years/NN ]
old/JJ ,/, will/MD join/VB
[ the/DT board/NN ]
as/IN
[ a/DT nonexecutive/JJ director/NN Nov./NNP 29/CD ]
./.

[ Mr./NNP Vinken/NNP ]
is/VBZ
[ chairman/NN ]
of/IN
[ Elsevier/NNP N.V./NNP ]
./,
[ the/DT Dutch/NNP publishing/VBG group/NN ]
./.

Figure 6.1: Examples of natural language text corpora
CHAPTER 6. A GOODNESS MEASURE FOR LEXICAL LEARNING

Part B in the figure is the tokenised version of the small corpus, where each token is a word. Each linguistic token is put into a pair of brackets. The entire corpus becomes a sequence of words. However, “what is a word?” remains a problematic issue in corpus linguistics. Is there one word or two words in each of the tokens isn’t and there’s? Also, there are a large number of multi-word lexical items in human lexicons that behave syntactically like individual words. How these compound lexical items are treated depends on the purpose of the corpus construction. In some corpus annotation practice [241], such a compound is assigned two part-of-speech tags to indicate the divisibility of such a two-part unit, for example, don’t_VD0+XX. Or it is divided into two parts and then each is assigned a tag such as <do_VD0 n’t_XX>. The extra angled brackets are used to mark the interdependency of the two parts. The simplest treatment is to allow all strings in the text corpus separated by spaces and known punctuation marks to be treated uniformly: they are individual orthographic units. In our studies, we will use these orthographic units as standard word forms to evaluate the lexical learning output.

Part C in the figure is a fragment of the tagged PTB-II corpus, where each square bracket is an identified noun phrase. Two corpora can be extracted from such a tagged corpus, one being a sequence of words and another being a sequence of POS tags. The sequence of POS tags extracted from this sub-corpus is as follows:

```
NNP NNP , CD NNS JJ , MD VB DT NN IN DT JJ NN NNP CD .
NNP NNP VBZ NN IN NNP NNP , DT NNP VBG NN .
```

It is expected that linguistically plausible structures like phrases can be learned from this sequence with the aid of some statistical or information-theoretical measures. But this has turned out to be an extremely difficult task. It is also questionable how much information is lost and how much distortion is introduced into the sentences like the ones above through the process of converting words into correspondent POS tags, which is known as tagging. Information loss is straightforwardly observed: the vocabulary size has been changed from the number of words in a language, which is at least tens of thousands, to the number of parts of speech in the language, which is around a hundred. In the so-called “gold standard” corpus PTB, there are 48 POS tags.
CHAPTER 6. A GOODNESS MEASURE FOR LEXICAL LEARNING

Learning Results

We do not specifically select the DCFG formalism as the representation for the language model in the best-first learning algorithm. As discussed before, the learning algorithm chooses it naturally by its own nature.

However, there are a few reasons to favour this formalism. The first one is that DCFG is the simplest grammar in the CFG class, in the sense that the computation is easier on the DCFG than on a non-deterministic CFG. The simplicity of a DCFG is even more straightforward when probabilistic CFGs are taken into account: DCFG is a special case of PCFG in which each rule has an equal probability, i.e., 1.

The second reason is that the results of phrase learning cannot always be represented by a regular grammar; they need to be expressed as CFG rules. CFGs have been the most popular formalism to represent linguistic structures in a hierarchy since the emergence of Chomskyian linguistics. Most mainstream linguistic theories, especially the generative grammar [63, 65, 162] and its relatives, such as lexical-functional grammar (LFG) [184, 36], and most work of sentence analysis in the NLP domain since the Cocke-Younger-Kasami (CYK) algorithm [186, 359] and Earley’s algorithm [112, 113] for natural language parsing employ the CFG formalism as the primary representation format for the hierarchical constituent structures in natural language sentences.

Another reason is that in the CFG class, DCFG is the one that enables both compression and decompression through string transformations by the substitution and concatenation operations. Contrary to this, a PCFG may enable the calculation of the conditional probability \( p(C|G) \) of a corpus \( C \) as a set of utterances given a grammar \( G \), and the logarithm of this probability equates, in theory, to the code-word length in bits for encoding \( C \) with the aid of \( G \), but the compression and decompression processes are problematic in practice.

Figure 6.2 shows an example to illustrate how words and phrases, represented as DCFG rules, are learned by a best-first learning algorithm from a small corpus of mother talk, borrowed from [50]. The symbol \( | \) is used as the delimiter between two rules, and the numbers in square brackets are rule indices, which do not need to show up in the model as left-hand sides.

The resulting DCFG rules from the learning are listed in Figure 6.3. The phrase “what a nice little smile” has a hierarchical structure, as in Figure 6.4, where each chunk carries the same index as in Figure 6.2. However, this artificial example of learning only serves to illustrate how the best-first learning takes place step by step; some rules
oh, what a nice little smile!
yes, isn’t it that nice?
there. there’s a nice little smile.
what a nice wind as well!
yes, that’s better, isn’t it?
yes. yes. yes! that’s a nice noise.[1]

“yes”[1], “little”[2] and “smile”[2] and “that”[4] are learned.

oh, what a nice [2] [3]!
[1], isn’t it [4] nice?
there. there’s a nice [2] [3].
what a nice wind as well!
[1], [4]’s better, isn’t it?
[1]. [1]. [1]! [4]’s a nice noise.]
yes|little|smile|that[]

“what”[2], “nice”[4], “there”[7], “it”[8] and “[2] [3]”[9] are learned.

[1], isn’t [8] [4] [6]?
[7]. [7]’s a [6] [9].
[1], [4]’s better, isn’t [8]?
yes|little|smile|that|what|nice|there|it|[2] [3][]

“isn’t”[8]”[10], “it’s”[11], “[6] [9]”[12] and “a [12]”[13] are learned.

oh, [5] [13]!
[1], [10] [4] [6]?
[7]. [7][11] [13].
[5] [13] wind as well!
[1], [4][11] better, [10]?
yes|little|smile|that|what|nice|there|it|[2] [3]
isn’t [8]’s|[6] [9]|a [12][]

Figure 6.2: A simple example process of best-first learning
CHAPTER 6. A GOODNESS MEASURE FOR LEXICAL LEARNING

[1] \rightarrow yes
[2] \rightarrow little
[3] \rightarrow smile
[4] \rightarrow that
[5] \rightarrow what
[6] \rightarrow nice
[7] \rightarrow there
[8] \rightarrow it
[9] \rightarrow [2] [3]
[10] \rightarrow isn’t [8]
[11] \rightarrow ’s
[12] \rightarrow [6] [9]
[13] \rightarrow a [12]

Figure 6.3: A simple example of DCFG language model

_______________________________
___[6] _______________________[13]
___[6] - _______________________[12]
what a nice little smile

Figure 6.4: An example of hierarchical structure

may not really bear a positive DLG. In the real learning, as formulated above, a rule is
carefully evaluated before it is really extracted. Thus, it is guaranteed that each learned
rule bears a positive DLG. The learner searches through the entire corpus for the best
positive rule every time.

6.6.2 Phrase Learning

The first 50 phrases learned from a PTB-II Brown sub-corpus of 101,723 POS tags
are listed below in Figure 6.5. The ones marked with y, x and ? are correct, false
and questionable ones, respectively. All sub-classes of noun are merged into NN. Other
PTB-II POS tags involved are listed in Figure 6.6. The number preceded by curDL in
each line is the combined DL of the current data and the model before the phrase is
acquired. If a rule index goes into another phrase, we attach its right-hand-side to it in
square brackets for readability. When all these phrases with a positive DLG are learned,
Figure 6.5: Phrase patterns output from phrase learning on the Brown corpus
CHAPTER 6. A GOODNESS MEASURE FOR LEXICAL LEARNING

| CC  | - Co-ordinating conjunction | CD  | - Cardinal number |
| DT  | - Determiner                | IN  | - Preposition or subordinating conjunction |
| JJ  | - Adjective                 | JJS | - Adjective, superlative              |
| MD  | - Modal                     | POS | - Possessive ending          |
| PRP | - Personal pronoun          | PRP$ | - possessive personal pronoun |
| RB  | - Adverb                    | RBH | - Adverb, comparative         |
| TO  | - 'to'                      | VB  | - Verb, base form             |
| VBD | - Verb, past tense          | VBG | - Verb, gerund or present participle |
| VBN | - Verb, past participle     | VBP | - Verb, non-3rd person singular present |
| VBZ | - Verb, 3rd person singular present | WDT | - Wh-determiner |
| WP  | - Wh-pronoun                | $   | - Dollar sign                  |

Figure 6.6: The PTB-II POS tags in addition to nouns involved in the phrase learning

the algorithm shortens the combined description length by about 13% beyond Huffman coding.

6.6.3 Lexical Learning

Below in Figure 6.7 is the experimental output of lexical learning from a 2K-character article in the PTB WSJ corpus. The spaces are replaced by underscore for visualisation. Uppercase letters are converted into correspondent lowercase ones, in order to prevent the cases of letter from hiding some lexical regularities in the data. It is observed that the best-first learning algorithm further shortens the combined description length by 15% beyond Huffman coding.

6.6.4 Discussion

From Figure 6.5, we can see that most phrases learned from a Brown sub-corpus of POS tags are good ones. Although [IN DT] is also learned as a phrase, this notorious problem in phrase structure learning that has been handled by brute-force distitutes in previous research appears to have been alleviated to some extent in our learning approach, in the sense that so many instances of IN and DT appear as a proper part in many other phrases. If we could have an optimal hierarchical chunking algorithm based on the DLG available, these bad phrases could be expected to be further filtered out – if there is another better choice of bracketing leading to a shorter description length on the sequence of words (or tags) in question. For example, [[IN DT] CD NN] may be changed to [IN [DT CD NN]], if the latter bears a greater DLG.

The lexical learning result presented in Figure 6.7 appears more impressive. Not only
curDL: 8252, R0: [_coalition government_]
curDL: 8093, R1: [_the khmer rouge_]
curDL: 8010, R2: [_non-communist_]
curDL: 7958, R3: [_the state department_]
curDL: 7807, R4: [_communist_]
curDL: 7670, R5: [_cambodia_]
curDL: 7623, R6: [_coalition_]
curDL: 7575, R7: [_is that the_]
curDL: 7549, R8: [_khmer rouge_]
curDL: 7528, R9: [_sandinistas_]
curDL: 7496, R10: [_vietnam_]
curDL: 7478, R11: [_guerrilla_]
curDL: 7462, R12: [_nicaragua_]
curDL: 7360, R13: [_the_]
curDL: 7314, R14: [could]
curDL: 7301, R15: [_communist_]
curDL: 7255, R16: [_that_]
curDL: 7243, R17: [_moderates_]
curDL: 7186, R18: [_the_]
curDL: 7167, R19: [_with_]
curDL: 7159, R20: [_partners_]
curDL: 7150, R21: [_america_]
curDL: 7143, R22: [_interim_]
curDL: 7124, R23: [_and_]
curDL: 7097, R24: [_of_]
curDL: 7085, R25: [_and_]
curDL: 7060, R26: [_in_]
curDL: 7052, R27: [_for_]
curDL: 7031, R28: [_ing_]
curDL: 7026, R29: [_groups_]
curDL: 7021, R30: [_regime_]
curDL: 7012, R31: [_part_]
curDL: 7006, R32: [_current_]
curDL: 7006, R33: [_include_]
curDL: 7002, R34: [_government_]
curDL: 6999, R35: [_were_]
curDL: 6994, R36: [_all_]
curDL: 6991, R37: [_in R5_]
curDL: 6987, R38: [_to_]
curDL: 6986, R39: [_gest_]
curDL: 6982, R40: [_re_]
curDL: 6976, R41: [_er_]
curDL: 6974, R42: [ou]

Figure 6.7: Lexical items output from the best-first learning on a WSJ article
CHAPTER 6. A GOODNESS MEASURE FOR LEXICAL LEARNING

words and multi-word linguistic units (e.g., place and organisation names) are properly learned, but a number of phonological regularities in the orthographic form, e.g., [tion], [ing], [re], [er] and [ou], are also detected.

We have not applied the best-first learning algorithm to a corpus as large as the entire Brown corpus yet, because the replacement operation after learning a rule is too expensive. But a lexical learning experiment on the PTB ATIS corpus shows that it can achieve an entropy rate of 1.92 bits/character, over the data and the grammar (i.e., the lexical items learned). It is hard at the moment to jump to the conclusion that our algorithm compares favourably with other researchers’ work, because the corpora we have used so far are relatively small and simple. The best result reported in this direction in the past is de Marcken’s MDL lexical learner [95] with a forward-backward algorithm to tune the mode structure and estimate the probabilities. The entropy rates it achieves in lexical learning from the Brown corpus are reported to be 1.92 bits/character on training data and 2.04 bits/characters on test data, omitting the grammar (i.e., the lexical items learned) and some other overheads. Ristad and Thomas’ [300] non-monotonic context modelling following the MDL principle achieves a massage entropy of 1.97 bits/characters on the Brown corpus, leaving the model aside.

It is unlikely that our best-first learning algorithm can beat other sophisticated algorithms with a better optimisation procedure. Our main purpose here is to test the validity of the DLG goodness measure for unsupervised language learning. Our learning results reported above appear to have confirmed the validity.

6.7 Summary and Concluding Remarks

In this chapter, we have developed an information-theoretical measure, namely, the description length gain, as a goodness criterion for unsupervised language learning via compression to infer linguistic patterns (e.g., words and phrases) from natural language corpora within the MDL learning paradigm. The preliminary experiments on phrase and lexical learning with a best-first learning algorithm guided by the DLG measure have shown promising results and verified the validity of this goodness measure.

The calculation of the DLG for an n-gram item as a rule candidate in a given corpus is very simple, in that it can be carried out merely based on the symbol count changes caused by the virtual string substitution for the extraction of an n-gram as a rule. There is no need to carry out the expensive corpus transformation for the calculation. This enables a learning algorithm to run very efficiently.
CHAPTER 6. A GOODNESS MEASURE FOR LEXICAL LEARNING

A learning algorithm using this goodness measure has two other distinct features. First, it counts the description length in bits, instead of calculating probabilities and then estimating the compression effect in terms of the probability of the data given the model. Counting bits and calculating probability are theoretically equivalent, but operationally different in practice. This leads to another distinct feature of our approach: the input corpus and the model as learning result are encoded together as one by the same coding scheme, instead of as two separate parts encoded by two different schemes. Theoretically, we assume an ideal universal compression scheme for description length calculation for a corpus and a language model. In practice, an encoding scheme like Huffman coding suffices for the purposes of language learning and that of deriving DCFG rules for text compression. In this sense, our learning approach is theoretically and operationally elegant, in addition to being effective.

Our preliminary experiments have shown encouraging results. However, in order to have a clearer idea on how well the DLG criterion can work in the context of lexical learning, we still need to integrate it into some more sophisticated learning algorithms, e.g., an optimal segmentation algorithm using DLG to induce lexical units from utterances, and have some more thorough evaluations of the learning results. The design and evaluation of the optimal segmentation algorithm for unsupervised lexical learning will be the two main tasks in the next chapter.
Chapter 7

Learning Algorithms and Evaluation

7.1 Overview

In the previous two chapters we developed the necessary basic techniques to support the unsupervised language learning that we intend to study, with a focus on unsupervised lexical learning. In Chapter 5 we reported our implementation of a practical and very efficient approach for deriving n-grams of any length from a large-scale corpus with the aid of a position indexing technique that is based on the suffix array data structure. In Chapter 6 we formulated the DLG goodness measure for unsupervised language learning and a simple and efficient approach for its calculation for a given n-gram based on the n-gram and symbol counts involved. In this chapter we will present the unsupervised lexical learning algorithms with the DLG measure as a guidance to induce lexical items from an input corpus, and evaluate the learning performance with a number of empirical evaluation measures. Since all supporting facilities for the learning have been well developed, the remaining work for the formulation and implementation of the learning algorithms is relatively simple.

In Section 7.2 we will first briefly introduce the format of the input data for the lexical learner – a more thorough discussion about the origin of the input data is postponed to Section 7.4.1 – and then present the representation formalism for the intermediate and final results of the learning with illustrations, to complement and tailor what we have introduced in Section 6.6.1 for lexical learning. In contrast with the deterministic context-free grammar used in the best-first learning algorithm in Section 6.6.1, the representation formalism here specifically for the lexical learning is a special deterministic regular grammar – the simplest one, where the left-hand side of a rule will never go
into the right-hand side of any other rule. The right-hand sides, all of which consist of
terminal symbols only, are the lexical candidates for the target lexicon. It is a simple
but adequate representation formalism for a lexicon in our study.

In Section 7.3, we will present the details of the lexical learning algorithms. The first
one is to induce lexical candidates from the input, the second one to refine the candidates
into finer-grained lexical items, and the last one to segment the original input into words
using the final lexicon yielded from the lexical refinement process. The kernel of these
algorithms is an optimal segmentation algorithm with respect to the DLG goodness
measure. It is a Viterbi algorithm that segments an utterance into a sequence of chunks
(i.e., lexical candidates) such that the sum of the DLG over all these chunks is maximal.

The focus of this section is on the formulation of the optimal chunking algorithm
that aims at extracting as many well-formed words as possible from given data, guided
by the DLG goodness measure. Computational complexities of these algorithms are also
analysed.

In Section 7.4 we will present the empirical evaluation of the learning performance
of our lexical learning algorithms on large-scale corpora of naturally-occurring child-
directed speech. We will discuss the original data and the pre-processing for extracting
the input data from the original corpora for the learner. Then we will report the learning
results and learning performance in terms of a number of empirical evaluation measures,
including segmentation precision and recall, word boundary precision and recall, lexicon
precision and recall, and correct segmentation ratio – we will define these measures
accordingly in the section.

In Section 7.5 we will discuss the learning results and analyse a number of problems
that we encountered in our unsupervised lexical learning approach, for example, some
natural “constraints” on the learning such as the vowel constraint – any word must
contain at least a vowel, i.e., a syllable. Finally, we will give the conclusions of this
chapter in Section 7.6.

7.2 Representation for Lexical Learning

Not all of the details of the representation for unsupervised language learning with the
best-first approach using the DLG measure discussed in Section 6.6.1 are applicable to
the lexical learning here, in particular the deterministic context-free grammar formalism
is not used. Instead, we use a simpler grammar formalism, a deterministic regular
grammar, to represent the learning results. But the format of the input data remains
unchanged. In this section, we will illustrate the representation formats with real input data and learning output from our learning experiments. We hope what we present here can make the learning algorithms in the next section easier to read.

7.2.1 Input Data

The input data is a corpus of plain text transcribed from child-directed speech. In Section 7.4 we will give more rationales for using text corpora to study lexical learning mechanisms. Below in Figure 7.1 is an exemplar fragment of the original transcription of a mother’s speech to language-learning children in the Bernstein corpus from the CHILDES collection.

However, this is not yet the input for our lexical learner. We have to do some pre-processing to filter out the noise in the data. First of all, it is necessary to filter out the non-speech content in the data, including commentary notes, punctuation marks\footnote{The only exception is apostrophe [’]. It is not removed from the input corpus, because it is used to represent a reduced vowel in bound morphemes in English orthographic texts, e.g., [-n’s] and [-‘re]. In order to enable lexical learner with a vowel constraint to detect individual bound morphemes, we have to inform the learner that an apostrophe is an equivalent vowel character. See later sections for more discussion.}, etc. Next, we convert all capital letters into lowercase ones, except the initial letters in proper names, to prevent the learner from making an unnecessary distinction between a word and its capitalised version. Also, we add a special end-of-utterance symbol “#” to each utterance, to tell the learner where an utterance ends. This symbol is not part of the data on which the learner will perform the learning. It is necessary because we intend to feed the data to the learner utterance by utterance. After these steps, we have a text corpus consisting of a list of utterances, each of which is a sequence of characters ended by “#”. Below in Figure 7.2 is the counterpart of the data in Figure 7.1 after the pre-processing.

This text corpus contains spaces as word delimiters between words. In order to test the learning ability of an unsupervised learning algorithm, we have to make such word delimiters unnoticeable to the learner. A good way to hide the spaces from the learner is to delete them from the input data, as in Figure 7.3. The spaces are artificial delimiters in written texts, in the sense that there is nothing in continuous speech corresponding to such spaces. Removing them from the input corpus appears to be the right thing to do for the purpose of testing a learner’s ability of learning the orthographic transcription of speech. A spaceless text such as the one in Figure 7.3 is also known as an unsegmented text; its counterpart with spaces is accordingly called a segmented text. We will use an
*MOT: She’s really into books right now.  *MOT: Get it.
*MOT: You want to see the book?  *MOT: Get it.
*MOT: Oh, look there’s a boy with his hat.  *MOT: Get it.
*MOT: And a doggie.  *MOT: Is that for the doggie?
*MOT: Oh, you want to look at this?  *MOT: Can you feed it to the doggie?
*MOT: W’ look at this?  *MOT: Feed it +...
*MOT: Have a drink.  *MOT: Oh # put it in OK.
*MOT: OK now.  *MOT: OK.
*MOT: Oh what’s this?  *MOT: What are you gonna do?
*MOT: What’s that?  *MOT: I’ll let her <play with these> [///] play with this for a while.

*MOT: What is it, huh?  *MOT: What?
*MOT: Look can you take it out?  *MOT: What’s this?  *MOT: What’s this?
*MOT: Take it out.  *MOT: A smile, it has a smile. [=! laugh]
*MOT: You want it in?  *MOT: Give him a kiss, OK, come on.
*MOT: xxx.  *MOT: Yeah! [=! laugh]
*MOT: Put that on.  *MOT: What is it, a ball?
*MOT: That.  *MOT: Look # What’s this?
*MOT: What’s this?

*MOT: OK.  *MOT: What’s that? [=! high]
*MOT: Open it up. [=! falsetto]  *MOT: What’s that?
*MOT: I think it will come out.  *MOT: Yeah, Yeah.
*MOT: Let’s see.  *MOT: Wh # this a butterfly?
*MOT: Yeah.  *MOT: Oh.
*MOT: Pull it out.  *MOT: What’s that?
*MOT: Ooh # what’s it?  *MOT: What’s that?
*MOT: Look # look # [=! squeal] what’s that?  *MOT: For milk?
she’s really into books right now #
you want to see the book #
oh look there’s a boy with his hat #
and a doggie #
oh you want to look at this #
w’ look at this #
have a drink #
ok now #
oh what’s this #
what’s that #

what is it huh #
oh #
look can you take it out #
take it out #
you want it in #
xxx #
put that on #
can you xxx it #
that #
yes #

ok #
oh #
open it up #
take the doggie out #
i think it will come out #
let’s see #
yeah #
pull it out #
ooh # what’s it #
look # look # what’s that #

get it #
get it #
get it #
is that for the doggie #
can you feed it to the doggie #
feed it #
oh # put it in ok #
ok #
what are you gonna do #
i’ll let her play with this for a while #

what #
what #
what’s this #
a smile it has a smile #
give him a kiss ok come on #
yeah #
what is it a ball #

ok look at the book #
look # what’s this #
what’s this #

what’s that #
what’s that #

that #
oh what a nice doggie #
yeah yeah #

wh # this a butterfly #
oh #
what’s that #
what’s that #

for milk #

Figure 7.2: A fragment of the input corpus after pre-processing to filter out non-speech content
CHAPTER 7. LEARNING ALGORITHMS AND EVALUATION

unsegmented corpus of child-directed speech to test our lexical learner’s performance. It is not easy for a human speaker to read off the words in an unsegmented text as the one in Figure 7.3. Without a certain learning capacity an unsupervised learner would not be able to infer words from a corpus of this type with an acceptable degree of success.

7.2.2 Learning Result

The unsupervised lexical learning is aimed at acquiring lexical forms from speech data where word boundaries are not marked by any means. The expected outcome from the learning algorithms is a lexicon consisting of a list of individual word forms. This representation for both the intermediate and final results of the learning in our research is in fact a deterministic regular grammar. In this grammar, the right-hand sides of rules are lexical candidates that are represented plainly as strings of the atomic symbols in the input corpus. The left-hand sides, which merely function as indices for the lexical candidates, are skipped in the representation, because the positions of the candidates in the lexicon play the same role as the skipped indices. The choice of skipping the left-hand sides in the lexicon also reflects the principle of simplicity – the underlying philosophy of our learning approach. Recall that our learning approach seeks for a lexicon as the simplest representation for the input data.

Here we will illustrate the representation formalism with fragments of the real output from our lexical learning experiments. The learning algorithms yielding such output will be formulated in the next section. The output from the first step of the lexical learning – an optimal segmentation of input utterances into lexical candidates in terms of the DLG measure – consists of two parts: One is the results of the optimal segmentation on the input utterances, such as in Figure 7.4; and the other is the correspondent lexical candidates resulting from the segmentation, such as the ones in Table 7.1, where each candidate is has its count (or frequency), length, coverage and average DLG attached. The spacing between characters is for readability, and the symbol ”+” is originally in the corpus. Many of the infrequent candidates in this lexicon, not shown here, are non-words; and most of the frequent ones, as shown in the figure, are real words or clumps of real words.

Therefore it is necessary to have a lexical refinement process in the lexical learning to turn the word clumps into individual real words. Table 7.2 gives a number of decompositions of word-clump lexical candidates into words (and other shorter clumps) during the lexical refinement process, where the right-most column is the DLG of each
she's really into books right now# get it#
you want to see the book# get it#
oh look there's a boy with this hat# get it#
and a doggie# isthat for the doggie#
oh you want to look at this# can you feed it to the doggie#
w'look at this# feed it#
have a drink# oh put it in ok#
ok now# ok#
oh what's this# what are you gonna do#
what's that# i'll let her play with this for a while#

what is it huh# what#
look can you take it out# what's this#
take it out# asmile as a smile#
you want it in# give him a kiss so come on#
xxx# yeah#
put that on# what is it a ball#
canyou xxx it# oh look at the book#
that# look# what's this#
what's this#

ok# what's that#
oh# what's that#
open it up# that#
take the doggie out# oh what an nice doggie#
it think it will come out# yeah yeah#
let's see# wh# this is a butterfly#
yeah#
pull it out# oh#
ooh# what's it#
look# look# what's that#
what's that#
what's that#
formilk#

---

Figure 7.3: A fragment of the input corpus with spaces deleted
[she’s][really][int][o][book][sright][now] [getit] [youwantto][seethe][book] [getit] [ohlook][there’s][abo][ywith][his][hat] [getit] [and][a][doggie] [isthat][for][thedoggie] [oh][youwantto][lookatthis] [canyou][feeditto][thedoggie] [w’][lookatthis] [feed][it] [havea][drink] [oh][putitin][ok] [ok] [o][know] [whatareyou][gonnado] [oh][what’sthisis] [i’ll][le][ther][playwiththe][se] [what’sthat] [playwithth][isfor][a][whi][le] [whatisit][huh] [what] [oh] [what’sthis] [look][canyou][takeitout] [what’sthis] [takeitout] [as][mil][e][i][tha][sas][mil][e] [youwant][itin] [gives][him][a][kiss][ok][come][on] [xxx] [yeah] [put][thaton] [whatisit][a][ball] [canyou][xxx][it] [oh][lookatthebook] [tthat] [look][what’sthisis] [what’sthis] [yes] [what’sthis] [ok] [ok] [what’sthat] [what’sthat] [open][itup] [that] [take][thedoggie][out] [ohwhat][a][nice][doggie] [think][it][will][comeout] [yeah][yeah] [let’see] [wthis][a][butterfly] [yeah] [pullit][out] [what’sthat] [oh] [what’sthat][it] [what’sthat] [look][look][what’sthat] [for][milk]

Figure 7.4: A fragment of the output from the optimal segmentation of the Bernstein corpus
<table>
<thead>
<tr>
<th>Token</th>
<th>Count</th>
<th>Coverage</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>57</td>
<td>6</td>
<td>342</td>
<td>pretty</td>
</tr>
<tr>
<td>44</td>
<td>8</td>
<td>352</td>
<td>those are</td>
</tr>
<tr>
<td>30</td>
<td>12</td>
<td>360</td>
<td>closed the door</td>
</tr>
<tr>
<td>52</td>
<td>7</td>
<td>364</td>
<td>this one</td>
</tr>
<tr>
<td>73</td>
<td>5</td>
<td>365</td>
<td>mommy</td>
</tr>
<tr>
<td>73</td>
<td>5</td>
<td>365</td>
<td>hello</td>
</tr>
<tr>
<td>37</td>
<td>10</td>
<td>370</td>
<td>where's the</td>
</tr>
<tr>
<td>53</td>
<td>7</td>
<td>371</td>
<td>bye bye</td>
</tr>
<tr>
<td>62</td>
<td>6</td>
<td>372</td>
<td>you can</td>
</tr>
<tr>
<td>188</td>
<td>2</td>
<td>376</td>
<td>it</td>
</tr>
<tr>
<td>67</td>
<td>6</td>
<td>402</td>
<td>doggie</td>
</tr>
<tr>
<td>41</td>
<td>10</td>
<td>410</td>
<td>what is that</td>
</tr>
<tr>
<td>207</td>
<td>2</td>
<td>414</td>
<td>-1.8734</td>
</tr>
<tr>
<td>52</td>
<td>8</td>
<td>416</td>
<td>what is it</td>
</tr>
<tr>
<td>85</td>
<td>5</td>
<td>425</td>
<td>it's a</td>
</tr>
<tr>
<td>43</td>
<td>10</td>
<td>430</td>
<td>look at this</td>
</tr>
<tr>
<td>108</td>
<td>4</td>
<td>432</td>
<td>book</td>
</tr>
<tr>
<td>63</td>
<td>7</td>
<td>441</td>
<td>there's</td>
</tr>
<tr>
<td>37</td>
<td>12</td>
<td>444</td>
<td>what are those</td>
</tr>
<tr>
<td>76</td>
<td>6</td>
<td>456</td>
<td>what's</td>
</tr>
<tr>
<td>152</td>
<td>3</td>
<td>456</td>
<td>-1.0154</td>
</tr>
<tr>
<td>477</td>
<td>1</td>
<td>477</td>
<td>-6.0098</td>
</tr>
<tr>
<td>161</td>
<td>3</td>
<td>483</td>
<td>see</td>
</tr>
<tr>
<td>81</td>
<td>6</td>
<td>486</td>
<td>is that</td>
</tr>
<tr>
<td>122</td>
<td>4</td>
<td>488</td>
<td>this</td>
</tr>
<tr>
<td>60</td>
<td>9</td>
<td>540</td>
<td>the dragon</td>
</tr>
<tr>
<td>98</td>
<td>6</td>
<td>588</td>
<td>can you</td>
</tr>
<tr>
<td>197</td>
<td>3</td>
<td>591</td>
<td>you</td>
</tr>
<tr>
<td>74</td>
<td>8</td>
<td>592</td>
<td>all right</td>
</tr>
<tr>
<td>66</td>
<td>9</td>
<td>594</td>
<td>the doggie</td>
</tr>
<tr>
<td>149</td>
<td>4</td>
<td>596</td>
<td>here</td>
</tr>
<tr>
<td>161</td>
<td>4</td>
<td>644</td>
<td>okay</td>
</tr>
<tr>
<td>65</td>
<td>10</td>
<td>650</td>
<td>peek a boo</td>
</tr>
<tr>
<td>222</td>
<td>3</td>
<td>666</td>
<td>and</td>
</tr>
<tr>
<td>62</td>
<td>11</td>
<td>682</td>
<td>that's right</td>
</tr>
<tr>
<td>122</td>
<td>6</td>
<td>732</td>
<td>that's</td>
</tr>
<tr>
<td>110</td>
<td>7</td>
<td>770</td>
<td>15.4102</td>
</tr>
<tr>
<td>157</td>
<td>5</td>
<td>785</td>
<td>there</td>
</tr>
<tr>
<td>198</td>
<td>4</td>
<td>792</td>
<td>that</td>
</tr>
<tr>
<td>203</td>
<td>4</td>
<td>812</td>
<td>look</td>
</tr>
<tr>
<td>452</td>
<td>2</td>
<td>904</td>
<td>oh</td>
</tr>
<tr>
<td>230</td>
<td>4</td>
<td>920</td>
<td>what</td>
</tr>
<tr>
<td>329</td>
<td>4</td>
<td>1316</td>
<td>yeah</td>
</tr>
<tr>
<td>139</td>
<td>10</td>
<td>1390</td>
<td>what's this</td>
</tr>
<tr>
<td>247</td>
<td>10</td>
<td>2470</td>
<td>what's that</td>
</tr>
</tbody>
</table>

Table 7.1: Examples of lexical candidates output from the optimal segmentation, in ascending order of coverage ( = count x length), each with its count, length, coverage and average DLG attached on the left.
Chapter 7. Learning Algorithms and Evaluation

Round 1 (The first 12 in 503 decompositions):

[what’s that]  ==>  [what’s]  [that]    15.8447
[what’s this]  ==>  [what’s]  [this]   16.3159
[that’s right]  ==>  [that]  [’s]  [right]  8.8198
[all right]   ==>  [all]  [right]   8.4251
[can you]     ==>  [can]  [you]    3.3597
[the dragon]  ==>  [the]  [dragon]  11.5103
[what are those]  ==>  [what]  [are those]  14.4629
[there’s]     ==>  [there]  [’s]      4.8175
[look at this] ==>  [look]  [at]  [this]  5.2867
[what is]     ==>  [what]  [is]       7.5195
[what is that] ==>  [what]  [is that]   14.9419
[you can]     ==>  [you]  [can]      3.3597

Round 2 (The first 12 in 112 decompositions):

[i is that]     ==>  [is]  [that]      0.6786
[that’s]        ==>  [that]  [’s]      3.3377
[that’s a]      ==>  [that]  [’s a]    1.1489
[the doggie]    ==>  [the]  [doggie]   15.4579
[where’s]       ==>  [where]  [’s]     6.5925
[in there]      ==>  [in]  [there]     5.4346
[the dog]       ==>  [the]  [dog]      0.2781
[this is]       ==>  [this]  [is]      1.6062
[you like]      ==>  [you]  [like]     10.3563
[the bunny]     ==>  [the]  [bunny]    11.6377
[another one]   ==>  [a]  [another]  [one]  2.7744
[the book]      ==>  [the]  [book]     8.0009

Round 3 (The first 10 in 35 decompositions):

[what’s]        ==>  [what]  [’s]      6.4455
[you want]      ==>  [you]  [want]     8.6686
[are you]       ==>  [are]  [you]      3.6702
[doyo you]      ==>  [do]  [you]       1.4109
[dowith]        ==>  [do]  [with]      0.8707
[didn’t]        ==>  [did]  [n’t]      0.6807
[goodbye]       ==>  [good]  [bye]     4.4128
[with the]      ==>  [with]  [the]     5.6365
[cow jumping over the moon]  ==>  [cow]  [jump]  [ing] [over]  [them]  [o]  [on]  1.2160
[knocked them]  ==>  [knock]  [ed]  [them]  0.1631
[put it]        ==>  [put]  [it]       0.4277
[what]          ==>  [what]  [’]       0.1275

Round 4 (2 decompositions):

[does she]      ==>  [does]  [she]     0.0765
[any you]       ==>  [an]  [you]       0.2196

Table 7.2: Examples of decomposition of word-clump lexical candidates into finer-grained lexical items during the lexical refinement process
<table>
<thead>
<tr>
<th>Count</th>
<th>Length</th>
<th>Coverage</th>
<th>Average DLG</th>
</tr>
</thead>
<tbody>
<tr>
<td>56 7 392</td>
<td>24.7372</td>
<td>11.2601</td>
<td>2.3465</td>
</tr>
<tr>
<td>80 5 400</td>
<td>8.9886</td>
<td>2.0056</td>
<td>2.0364</td>
</tr>
<tr>
<td>68 6 408</td>
<td>11.2601</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>207 2 414</td>
<td>-1.8734</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>71 6 426</td>
<td>10.1195</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>144 3 432</td>
<td>2.0056</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>145 3 435</td>
<td>0.4780</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>109 4 436</td>
<td>4.6727</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>91 5 455</td>
<td>8.5864</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>76 6 456</td>
<td>17.3798</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>92 5 460</td>
<td>9.5032</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>155 3 465</td>
<td>2.5998</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>162 3 486</td>
<td>-0.1738</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>122 4 488</td>
<td>5.4852</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>168 3 504</td>
<td>2.8772</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>64 8 512</td>
<td>20.7181</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>106 5 530</td>
<td>11.9946</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>268 2 536</td>
<td>-2.9071</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>134 4 536</td>
<td>4.7049</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>181 3 543</td>
<td>3.5252</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>109 5 545</td>
<td>8.1970</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>93 6 558</td>
<td>14.9853</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>144 4 576</td>
<td>6.0441</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>116 5 580</td>
<td>13.5829</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>147 4 588</td>
<td>7.6852</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>629 1 629</td>
<td>-5.5904</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>160 4 640</td>
<td>6.7353</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>161 4 644</td>
<td>6.1727</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>65 10 650</td>
<td>41.8614</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>372 2 744</td>
<td>-0.2737</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>194 4 776</td>
<td>7.1221</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>275 3 825</td>
<td>2.7419</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>179 5 895</td>
<td>10.0591</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>224 4 896</td>
<td>7.8500</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>329 3 987</td>
<td>0.7991</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>560 2 1120</td>
<td>-2.3705</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>292 4 1168</td>
<td>4.8850</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>342 4 1368</td>
<td>5.3697</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>393 4 1572</td>
<td>5.5280</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>455 4 1820</td>
<td>7.8423</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>421 5 2105</td>
<td>8.9229</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>776 3 2328</td>
<td>1.5757</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>985 3 2955</td>
<td>5.0111</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>784 4 3136</td>
<td>7.1216</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
<tr>
<td>856 4 3424</td>
<td>5.5652</td>
<td>2.0056</td>
<td>2.3465</td>
</tr>
</tbody>
</table>

Table 7.3: Examples of finer-grained lexical items output from the lexical refinement process, in ascending order of coverage, each with its count, length, coverage and average DLG attached on the left.
[she's][really][in][to][books][right][now] [you][want][to][see][the][book] [oh][look][there]['s][a][boy][with][his][hat] [and][a][doggie] [oh][you][want][to][look][at][this] [w'][look][at][this] [have][a][drink] [ok][now] [oh][what][‘s][this] [what][‘s][that] [what][is][it][huh] [oh] [look][can][you][take][it][out] [take][it][out] [you][want][it][in] [xxx] [put][that][on] [can][you][xxx][it] [that] [yes] [ok] [oh] [open][it][up] [take][the][doggie][out] [think][it][will][come][out] [let’s][see] [yeah] [pull][it][out] [of][oh][what][‘s][it] [look][look][what][‘s][that] [getit] [getit] [getit] [is][that][for][the][doggie] [can][you][feed][it][to][the][doggie] [feed][it] [oh][put][it][in][ok] [ok] [what][are][you][gonna][do] [i’ll][le][ther][play][with][the][se] [play][with][this][for][aw][hi][le] [what] [what] [what][‘s][this] [asmile][hasasmile] [give][him][a][kiss][ok][come][on] [yeah] [what][is][it][a][ball] [oh][look][at][the][book] [look][what][‘s][this] [what][‘s][this] [what][‘s][that] [what][‘s][that] [that] [that] [oh][what][is][nice][doggie] [yeah][yeah] [whhisa][butterfly] [oh] [what][‘s][that] [what][‘s][that] [for][milk]
<table>
<thead>
<tr>
<th>Count</th>
<th>Word</th>
<th>Count</th>
<th>Word</th>
<th>Count</th>
<th>Word</th>
<th>Count</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>516</td>
<td>10.4986</td>
<td>other</td>
<td>5</td>
<td>520</td>
<td>9.7007</td>
<td>who's</td>
</tr>
<tr>
<td>5</td>
<td>550</td>
<td>12.0695</td>
<td>daddy</td>
<td>5</td>
<td>550</td>
<td>8.2140</td>
<td>hello</td>
</tr>
<tr>
<td>4</td>
<td>556</td>
<td>5.7148</td>
<td>it's</td>
<td>5</td>
<td>565</td>
<td>12.7322</td>
<td>bunny</td>
</tr>
<tr>
<td>4</td>
<td>580</td>
<td>4.8412</td>
<td>with</td>
<td>1</td>
<td>580</td>
<td>-5.7134</td>
<td>a</td>
</tr>
<tr>
<td>4</td>
<td>584</td>
<td>6.0685</td>
<td>good</td>
<td>5</td>
<td>590</td>
<td>13.6191</td>
<td>mommy</td>
</tr>
<tr>
<td>3</td>
<td>600</td>
<td>14.7449</td>
<td>dragon</td>
<td>2</td>
<td>608</td>
<td>-1.7303</td>
<td>do</td>
</tr>
<tr>
<td>2</td>
<td>616</td>
<td>-3.4989</td>
<td>to</td>
<td>2</td>
<td>630</td>
<td>-2.6536</td>
<td>it</td>
</tr>
<tr>
<td>4</td>
<td>632</td>
<td>7.8306</td>
<td>have</td>
<td>4</td>
<td>644</td>
<td>6.1727</td>
<td>okay</td>
</tr>
<tr>
<td>5</td>
<td>675</td>
<td>6.3450</td>
<td>those</td>
<td>3</td>
<td>708</td>
<td>0.4372</td>
<td>one</td>
</tr>
<tr>
<td>5</td>
<td>710</td>
<td>10.3046</td>
<td>don't</td>
<td>3</td>
<td>729</td>
<td>4.0315</td>
<td>put</td>
</tr>
<tr>
<td>2</td>
<td>770</td>
<td>-1.5703</td>
<td>is</td>
<td>3</td>
<td>837</td>
<td>1.4219</td>
<td>are</td>
</tr>
<tr>
<td>4</td>
<td>884</td>
<td>7.8259</td>
<td>book</td>
<td>3</td>
<td>900</td>
<td>2.8851</td>
<td>and</td>
</tr>
<tr>
<td>5</td>
<td>925</td>
<td>10.1183</td>
<td>wanna</td>
<td>4</td>
<td>944</td>
<td>7.4020</td>
<td>your</td>
</tr>
<tr>
<td>4</td>
<td>976</td>
<td>7.5159</td>
<td>like</td>
<td>4</td>
<td>976</td>
<td>5.9158</td>
<td>want</td>
</tr>
<tr>
<td>5</td>
<td>1030</td>
<td>9.4411</td>
<td>where</td>
<td>5</td>
<td>1050</td>
<td>16.2244</td>
<td>doggie</td>
</tr>
<tr>
<td>3</td>
<td>1077</td>
<td>4.0248</td>
<td>can</td>
<td>3</td>
<td>1092</td>
<td>0.9634</td>
<td>see</td>
</tr>
<tr>
<td>5</td>
<td>1105</td>
<td>10.8338</td>
<td>right</td>
<td>5</td>
<td>1148</td>
<td>4.8562</td>
<td>here</td>
</tr>
<tr>
<td>4</td>
<td>1230</td>
<td>-2.2253</td>
<td>oh</td>
<td>4</td>
<td>1384</td>
<td>5.3887</td>
<td>yeah</td>
</tr>
<tr>
<td>4</td>
<td>1884</td>
<td>7.9047</td>
<td>look</td>
<td>5</td>
<td>2280</td>
<td>9.0590</td>
<td>there</td>
</tr>
<tr>
<td>5</td>
<td>2304</td>
<td>6.1544</td>
<td>this</td>
<td>2</td>
<td>2550</td>
<td>1.8949</td>
<td>’s</td>
</tr>
<tr>
<td>5</td>
<td>3225</td>
<td>2.1182</td>
<td>the</td>
<td>3</td>
<td>3768</td>
<td>5.4795</td>
<td>you</td>
</tr>
<tr>
<td>4</td>
<td>4704</td>
<td>6.1291</td>
<td>that</td>
<td>4</td>
<td>5096</td>
<td>8.0147</td>
<td>what</td>
</tr>
</tbody>
</table>

Table 7.4: Examples of lexical items in the final lexicon after word segmentation using the refined lexicon, in ascending order of coverage, each with its count, length, coverage and average DLG attached on the left.
CHAPTER 7. LEARNING ALGORITHMS AND EVALUATION

decomposition. Table 7.3 is the finer-grained lexicon that is output from the lexical refinement.

The final results of the lexical learning, i.e., the output from the optimal segmentation of the input utterances using the refined lexicon (obtained by the first two steps of the learning, namely, the optimal segmentation and lexical refinement) consist of two parts: One is the segmentation result, as illustrated in Figure 7.5, and the other is the correspondent lexicon, as illustrated in Table 7.4.

7.3 Learning Algorithms

In addition to the best first learning algorithm presented in Section 6.5, which was designed for the purpose of testing the validity of the description length gain (DLG) as the goodness measure for unsupervised language learning, several more sophisticated learning algorithms will be developed in this section with the aid of the DLG measure, for the purpose of inferring words as linguistic patterns from naturally-occurring natural language data.

All these algorithms rely largely on the DLG calculation instead of on any particular string operations, although the string operations are inevitably involved in the learning process. By this virtue these learning algorithms get around a serious operational problem that the best-first learning algorithm encounters, that is, when a word is determined in the best-first learning, all occurrences of the word need to be extracted from the corpus in question and then replaced by a new symbol that represents the word. This string substitution is a very costly operation, because it has to go through the entire corpus of millions of characters in order to carry out the string substitution. What really makes the operation so time-consuming is not merely the substitution, but the work to adjust the rest of the corpus after the substitution.

Beyond overcoming the efficiency disadvantage of the best-first learning, these learning algorithms hold another more important characteristic: they output the optimal segmentation of the input utterances into lexical candidates with respect to the DLG goodness measure. We will formulate a Viterbi algorithm for the optimal segmentation with the aid of the supporting tools presented in the previous chapters. All other algorithms for lexical learning in our study are variants of the Viterbi algorithm. In this section, we will first present the Viterbi algorithm, and then its variants to work with different word space (i.e., the set of possible lexical candidates) under different situations, and finally we combine these variant algorithms into a program for unsupervised
CHAPTER 7. LEARNING ALGORITHMS AND EVALUATION

lexical learning. The evaluation of the lexical learning will be presented in the next section with experimental learning results from a child-directed speech corpus from the CHILDES collection.

7.3.1 Optimal Segmentation with DLG

In this section we present the Viterbi algorithm for the optimal segmentation of an input utterance into chunks, in terms of their DLG computed with respect to the relevant chunks’ (i.e., n-grams’) counts in an input corpus. These chunks will be taken as lexical candidates for further refinement in the unsupervised lexical learning.

Given an utterance as a string (or sequence) of some linguistic tokens (either characters, phonemes or syllables) \( U = t_1 t_2 \cdots t_n \) in a given corpus \( C \), the Viterbi algorithm searches for an optimal segmentation \( OS(U) \) over the utterance \( U \) such that the sum of the compression effect over all these segments is maximal. Formally put, it is\(^2\)

\[
OS(U) = \arg\max_{s_1 \cdots s_k \text{ s.t. } U = s_1 \oplus \cdots \oplus s_k} \sum_{i=1}^{k} \text{avDLG}(s_i)
\]  

(7.1)

for \( 0 < k < |U| \), where the \( \oplus \) symbol indicates string concatenation, and \( \text{avDLG}(s_i) \) is the average description length gain by each occurrence of the string \( s_i \) - the compression effect of extracting one individual instance of \( s_i \) from the input corpus. The method of calculation of the avDLG for any n-gram in a given corpus was given in the previous chapter.

The Viterbi Algorithm for Optimal Segmentation with DLG

The Viterbi algorithm to search for the best segmentation for an utterance \( U = t_1 t_2 \cdots t_n \) that fulfils (7.1) is presented in Figure 7.6, with an illustration. The algorithm needs to use a list of intermediate variables \( BS[i] \) (for \( i = 0, 1, 2, \cdots, n \)), each of which stores the best segmentation over \( [t_1 t_2 \cdots t_i] \). A segmentation is a list (i.e., an ordered set) of adjacent segments. The sign \( \uplus \) represents the union operation of two ordered sets. The DLG over a set of segments (or strings), e.g., \( DLG(BS[j]) \), is the sum of the DLG of the individual segments in the set, as defined in (7.2) below.

\[
DLG(BS[j]) \uplus \sum_{s \in BS[j]} DLG(s)
\]  

(7.2)

\(^2\)Under the circumstance that the input corpus in question is clear, we denote \( DLG(s_i \in C) \) and \( \text{avDLG}(s_i \in C) \) as \( DLG(s_i) \) and \( \text{avDLG}(s_i) \), respectively, for the sake of simplicity.
(A) An illustration for the Viterbi segmentation

$OpSeg(U = t_1 t_2 \cdots t_n)$

For $i = 1, 2, \cdots, n$ do

  Initialise $BS_s[i] = \emptyset$;

  For $j = i - 1, \cdots, 2, 1, 0$ do

    If $c([t_{j+1} \cdots t_i]) < 2$ break;

    If $DLG(BS_s[j] \cup \{[t_{j+1} \cdots t_i]\}) > DLG(BS_s[i])$

      then $BS_s[i] = BS_s[j] \cup \{[t_j \cdots t_i]\}$.

The final result: $OS(U) = BS_s[n]$.

(B) The Viterbi segmentation algorithm

---

Figure 7.6: The Viterbi algorithm for optimal segmentation with an illustration
Also, the algorithm should avoid taking a single token as a rule, because a single
token rule bears a negative DLG. That is, when \( j = i - 1 \), \( BS_s[j] \cup \{t_{i+1} \cdots t_i\} \) becomes \( BS_s[i-1] \cup \{t_i\} \) instead of \( BS_s[i-1] \cup \{\{t_i\}\} \). Notice the difference between
the denotations \( BS_s[i-1] \cup \{t_i\} \) and \( BS_s[i-1] \cup \{\{t_i\}\} \): the latter means that the string
\( t_i \) is extracted from the corpus as the right-hand side of a new rule, which results in
a negative DLG; whereas the former only treats \( t_i \) as a single token segment without
making it a rule, which has no compression effect.

It is worth noting the importance of the breaking condition \( c([t_j \cdots t_i]) < 2 \) in the
inner loop. It is an empirical condition. Its main purpose is to speed up the algorithm by
avoiding fruitless iterations on strings of count 1. There are two considerations for setting
this breaking condition. First, all strings with a count \( c = 1 \) have a negative DLG, thus
none of them could contribute a positive compression effect to any segmentation
of the utterance. According to our observations from many experiments, the learning
algorithm without this breaking condition outputs the same result as the one with it,
but the running time is a number of times longer. Another reason for strings of count 1
to be skipped in the learning from a large-scale corpus is that they are all long strings
that can be broken into shorter ones with a positive DLG. It is better for the algorithm
to manage on its own to break it into positive chunks rather than make it a chunk with
a negative DLG.

The second consideration is that the breaking condition can speed up the algorithm

\(^3\text{Because extracting a string s of count 1 as a rule does not change any token t’s count (for all t ∈ V) in the new corpus C}^\varepsilon \rightarrow s\ \square s, \text{ except the new non-terminal I (a rule index) and the delimiter \&, whose counts become 1 (i.e., c(I) = c(s) = 1 and c(\&) = 1) after the extraction. Thus,}

\[
DLG(s) = DL(C) - DL(C^V \rightarrow s) \square s
\]

\[
= - \sum_{t \in V} c(t) \log_2 \frac{c(t)}{|C|} - \sum_{I \in \{I, \&\}} c(I) \log_2 \frac{c(I)}{|C| + 2}
\]

\[= - \sum_{t \in V} c(t) \log_2 \frac{|C| + 2}{|C|} + \log_2 \frac{1}{|C| + 2} + \log_2 \frac{1}{|C| + 2}
\]

\[= - \log_2 \frac{|C| + 2}{|C|} \sum_{t \in V} c(t) - 2 \log_2 |C| + 2
\]

\[= - |C| \log_2 \frac{|C| + 2}{|C|} - 2 \log_2 |C| + 2
\]

\[= - (|C| + 2) \log_2 (|C| + 2) + |C| \log_2 |C|
\]

\[< 0
\]

This result holds for all strings of any length of count 1.
significantly. Time complexity analysis on the algorithm confirms this. Without this empirical condition, the computational complexity of the algorithm working on an utterance of length \( n \) is \( O(n^2) \). With it, the complexity is bound by \( O(mn) \), where \( m \) is the maximal common prefix length in the correspondent virtual corpus for the input corpus. Notice that only a string with a length smaller than \( m \) can have a count \( c > 1 \) (See Chapter 5 for more details). Accordingly, the average complexity of the Viterbi algorithm is approximately \( O(an) \), where \( a \) is the average common prefix length.

After introducing this conditional break, a question one may ask is, would it cause any trouble for the algorithm? At first glance, it is possible, because there can be a \([ t_k \cdots t_i ]\) whose count \( c([ t_k \cdots t_i ]) < 2 \) (therefore, \( DLG([ t_k \cdots t_i ]) < 0 \)) such that

\[
DLG(BS_s[k]) + DLG([t_{k+1} \cdots t_i]) > DLG(BS_s[j]) + DLG([t_{j+1} \cdots t_i])
\] (7.4)

for all integer \( j \in [k+1, i-1] \). Notice that the right-hand side of the inequality subsumes the best segmentation \( BS_s[i] = BS_s[j'] \cup \{ t_{j'+1} \cdots t_i \} \) for some \( j' \in [k+1, i-1] \), and that in the case \( j = i - 1 \), the right-hand side becomes

\[
DLG(BS_s[j]) + DLG(t_i) = DLG(BS_s[j])
\]

instead of

\[
DLG(BS_s[j]) + DLG([t_i]) < DLG(BS_s[j])
\]

following the algorithm.

Once we take a closer look at the problem, we find that it is rather straightforward to prove that if the inequality in (7.4) holds, \( BS_s[k] \) becomes a part of the optimal segmentation \( BS_s[i] \) that is capturable by the above Viterbi segmentation algorithm. With respect to (7.4), we have

\[
DLG(BS_s[k]) > DLG(BS_s[j]) + DLG([t_{j+1} \cdots t_i])
\] (7.5)

for all \( j \in [k+1, i-1] \). So, an even better segmentation than \( BS_s[i] = BS_s[j'] \cup [t_{j'} \cdots t_i] \) can be simply constructed as follows:

\[
BS_s[i] = BS_s[k] \cup t_{k+1} \cdots \cup t_i
\]

And this better segmentation is capturable by the Viterbi segmentation algorithm with the conditional break.
CHAPTER 7. LEARNING ALGORITHMS AND EVALUATION

![Diagram](image)

Figure 7.7: The model of lexical learning behind our lexical learning algorithms

### 7.3.2 Unsupervised Lexical Learning

The algorithm for unsupervised lexical learning using the above Viterbi optimal segmentation is formulated in the following three steps:

1. Induction of lexical candidates by optimal segmentation on input utterances,
2. Lexical refinement by optimal segmentation on individual lexical candidates, and
3. Word segmentation using the lexicon resulting from the above two steps.

Each of these steps is an application of the optimal segmentation algorithm under a specific condition. The first two steps constitute the process of the lexical learning, and the last step uses the learned lexicon to perform word segmentation to get the final results for evaluation – the evaluation of the learning performance is based on the output of this segmentation.

The lexical learning model behind this learning process is assumed to involve two steps of optimal segmentation. When an utterance is perceived by a learner, it is segmented into a sequence of lexical candidates. This first step derives an optimal representation for the input utterance. When a lexical candidate is to be added to the lexicon, it is further segmented into some other shorter core lexical items if the segmentation is preferable. Only these core lexical items have the opportunity to get into the lexicon – the core lexicon. This second step drives an optimal representation for a lexical candidate, and functions, more importantly, as a filter to prevent clumps of words from getting into the core lexicon. This model can be depicted by Figure 7.7.
CHAPTER 7. LEARNING ALGORITHMS AND EVALUATION

This learning model is consistent with reported discovery in language acquisition research, e.g., as in [278]: Young infants do learn word clumps such as [what’s that] as lexical units at an early stage of lexical acquisition, and when they are exposed to more language data, they undergo a lexical refinement process to decompose the word clumps into finer-grained lexical items.

7.3.3 Induction of LexicalCandidates via Optimal Segmentation

This is the first step of the lexical learning. It applies the optimal segmentation to each utterance in the input corpus and outputs a sequence of chunks of the utterance as lexical candidates. The algorithm is formulated as follows:

1. (Count n-grams on the input corpus $C$ with the VC approach.)
2. (Calculate DLG in terms of n-gram counts in the VC.)
3. Initialise the set of lexical candidates $L_1 = \emptyset$.
4. For each utterance $u \in C$,
   4.1. $S = \text{OpSeg}(u)$, with all n-gram items in the VC as the word space;
   4.2. Add each chunk $c \in S$ to $L_1$ as a lexical candidate:
      - If $c \in L_1$, $f(c)++$;
      - Otherwise, $L_1 = L_1 \cup \{c\}$ and $f(c) = 1$.
5. Output $L_1$ as the result of the induction.

This application of the optimal segmentation has no constraint on the word space: all n-gram items in the original corpus are considered as possible word candidates.

However, for the purpose of exploring human lexical learning mechanisms with this computational approach, it is reasonable to assume that only those n-gram items containing at least a vowel can be a word candidate. It is known that syllables are the basic units of speech representation and every word must contain at least a syllable, that is, a word contains at least a vowel (character). It is rather straightforward to implement this constraint upon the optimal segmentation process: every chunk in the segmentation must contain at least a vowel character. From now on we will denote the optimal segmentation with the vowel constraint as $\text{OS}^V$ and the one without as $\text{OS}$. In the case that we specify that the apostrophe [‘] is an equivalent to a vowel character, in addition
to [aeiou], we denote the algorithm as $\text{OS}^+\text{V}$. Clearly, it is a less constrained version of $\text{OS}^+\text{V}$. This specification is to enable the learning algorithm to recognise the bound morphemes like [-n’t] and [’ll] as individual lexical items in English. Without this specification, the $\text{OS}^+\text{V}$ algorithm will have no chance to show its ability to learn these morphemes, because it always has to adhere them to lexical items with a vowel under the vowel constraint.

We can also iterate these two algorithms, in the essence of the EM algorithm, to test their learning capacity – we will see how far they can go in lexical learning on their own. The implementation is very simple: repeat each of the algorithms again and again and update the frequency of the n-grams according to the segmentation result in each iteration. It can be expected that the learning performance will be improved iteration by iteration, until there is no more improvement, then we stop. We refer to these algorithms as $\text{OS}^+\text{EM}$, $\text{OS}^+\text{V}+\text{EM}$ and $\text{OP}^+\text{V}+\text{EM}$, and the experimental results of these algorithms will be presented in the next section.

### 7.3.4 Lexical Refinement

This step for lexical refinement in the unsupervised lexical learning is aimed at handling a specific problem that the previous step does not deal with – the previous step for optimal segmentation outputs many clumps of words as lexical candidates. These clumps of words co-exist with the their constituent words in the intermediate lexicon $L_1$. For example, word clumps such as *what’s that, the doggie* co-exist with individual words *what, ’s, that, the* and *doggie*.

The necessity to further decompose these word clumps into finer-grained individual lexical items is obvious, in that this further decomposition can lead to a more compact representation of the input corpus. We can achieve this purpose by one more application of the Viterbi optimal segmentation algorithm on the lexical candidates, as follows.

1. Calculate the DLD for lexical candidates in $L_1$ in terms of their new counts.
2. Initialise the set of finer-grained lexical candidates $L_2 = \emptyset$.
3. For each lexical candidate $w \in L_1$,
   1. $S = \text{OpSeg}(w)$, with $L_1 - \{w\}$ as the word space;
   2. If $\text{DLG}(S) > 0$, add each chunk $c \in S$ to $L_2$ as a lexical candidate:
      
      If $c \in L_2$, $f(c) + = f(w)$;
Otherwise, \( L_2 = L_2 \cup \{ c \} \) and \( f(c) = f(w) \).

\[
(3.3) \quad f(w) = 0.
\]

(4) Output \( L_2 \) as the finer-grained lexicon.

We denote this lexical refinement algorithm as LR.

The condition \( DLG(S) > 0 \) in Step (3.2) is necessary, because in our learning-via-compression approach we want every decomposition or chunking to bring some positive compression effect. Experimental results show that this condition prevents many superfluous decompositions with a negative DLG, such as [re][all][y], [the][re] and [ok][ay]. Also, Step (3.2) above involves frequency accumulation; otherwise, there will a frequency leakage, and, consequently, the DLG computation in terms of the resulting frequencies of the lexical items is no longer reliable.

Another constraint that is necessary for the lexical refinement process is that once a lexical candidate is decomposed into several shorter lexical items – this means that these items are considered as core lexical items at the time that the decomposition is performed - the shorter lexical items are not further decomposed into any other items at the same time. If they were also decomposed at the same time, that means they were not considered as core lexical items – consequently, the lexical candidate in question should be decomposed into other lexical items instead of them.

A straightforward way to carry out the lexical refinement observing this constraint is to run the above algorithm twice. The first time we mark the lexical candidates that appear as core lexical items in the decomposition of any lexical candidates. The second time we run the algorithm on those non-marked lexical candidates to carry out the decomposition on all of them. The two steps finish a round of lexical refinement.

Once a round of lexical refinement is carried out, the counts of many remaining lexical candidates are changed, and some of these candidates may become decomposable – by the term “decomposable” we mean that a lexical candidate can have a decomposition with a positive DLG. In order to carry out all positive decompositions, it is necessary to repeat the lexical refinement round after round until there is a round in which there is no more decomposition carried out.

7.3.5 Word Segmentation with a Refined Lexicon

Basically, the unsupervised lexical learning has mostly been done by the previous two steps for lexical candidate induction and lexical refinement. However, what we have is
a finer-grained lexicon $L_2$, we do not yet know what the result would be if this lexicon were used to segment the input utterance into words.

In order to do the segmentation, it is necessary to apply the optimal segmentation algorithm $O S$ once again to the input utterances with the $L_2$ as the word space. The algorithm, denoted as $W S$, is straightforward as follows.

1. Calculate the DLD for lexical items in $L_2$ in terms of their new counts.
2. Initialise the final lexicon $L_3 = \phi$.
3. For each utterance $u \in C$,
   1. $S = O p S e g (u)$, with $L_2$ as the word space;
   2. Output $S$ as the segmentation result for the utterance $u$;
   3. Add each chunk $c \in S$ to $L_3$ as a lexical item:
      - If $c \in L_3$, $f (c) +=$;
      - Otherwise, $L_3 = L_3 \cup \{c\}$ and $f (c) = 1$.
4. Output $L_3$.

A problem with this algorithm that we have encountered in many experiments but have not resolved yet is that there are many cases where the output of the segmentation has a negative DLG, i.e., \(DLG(S) < 0.0\), and consequently the segmentation output is not reliable - many real words are embedded in the non-word segments in the output, e.g., [you][\texttt{reringingthephones}] and [there][\texttt{yagoalloyourfriends}]. In many cases this negative DLG problem is actually an \"unknown\" word problem, because its cause is that the low-frequency words like \texttt{phones} and \texttt{friends} are not in the word space. Recall that the word space has been narrowed down by the lexical refinement process. If we could have a sound solution for this problem, to dig out the embedded words, our learning algorithms should be able to reach a significantly higher performance than what we will report in the next section.

So far, we can have several unsupervised lexical learning algorithms that go through the three stages, namely, optimal segmentation of input utterance, lexical refinement and word segmentation with the refined lexicon. They are $O S + L R + W S$, $O S + V + L R + W S$ and $O S + V' + L R + W S$. Notice that the only difference of $O S$ and $W S$ is their word space.

Inspired by the EM algorithm, we can repeat the LR and WS part in these algorithms to see if the learning performance can be further improved. We denote these algorithms
as OS+(LR+WS)x2, OS+V+(LR+WS)x2 and OS+V+(LR+WS)x2, where “x2” indicates the repetition.

7.4 Evaluation

In this section we will report the evaluation of our lexical learning algorithms’ performance based on the experimental results on the Bernstein corpus [18], a naturally-occurring child-directed speech corpus, from the CHILDES collection [222, 220]. We will first give some rationales for selecting this text corpus as testing data, then define a number of empirical measures for the evaluation, and finally present the learning performance in terms of these measures.

7.4.1 Input Data

It is understood that written text corpora like the Brown corpus and the PTB corpus are not appropriate for testing an unsupervised lexical learning approach that is aimed at exploring the language-learning infants’ lexical learning mechanisms. Instead, we have to use the language data that the children really receive in their language-learning environments. The CHILDES database is a collection of such data, contributed by a large number of scholars in the field of language acquisition. It is the most suitable database for our purpose.

The most important reason for picking the Bernstein corpus out of the CHILDES collection as our testing data is that we intend to compare our work with the state-of-the-art approach that Brent has recently reported in [32], where the Bernstein corpus was used as testing data. The details of the input data have been illustrated in Section 7.2 with examples.

A question one may ask about the testing data is, why use orthographic text as input, not speech input? In order to answer this question, several points need to be clarified. First, we do not really need the speech input in the form of sound waves as input data for our study. We know that human infants’ categorical speech perception turns the speech signals they receive in a sequence of sounds, known as phones, for each utterance. Our research is aimed at exploring the learning mechanisms (or strategies) that the language-learning infants may exploit to map the sound sequences to lexical items at the early stage of lexical learning when they have no knowledge about words in their languages. Involvement of too much speech processing details would not help highlight the purpose of our study. Furthermore, for technical reasons, all speech data
suitable for computational studies of language learning are actually in text format as some kind of transcript, e.g., phonetic transcript.

So why don’t we use phonetic transcripts as input? The only reason is the unavailability. If they were available, we would use them for the test with no hesitation. However, we are also happy with testing on the orthographic transcripts of the same corpus, for the following reasons. First, our learning approach is aimed at exploring the general learning mechanisms that human infants may use for learning words from language data, no matter what format the data is in and what distributional regularities that data may have. We are not interested in a learning mechanism that can deal with input data in one format but not others or in one language but not others. We believe all human infants use similar learning mechanisms, regardless of the language involved, to deal with the lexical learning problem at the initial stage of language acquisition.

Second, our learning approach works through detecting regularities in the input data. Language data from different languages exhibit different regularities, and no matter what regularities there are in the input, our learning approach must be able to capture them; otherwise, it is not really a general approach. We know that both the phonetic and orthographic transcripts are transcribed from the same speech data, and that most regularities in the original speech data are well maintained in both types of transcript, in different form though. Furthermore, in comparison with an artificial phonetic transcription scheme, the orthographic transcript is more natural because it has evolved for thousands of years. The orthographic transcript carries its own inconsistency with the speech data and its own irregularities (e.g., \( \ddot{a} \) appears as -er, -or, -ur, -ir or in some other forms in orthographic texts). All these provide more challenges to, and therefore are a more creditable test for, our learning algorithms.

As discussed in Section 7.2, the orthographic transcripts of the Bernstein corpus have to undergo some necessary pre-processing before being input to the lexical learner as testing data. The main purpose of the pre-processing is to filter out the non-speech content from the corpus and do some necessary adjustment. We follow an essential principle in the data pre-processing, that is, we only do the minimally necessary adjustment.

The job of pre-processing that we have carried out includes the following aspects:

- Filtering out the non-speech content, including punctuation marks and commentary notes in square brackets;
- Converting capitalised words, except for proper nouns, to lowercase ones;
CHAPTER 7. LEARNING ALGORITHMS AND EVALUATION

- Adjusting some special forms of words or abbreviated words (e.g., the abbreviated form of negation) back to standard orthographic forms, e.g., “i (the)m” → “i am”, “wouldn (i)t” → “wouldn’t”.

Notably, the speech-related marking symbols, mainly braces and colons inside words, e.g., as in (a)n(d), beau:ti:ful, fa:v(o)rite and tel:e:phone are all filtered out. Table 7.5 summarises a number of problems in the original corpus and the adjustments that have been carried out by the pre-processing process.

However, many non-standard word forms remain in the data, e.g., w’ (we), y’ (you), ya (you), whatcha (what do you) and whatchya (what do you). This kind of inconsistency really plays a critical role in testing the learner’s learning ability.

Also, we allow one non-speech symbol, “*”, to remain in the data, as in peek+a+boo and bye+bye, because it reflects the corpus constructors’ intention that these strings should be considered as individual words or word-like compounds instead of as several words. And the unknown words in the form xxx, incomplete words such as uh, onomatopoeia (e.g., woof woof woof) and interjections (e.g., oh, ooh and uh) are also kept, contrary to how Brent prepared his testing data. If all these irregularities were cleaned up, the lexical learner would have a better performance.

The entire testing corpus of child-directed speech extracted from the Bernstein corpus is of 9702 utterances, 35K words and 143K characters. The average utterance length is 14.7 characters and 3.6 words; the average word length is 4.1 characters.

7.4.2 Evaluation Measures

We are going to use the following empirical measures to evaluate a lexical learner’s performance on a given testing corpus.

- Word precision and recall

- Word boundary precision and recall

- Correct character ratio

- Word type precision and recall

The word precision is defined as the proportion of correct words among the learned words, and the word recall as the proportion of real words that are learned by the learner. Given that the input corpus has $N$ words, and the learner learns $M$ words
<table>
<thead>
<tr>
<th>Problem</th>
<th>Example</th>
<th>Freq.</th>
<th>Adjustment</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td>doggie</td>
<td>6</td>
<td>Removal</td>
<td>doggie</td>
</tr>
<tr>
<td></td>
<td>kitties</td>
<td>6</td>
<td></td>
<td>kittens</td>
</tr>
<tr>
<td></td>
<td>kitty</td>
<td>4</td>
<td></td>
<td>kitty</td>
</tr>
<tr>
<td>O</td>
<td>(a)bout</td>
<td>18</td>
<td>Removal</td>
<td>about</td>
</tr>
<tr>
<td></td>
<td>(a)n(d)</td>
<td>15</td>
<td></td>
<td>and</td>
</tr>
<tr>
<td></td>
<td>an(d)</td>
<td>12</td>
<td></td>
<td>and</td>
</tr>
<tr>
<td></td>
<td>d(o)</td>
<td>34</td>
<td></td>
<td>d</td>
</tr>
<tr>
<td></td>
<td>(doe)s</td>
<td>8</td>
<td></td>
<td>does</td>
</tr>
<tr>
<td></td>
<td>goin(g)</td>
<td>3</td>
<td></td>
<td>going</td>
</tr>
<tr>
<td></td>
<td>(i)s</td>
<td>29</td>
<td></td>
<td>is</td>
</tr>
<tr>
<td></td>
<td>(i)t’s</td>
<td>24</td>
<td></td>
<td>it’s</td>
</tr>
<tr>
<td></td>
<td>(it’)s</td>
<td>4</td>
<td></td>
<td>it’s</td>
</tr>
<tr>
<td></td>
<td>(it)’s</td>
<td>8</td>
<td></td>
<td>it’s</td>
</tr>
<tr>
<td></td>
<td>(o)k</td>
<td>14</td>
<td></td>
<td>ok</td>
</tr>
<tr>
<td></td>
<td>(o)kay</td>
<td>2</td>
<td></td>
<td>okay</td>
</tr>
<tr>
<td></td>
<td>(th)at</td>
<td>6</td>
<td></td>
<td>that</td>
</tr>
<tr>
<td></td>
<td>(th)em</td>
<td>37</td>
<td></td>
<td>them</td>
</tr>
<tr>
<td></td>
<td>wan(t)</td>
<td>26</td>
<td></td>
<td>want</td>
</tr>
<tr>
<td></td>
<td>y(ou)</td>
<td>11</td>
<td></td>
<td>you</td>
</tr>
<tr>
<td>O + ’</td>
<td>(doe)’s</td>
<td>1</td>
<td>Removal</td>
<td>does</td>
</tr>
<tr>
<td></td>
<td>(i)’s</td>
<td>5</td>
<td></td>
<td>is</td>
</tr>
<tr>
<td></td>
<td>wha’(t)s</td>
<td>1</td>
<td></td>
<td>what’s</td>
</tr>
<tr>
<td></td>
<td>wha(t)’s</td>
<td>2</td>
<td></td>
<td>what’s</td>
</tr>
<tr>
<td>-n (i)t</td>
<td>aren (i)t</td>
<td>16</td>
<td>Transformation</td>
<td>aren’t</td>
</tr>
<tr>
<td></td>
<td>can (i)t</td>
<td>22</td>
<td>[-n (i)t] → [-n’t]</td>
<td>can’t</td>
</tr>
<tr>
<td></td>
<td>couldn (i)t</td>
<td>2</td>
<td></td>
<td>couldn’t</td>
</tr>
<tr>
<td></td>
<td>doesn (i)t</td>
<td>42</td>
<td></td>
<td>doesn’t</td>
</tr>
<tr>
<td></td>
<td>don (i)t</td>
<td>151</td>
<td></td>
<td>don’t</td>
</tr>
<tr>
<td></td>
<td>don (i)tcha</td>
<td>2</td>
<td></td>
<td>don’tcha</td>
</tr>
<tr>
<td></td>
<td>has (i)t</td>
<td>1</td>
<td></td>
<td>hasn’t</td>
</tr>
<tr>
<td></td>
<td>haven (i)t</td>
<td>5</td>
<td></td>
<td>haven’t</td>
</tr>
<tr>
<td></td>
<td>isn (i)t</td>
<td>17</td>
<td></td>
<td>isn’t</td>
</tr>
<tr>
<td></td>
<td>mustn (i)t</td>
<td>1</td>
<td></td>
<td>mustn’t</td>
</tr>
<tr>
<td></td>
<td>shouldn (i)t</td>
<td>1</td>
<td></td>
<td>shouldn’t</td>
</tr>
<tr>
<td></td>
<td>wasn (i)t</td>
<td>1</td>
<td></td>
<td>wasn’t</td>
</tr>
<tr>
<td></td>
<td>won (i)t</td>
<td>5</td>
<td></td>
<td>won’t</td>
</tr>
<tr>
<td></td>
<td>wouldn (i)t</td>
<td>2</td>
<td></td>
<td>wouldn’t</td>
</tr>
<tr>
<td>(the)m i (the)m</td>
<td>86</td>
<td>Transformation</td>
<td>i am</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.5: A summary of problems in the orthographic transcripts of the Bernstein corpus and their adjustment in the pre-processing
CHAPTER 7. LEARNING ALGORITHMS AND EVALUATION

among which $C$ words are correct, the word precision and recall are $C/M$ and $C/N$, respectively, computed in terms of the numbers of word tokens.

However, the measures of word precision and recall do not give any credit to a learning output like `[it think][it will][come out]`, where although none of the chunks in the segmentation is a real word, the learner really detects some regularity within the input data – many word boundaries are correctly discovered. In order to complement the measures of word precision and recall, we need to have the word boundary precision and recall as evaluation measures.

The word boundary precision is defined as the proportion of correct word boundaries among the word boundaries detected by the learner, and the word boundary recall as the proportion of correct word boundaries that the learner detects. Given that the input corpus has $N'$ word boundaries, and the learner detects $M'$ word boundaries among which $C'$ boundaries are correct, the word boundary precision and recall are $C'/M'$ and $C'/N'$, respectively.

The correct character ratio is the proportion of the entire input corpus in terms of the number of characters that is in correctly learned words. Given an input corpus of $L$ characters long and $L'$ characters out of $L$ are in the correctly learned words, the correct character ratio of the learning is $L'/L$.

The word type precision and recall are the precision and recall in terms of the learned and standard word types in the learned and standard lexicons, respectively, instead of word tokens in the original corpus and the segmentation result by the learning. The word type precision was also called `lexicon precision` in Brent’s recent work [32].

These measures form a systematic evaluation for computational lexical learning.

**Words versus Morphemes in the Evaluation**

However, the above measures would be in vain if it is not clear about what words are in a language. Thus what words are is a critical issue in this evaluation.

The basic rule we opt to follow is that we recognise the orthographic words in the input corpus: spaces are word delimiters. That is, any strings separated by spaces in the input corpus are, basically, recognised as words. This rule is consistent with our principle of minimally necessary change in the data pre-processing: we respect the original corpus as much as we can. Therefore, although we realise that there are many non-conventional word forms in the Bernstein corpus, e.g., “i gotchyou gotchya gotchya”, “what d’ya want to do”, “didja knock’em over” and “wa’dja like to
call it”, where several words are wrapped up into one “word”, we use them as “it is”.

Unfortunately, this simple rule does not help solve another problem that is caused by abbreviated (i.e., phonetically reduced) words, e.g., [-’s] as in [that’s], [what’s] and [there’s], [-’re] in [you’re] and [there’re], [-’ll] as in [i’ll] and [we’ll], and [-n’t] as in [can’t], [don’t], [doesn’t] and [isn’t]. The problem is, if the learner outputs segmentation results like [that][’s] and [does][n’t], how do we evaluate such cases? Should we count them as correct, or as wrong?

It is difficult to have a clear decision here about correctness. If we only count real words in the evaluation, they should certainly be counted as wrong words, because we know for sure that [’s] and n’t] are not words. However, our task here is not counting real words; instead, we are to evaluate the performance of an unsupervised lexical learning approach which is expected to simulate human lexical learning mechanisms.

What do infant learners learn in lexical acquisition? Only words? No, they also learn many morphemes as lexical items in addition to words. Morphemes are defined in linguistics as the minimal units that have meaning in a language. There are two types of morpheme: one is free morpheme, and the other is bound morpheme. The free morphemes are words. Most morphemes in a language like English are words. The bound morphemes are the ones that cannot occur alone by themselves, e.g., [-ing] and [-ed]. The aforementioned reduced word forms such as [-’s] and [-n’t] become bound morphemes mostly due to the fact that they lack a vowel to form independent syllables. It is understood that the apostrophe [’] in [-n’t] is used to represent a reduced vowel that is not qualified to make a syllable\(^4\). The number of bound morphemes in a language like English is rather small, and they only can occur in adherence to a word as in [don’t]. They are a close class of lexical items in the lexicon of a language.

Now, our problem concerning the question of “what are words?” in the evaluation becomes how do we credit the two types of morphemes that a learner learns in the learning? If we only count the correct words in the learning output, that means that the credit is only given to free morphemes but not to any bound morphemes – an evaluation like this turns out to be imperfect, unfair, and even flawed, in a sense. If we credit a bound morpheme in the same way as a word, it can be quite controversial – those bound morphemes are not words! And we even have no idea about whether a bound morpheme should credited 0.5 or 0.9 as much as a word.

\(^4\)Thus, in order to enable the GI + V algorithm to learn bound morphemes such as -n’t, we have to tell the learner that [’] is something equivalent to a vowel character.
CHAPTER 7. LEARNING ALGORITHMS AND EVALUATION

All these unsettled problems indicate that it is inappropriate, and unrealistic, to have a clear-cut means of evaluation for unsupervised lexical learning that has both words and bound morphemes as learning output. The only conceivable way out of this dilemma is that we conduct separate evaluations for the learner’s performance on learning words and on learning lexical items including both words and bound morphemes.

To evaluate the performance of learning words, we simply count the chunks in the learning output that match the space-delimited words in the input corpus, and then produce the evaluation results in terms of the above measures. But in the evaluation of the performance of learning words and bound morphemes, the case is still a bit complicated. In general, we take a flexible approach: either the learner recognises, for example, [that’s] as one word or as two lexical items [that][‘s], we consider both as correct responses.

But this approach is problematic in dealing with the contracted negative marker [-n’t], because sometimes the [-] part is not a well-formed word, e.g., [ca][n’t] and [wo][n’t]. Thus, we need to adjust it accordingly, that is, either the learner recognises [Vn’t] as one word a [Vn’t] or two lexical items as [V][n’t] is considered as correct, conditioned on the fact that the [V] part must form a well-formed word. That is, output chunks like [can’t] and [won’t] are all considered correct lexical items, but [ca][n’t] and [wo][n’t] are each recognised as a wrong item and a correct one, instead of as two correct items. In contrast, [do][n’t] and [is][n’t] are each recognised as two correct items.

The bound morphemes in English texts that we need to consider in the evaluation include, at least, the following: [-’s], [-’d], [-’re], [-’ll], [-’ve] and [-n’t], all of which carry an apostrophe. We divide these bound morphemes into two groups. One group, denoted as G1, includes all the above ones but the last. These morphemes only co-occur with a noun. The last one [-n’t] forms another group, denoted as G2, which only co-occurs with a verb.

7.4.3 Learning Performance

The learning performance of the twelve unsupervised lexical learning algorithms, each with a slightly different combination of our GS, LR, WS algorithms and the EM algorithm, are presented in Table 7.6 and Table 7.7 with the measures of word and word boundary precision and recall. Table 7.6 presents their performance in learning real words, and Table 7.7 their performance in learning words and bound morphemes.
<table>
<thead>
<tr>
<th>Learning Algorithms</th>
<th>Word (Token) P (%)</th>
<th>R (%)</th>
<th>Word Boundary P (%)</th>
<th>R (%)</th>
<th>Cor. Char. Ratio (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>32.43</td>
<td>31.03</td>
<td>70.48</td>
<td>67.43</td>
<td>33.14</td>
</tr>
<tr>
<td>OS+EM</td>
<td>58.66</td>
<td>56.06</td>
<td>83.11</td>
<td>79.42</td>
<td>54.71</td>
</tr>
<tr>
<td>OS+LR+WS</td>
<td>61.60</td>
<td>67.99</td>
<td>81.28</td>
<td>89.72</td>
<td>64.10</td>
</tr>
<tr>
<td>OS+(LR+WS)x2</td>
<td>58.98</td>
<td>68.49</td>
<td>78.99</td>
<td>91.72</td>
<td>63.17</td>
</tr>
<tr>
<td>OS+V</td>
<td>47.99</td>
<td>36.72</td>
<td>85.41</td>
<td>65.36</td>
<td>38.23</td>
</tr>
<tr>
<td>OS+V+EM</td>
<td>63.54</td>
<td>49.78</td>
<td>90.71</td>
<td>71.06</td>
<td>49.01</td>
</tr>
<tr>
<td>OS+V+LR+WS</td>
<td>74.99</td>
<td>70.92</td>
<td>90.29</td>
<td>85.39</td>
<td>70.04</td>
</tr>
<tr>
<td>OS+V+(LR+WS)x2</td>
<td>75.31</td>
<td>73.39</td>
<td>89.63</td>
<td>87.34</td>
<td>71.69</td>
</tr>
<tr>
<td>OS+V'</td>
<td>47.42</td>
<td>36.62</td>
<td>84.83</td>
<td>65.50</td>
<td>38.12</td>
</tr>
<tr>
<td>OS+V'+EM</td>
<td>63.42</td>
<td>51.12</td>
<td>90.72</td>
<td>73.13</td>
<td>50.51</td>
</tr>
<tr>
<td>OS+V'+LR+WS</td>
<td>70.17</td>
<td>70.13</td>
<td>87.58</td>
<td>87.53</td>
<td>68.12</td>
</tr>
<tr>
<td>OS+V'+(LR+WS)x2</td>
<td>69.26</td>
<td>72.22</td>
<td>86.06</td>
<td>89.75</td>
<td>68.80</td>
</tr>
</tbody>
</table>

Table 7.6: Learning performance of the unsupervised lexical learning algorithms on words

<table>
<thead>
<tr>
<th>Learning Algorithms</th>
<th>Word (Token) P (%)</th>
<th>R (%)</th>
<th>Word Boundary P (%)</th>
<th>R (%)</th>
<th>Cor. Char. Ratio (%)</th>
<th>Counted Morphemes</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>33.86</td>
<td>32.17</td>
<td>71.19</td>
<td>67.65</td>
<td>34.33</td>
<td>+G1</td>
</tr>
<tr>
<td>OS+EM</td>
<td>65.52</td>
<td>60.50</td>
<td>86.57</td>
<td>79.74</td>
<td>59.86</td>
<td>+G1</td>
</tr>
<tr>
<td>OS+LR+WS</td>
<td>65.50</td>
<td>60.52</td>
<td>86.58</td>
<td>79.94</td>
<td>59.88</td>
<td>+G1+G2</td>
</tr>
<tr>
<td>OS+(LR+WS)x2</td>
<td>69.89</td>
<td>73.77</td>
<td>85.42</td>
<td>90.17</td>
<td>71.33</td>
<td>+G1</td>
</tr>
<tr>
<td>OS+V'</td>
<td>70.36</td>
<td>74.09</td>
<td>85.66</td>
<td>90.19</td>
<td>71.75</td>
<td>+G1+G2</td>
</tr>
<tr>
<td>OS+V'+EM</td>
<td>67.08</td>
<td>74.40</td>
<td>83.04</td>
<td>92.10</td>
<td>70.60</td>
<td>+G1</td>
</tr>
<tr>
<td>OS+V'+LR+WS</td>
<td>68.39</td>
<td>75.31</td>
<td>83.69</td>
<td>92.15</td>
<td>71.81</td>
<td>+G1+G2</td>
</tr>
<tr>
<td>OS+V'+(LR+WS)x2</td>
<td>49.33</td>
<td>37.81</td>
<td>85.78</td>
<td>65.76</td>
<td>39.28</td>
<td>+G1</td>
</tr>
<tr>
<td>OS+V'</td>
<td>49.55</td>
<td>37.95</td>
<td>85.89</td>
<td>65.78</td>
<td>39.41</td>
<td>+G1+G2</td>
</tr>
<tr>
<td>OS+V'+EM</td>
<td>69.45</td>
<td>54.66</td>
<td>93.68</td>
<td>73.73</td>
<td>54.38</td>
<td>+G1</td>
</tr>
<tr>
<td>OS+V'+LR+WS</td>
<td>69.95</td>
<td>54.94</td>
<td>93.94</td>
<td>73.78</td>
<td>54.70</td>
<td>+G1+G2</td>
</tr>
<tr>
<td>OS+V'+(LR+WS)x2</td>
<td>79.42</td>
<td>75.87</td>
<td>92.20</td>
<td>88.08</td>
<td>75.42</td>
<td>+G1</td>
</tr>
<tr>
<td>OS+V'</td>
<td>79.92</td>
<td>76.17</td>
<td>92.45</td>
<td>88.11</td>
<td>75.82</td>
<td>+G1+G2</td>
</tr>
<tr>
<td>OS+V'+LR+WS</td>
<td>78.76</td>
<td>78.25</td>
<td>90.81</td>
<td>90.23</td>
<td>76.62</td>
<td>+G1</td>
</tr>
<tr>
<td>OS+V'+(LR+WS)x2</td>
<td>80.11</td>
<td>79.07</td>
<td>91.49</td>
<td>90.30</td>
<td>77.74</td>
<td>+G1+G2</td>
</tr>
</tbody>
</table>

Bound morphemes in group G1: [-'a], [-'d], [-'re], [-'11], [-'ve]
Bound morphemes in group G2: [-n't]

Table 7.7: Learning performance of the unsupervised lexical learning algorithms on word and bound morphemes
The performance of the OS and OS+V’ algorithms incorporated in the EM algorithm is presented in Figure 7.8. We can see that each of the two programs converges very quickly towards the top of their performance. In general, both algorithms’ performance drops significantly in the second iteration, except for OS+EM’s word precision and word boundary precision, which continue going up. Then, all measures increase rapidly in the next five iterations. After that, the growth slows down significantly in the next ten iterations, and all measures reach their local maxima gradually.

We have observed that the OS and OS+EM algorithms have a better balance between precision and recall than the OS+V’ and OS+V’+EM algorithms, as shown in the middle table in Table 7.8. The former two algorithms’ precision and recall for word and word boundary show a difference, defined as |P – R|, less than 3.69 percentage points and a divergence rate, defined as |P – R|/min(P, R), less than 4.65%; whereas the latter two algorithms have a difference of precision and recall in the range of 10 to 20 percentage points and a divergence rate in the range of 20% to 30%. But the EM algorithm seems not to enlarge this difference and divergence rate: OS+EM has a balance of precision and recall as good as OS, and OS+V’+EM has a slightly better balance of precision and recall than OS+V’.

We can see the effect of the EM algorithm on improving the learning performance of the OS and OS+V’ algorithms. As shown in the middle table in Table 7.8, the EM algorithm increases the OS algorithm’s word precision and recall both by about 81% and its word boundary precision and recall both by about 18%. In contrast, the EM algorithm appears less effective, but still quite effective, on the OS+V’ algorithm, in that it increases the word precision and recall by about 34% and 40%, respectively, and increases the word boundary precision and recall by about 7% and 12%, respectively. This loss in effectiveness is, most likely, due to the effect of the vowel constraint.

The bottom table in Table 7.8 presents the effect of the vowel constraint on learning performance, with V’ (a looser constraint) as an example. It shows that except for the case of working with the EM algorithm, the vowel constraint consistently enhances the learning performance by improving the word precision and recall and word boundary precision, at the price of lowering the word boundary recall slightly. While working with the EM algorithm, the vowel constraint improves both the word precision and word boundary precision, by about 8% and 9%, respectively, at the price of lowering the correspondent recalls by about -9% and -8%, respectively, and also lowering the correct character ratio by 7.68%. These results indicate that the vowel constraint is not a good
Figure 7.8: Performance of the $0S$ and $0S+V'$ algorithms in EM iterations.
CHAPTER 7. LEARNING ALGORITHMS AND EVALUATION

co-operator with the EM algorithm.

The EM algorithm can only reach a local minimum; and human learners do not learn in the same way by going through the input data many times. We are more interested in pursuing learning algorithms that can simulate human lexical learning better than the EM algorithm, in terms of both its learning strategies and learning performance. We have implemented several such learning algorithms, including the OS+LR+WS, OS+V+LR+WS and OS+V’+LR+WS algorithms. Their performance in learning words and in learning words and bound morphemes has been shown in Table 7.6 and 7.7, respectively, and the comparison of their performance and their correspondent EM algorithms’ performance is presented in Table 7.9. We can see that the unsupervised lexical learning algorithms have a much better performance than the EM algorithm: the word precision is better by 5-18%, the word recall is better by 21-42%, the word boundary precision is slightly lower, at most by 3.5%, the word boundary recall is better by 13-20%, and the correct character ratio is better by 17-42%. The superiority of unsupervised lexical learning through optimal segmentation, lexical refinement and word segmentation is clearly overwhelming.

The effect of repeating the (LR+WS) part in the unsupervised lexical learning appears insignificant. It increases the recall slightly but decreases, sometimes, the precision to a similar scale. For example, in the experiment of learning words and bound morphemes, the OS+V+(LR+WS)x2 algorithm, in comparison with the OS+V+LR+WS algorithm, increases the word recall by 1.2 percentage points at the price of lowering the word precision by 2 points and increases the word boundary recall by about 2 points at the price of lowering the word boundary precision by about 2 points. The OS+V’+(LR+WS)x2 algorithm demonstrates the best gain by the (LR+WS)x2 part on learning words and morphemes: a gain of 2 percentage points in word recall with no loss in word precision, a gain of 2.2 points in word boundary recall at the price of 1 point loss in word boundary precision, and a gain of about 2 points in correct character ratio.

When bound morphemes are counted as correctly learned lexical items in the learning output, the learning performance of the OS+LR+WS and OS+V’+LR+WS algorithms goes up significantly. These two algorithms’ precision and recall on lexical items, precision and recall on lexical boundaries, and correct character rate are, respectively, about 13%, 9%, 5.5%, 0.5% and 11-12% higher than those on words. Of all these increments, only the increment of the boundary recall is not notably significant.

In general, the learning performance of our algorithms compares favourably with the
<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Word</th>
<th></th>
<th>Word Boundary</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P (%)</td>
<td>R (%)</td>
<td>P - R</td>
<td>D (%)</td>
<td>P (%)</td>
</tr>
<tr>
<td>OS</td>
<td>32.43</td>
<td>31.03</td>
<td>1.40</td>
<td>4.51</td>
<td>70.48</td>
</tr>
<tr>
<td>OS+EM</td>
<td>58.66</td>
<td>56.06</td>
<td>2.60</td>
<td>4.64</td>
<td>83.11</td>
</tr>
<tr>
<td>OS+V'</td>
<td>47.42</td>
<td>36.62</td>
<td>10.80</td>
<td>29.49</td>
<td>84.83</td>
</tr>
<tr>
<td>OS+V'+EM</td>
<td>63.42</td>
<td>51.12</td>
<td>12.30</td>
<td>20.06</td>
<td>90.72</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>OS+EM</th>
<th></th>
<th>OS+V'+EM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Word</td>
<td></td>
<td>Word Boundary</td>
</tr>
<tr>
<td></td>
<td>P (%)</td>
<td>R (%)</td>
<td>P (%)</td>
</tr>
<tr>
<td>Beginning</td>
<td>32.43</td>
<td>31.03</td>
<td>70.48</td>
</tr>
<tr>
<td>End</td>
<td>58.66</td>
<td>56.06</td>
<td>83.11</td>
</tr>
<tr>
<td>Increment</td>
<td>26.23</td>
<td>25.03</td>
<td>12.63</td>
</tr>
<tr>
<td>Incr. Rate (%)</td>
<td>80.88</td>
<td>80.66</td>
<td>17.76</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Learning Algorithms</th>
<th>Word (Token)</th>
<th>Word Boundary</th>
<th>Cor. Char. Ratio (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P (%)</td>
<td>R (%)</td>
<td>P (%)</td>
</tr>
<tr>
<td>OS</td>
<td>32.43</td>
<td>31.03</td>
<td>70.48</td>
</tr>
<tr>
<td>OS+V'</td>
<td>47.42</td>
<td>36.62</td>
<td>84.83</td>
</tr>
<tr>
<td>Increment</td>
<td>14.99</td>
<td>5.59</td>
<td>14.35</td>
</tr>
<tr>
<td>Incr. Rate (%)</td>
<td>46.22</td>
<td>18.01</td>
<td>20.36</td>
</tr>
<tr>
<td>OS+EM</td>
<td>58.66</td>
<td>56.06</td>
<td>83.11</td>
</tr>
<tr>
<td>OS+V'+EM</td>
<td>63.42</td>
<td>51.12</td>
<td>90.72</td>
</tr>
<tr>
<td>Increment</td>
<td>4.76</td>
<td>-4.94</td>
<td>7.61</td>
</tr>
<tr>
<td>Incr. Rate (%)</td>
<td>8.11</td>
<td>-8.81</td>
<td>9.16</td>
</tr>
<tr>
<td>OS+(LR+WS)x2</td>
<td>61.60</td>
<td>67.99</td>
<td>81.28</td>
</tr>
<tr>
<td>OS+(V'+LR+WS)x2</td>
<td>70.17</td>
<td>70.13</td>
<td>87.58</td>
</tr>
<tr>
<td>Increment</td>
<td>9.00</td>
<td>2.14</td>
<td>6.30</td>
</tr>
<tr>
<td>Incr. Rate (%)</td>
<td>14.61</td>
<td>3.15</td>
<td>7.75</td>
</tr>
<tr>
<td>OS+(LR+WS)x2</td>
<td>58.98</td>
<td>68.49</td>
<td>78.99</td>
</tr>
<tr>
<td>OS+(V'+LR+WS)x2</td>
<td>69.26</td>
<td>72.22</td>
<td>86.06</td>
</tr>
<tr>
<td>Increment</td>
<td>10.28</td>
<td>3.73</td>
<td>7.07</td>
</tr>
<tr>
<td>Incr. Rate (%)</td>
<td>17.43</td>
<td>5.45</td>
<td>8.95</td>
</tr>
</tbody>
</table>

Table 7.8: The effectiveness of the EM algorithm and the vowel constraint
Table 7.9: Comparison of learning performance on words and on words and bound morphemes: unsupervised lexical learning algorithms versus the EM algorithm

<table>
<thead>
<tr>
<th>Learning Algorithms</th>
<th>Word (Token) P (%)</th>
<th>Word Boundary P (%)</th>
<th>Cor. Char. Ratio (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS+EM</td>
<td>58.66</td>
<td>56.06</td>
<td>83.11</td>
</tr>
<tr>
<td>OS+LR+WS</td>
<td>61.60</td>
<td>67.99</td>
<td>81.28</td>
</tr>
<tr>
<td>Increment</td>
<td>2.94</td>
<td>11.93</td>
<td>-1.83</td>
</tr>
<tr>
<td>Incr. Rate (%)</td>
<td>5.01</td>
<td>21.38</td>
<td>-2.20</td>
</tr>
<tr>
<td>OS+V+EM</td>
<td>63.54</td>
<td>49.78</td>
<td>90.71</td>
</tr>
<tr>
<td>OS+V+LR+WS</td>
<td>74.99</td>
<td>70.92</td>
<td>89.29</td>
</tr>
<tr>
<td>Increment</td>
<td>11.54</td>
<td>21.14</td>
<td>-0.42</td>
</tr>
<tr>
<td>Incr. Rate (%)</td>
<td>18.02</td>
<td>42.47</td>
<td>-0.46</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Learning Algorithms</th>
<th>Word (Token) P (%)</th>
<th>Word Boundary P (%)</th>
<th>Cor. Char. Ratio (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS+EM</td>
<td>65.55</td>
<td>60.52</td>
<td>86.58</td>
</tr>
<tr>
<td>OS+LR+WS</td>
<td>70.36</td>
<td>74.09</td>
<td>85.66</td>
</tr>
<tr>
<td>Increment</td>
<td>4.81</td>
<td>13.57</td>
<td>-0.92</td>
</tr>
<tr>
<td>Incr. Rate (%)</td>
<td>7.34</td>
<td>22.42</td>
<td>-1.06</td>
</tr>
<tr>
<td>OS+V'+EM</td>
<td>69.95</td>
<td>54.94</td>
<td>93.94</td>
</tr>
<tr>
<td>OS+V'+LR+WS</td>
<td>79.92</td>
<td>76.17</td>
<td>92.45</td>
</tr>
<tr>
<td>Incr. Rate (%)</td>
<td>14.25</td>
<td>38.64</td>
<td>-1.59</td>
</tr>
</tbody>
</table>

Table 7.10: Comparison of learning performance on words and on words and morphemes

<table>
<thead>
<tr>
<th>Learning Algorithms</th>
<th>Lexical Type</th>
<th>Lexical Item P (%)</th>
<th>Lexical Boundary P (%)</th>
<th>Cor. Char. Ratio (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS+LR+WS</td>
<td>Word</td>
<td>61.60</td>
<td>81.28</td>
<td>64.10</td>
</tr>
<tr>
<td></td>
<td>Word + Morph.</td>
<td>70.36</td>
<td>85.66</td>
<td>71.75</td>
</tr>
<tr>
<td></td>
<td>Increment</td>
<td>8.76</td>
<td>4.38</td>
<td>7.65</td>
</tr>
<tr>
<td></td>
<td>Incr. Rate (%)</td>
<td>14.22</td>
<td>5.34</td>
<td>11.93</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Learning Algorithms</th>
<th>Lexical Type</th>
<th>Lexical Item P (%)</th>
<th>Lexical Boundary P (%)</th>
<th>Cor. Char. Ratio (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS+V'+LR+WS</td>
<td>Word</td>
<td>70.17</td>
<td>87.58</td>
<td>68.12</td>
</tr>
<tr>
<td></td>
<td>Word + Morph.</td>
<td>79.92</td>
<td>92.45</td>
<td>75.82</td>
</tr>
<tr>
<td></td>
<td>Increment</td>
<td>9.75</td>
<td>4.87</td>
<td>7.70</td>
</tr>
<tr>
<td></td>
<td>Incr. Rate (%)</td>
<td>13.89</td>
<td>5.56</td>
<td>11.30</td>
</tr>
<tr>
<td>Learning Algorithms</td>
<td>Lexical Type</td>
<td>Lexical Item</td>
<td>Difference from MBDP-1</td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P (%)</td>
<td>R (%)</td>
<td>P (%)</td>
<td></td>
</tr>
<tr>
<td>0S+LR+WS</td>
<td>Word</td>
<td>61.60</td>
<td>67.99</td>
<td>-9.40 (-13.2%)</td>
</tr>
<tr>
<td></td>
<td>Word + Morph.</td>
<td>70.36</td>
<td>74.09</td>
<td>-0.64 (-0.9%)</td>
</tr>
<tr>
<td>0S+V+LR+WS</td>
<td>Word</td>
<td>74.99</td>
<td>70.92</td>
<td>+3.99 (+5.6%)</td>
</tr>
<tr>
<td></td>
<td>Word + Morph.</td>
<td>79.92</td>
<td>76.17</td>
<td>+8.08 (+11.4%)</td>
</tr>
</tbody>
</table>

Table 7.11: Comparison of our lexical learning algorithms’ performance with the state-of-the-art performance

state-of-the-art performance of Brent’s MBDP-1 algorithm. The MBDP-1 algorithm has an average word precision of around 71% and word recall of around 72%. The upper table in Table 7.11 shows the comparison in terms of the difference in word precision and recall.

In order to make the comparison a bit clearer, we can use the $F$ measure to combine the precision and recall together for the overall learning performance. The $F$ measure is a variant of van Rijssbergen’s $E$ measure introduced in [335]: $F = 1 - E$. The $F$ measure is defined as

$$F = \frac{1}{\frac{1}{P} + (1 - \alpha) \frac{1}{R}}$$  \hspace{1cm} (7.6)

In general, we consider precision and recall equally important, and consequently choose a value of $\alpha = 0.5$. Accordingly, we have

$$F = \frac{2PR}{P + R}$$  \hspace{1cm} (7.7)

Following this formula, the average overall learning performance of the MBDP-1 algorithm is $\frac{271+272}{71+72} = 71.5$. The bottom table in Table 7.11 shows the comparison of our algorithms’ overall performance with that of MBDP-1.

The result of this comparison indicates that our best algorithm for word learning, namely, the 0S+V+LR+WS algorithm, has an overall performance of learning words that
compares favourably with MBDP-1's overall performance. If bound morphemes are considered as correctly learned lexical items, the OS+LR+WS algorithm has a learning performance as good as MBDP-1, whereas the OS+VLR+WS algorithm has a significantly better performance\(^5\).

If Brent had evaluated MBDP-1's learning performance with the word boundary precision and recall and with the correct character ratio, we could have a more thorough comparison between our learning algorithms and MBDP-1. Based on the above comparison, we may not be certain that our learning approach really outperforms the MBDP-1 algorithm, because many factors in the learning algorithms and related to the testing data preparation are different. For example, MBDP-1 is an incremental online learning algorithm, whereas ours are not; onomatopoeia and interjections are cleaned out from the testing data for MBDP-1 but kept in the testing data for our algorithms; MBDP-1 learns from phonetic transcripts and our algorithms learn from orthographic transcripts. All these factors mean that the above comparison carries a certain degree of roughness.

However, the comparison has no doubt provided adequate evidence for the conclusion that our learning approach reaches the level of the state of the art of unsupervised lexical learning.

### 7.5 Discussion

Although our lexical learning approach has turned out to show outstanding performance in learning lexical items, including words and bound morphemes, we also have observed that there is still quite some room for further improvement. In this section, we will analyse a number of problems that our unsupervised lexical learners encountered in the experiments and discuss possible solutions.

#### 7.5.1 Negative DLG Segmentation

The first problem that still needs to be resolved is the negative DLG segmentation problem: some low frequency words in an utterance, which may or may not exist in the refined lexicon, can cause the segmentation of the entire utterance to have a segmentation with a negative DLG. Table 7.12 lists many examples of negative DLG segmentations output from the OS+VLR+WS algorithm, with the frequency of the problem-causing words

\(^5\)Notice that OS+V-based algorithms, including OS+VLR+WS, do not learn any bound morphemes like those listed in G1 and G2, which have an apostrophe for the reduced vowel.
<table>
<thead>
<tr>
<th>DLG</th>
<th>Examples of segmentation</th>
<th>Freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  -35.1325</td>
<td>asmileithassasmile</td>
<td>2</td>
</tr>
<tr>
<td>2  -32.6349</td>
<td>i just realizedhowitworks</td>
<td>1</td>
</tr>
<tr>
<td>3  -14.0336</td>
<td>that 'sawiked'laugh you monkey</td>
<td>1</td>
</tr>
<tr>
<td>4  -29.7511</td>
<td>one that willflapinthebreeze</td>
<td>1</td>
</tr>
<tr>
<td>5  -35.1325</td>
<td>iwanttoshowsomeother thing</td>
<td>3</td>
</tr>
<tr>
<td>6  -22.5597</td>
<td>does the chair partflatteniton</td>
<td>1</td>
</tr>
<tr>
<td>7  -34.0216</td>
<td>okletgetdownandgettothesetoys</td>
<td>2</td>
</tr>
<tr>
<td>8  -35.1232</td>
<td>shaahaaahawhatesdoeshe say</td>
<td>6</td>
</tr>
<tr>
<td>9  -27.7897</td>
<td>i don't know whysheshaassetsomuch</td>
<td>2</td>
</tr>
<tr>
<td>10 -35.1325</td>
<td>ohigotigotanidea</td>
<td>1</td>
</tr>
<tr>
<td>11 -5.3626</td>
<td>you 'reintodegradation aren't you Alice</td>
<td>1</td>
</tr>
<tr>
<td>12 -29.5773</td>
<td>that 'swhyehadyouresteirned</td>
<td>1</td>
</tr>
<tr>
<td>13 -22.5104</td>
<td>you can spankitfitbites</td>
<td>2</td>
</tr>
<tr>
<td>14 -22.7013</td>
<td>and this boy 's put tingonhisshirt</td>
<td>3</td>
</tr>
<tr>
<td>15 -26.2659</td>
<td>could n'tpossiblybe becauseyourmother'sshadaphoneattached</td>
<td>1, 2</td>
</tr>
<tr>
<td>16 -15.7839</td>
<td>yeah that 's what the horsedoesyehap</td>
<td>1</td>
</tr>
<tr>
<td>17 -0.7694</td>
<td>wanna look at look at look this onehaspaperpages you might</td>
<td>2, 1</td>
</tr>
<tr>
<td>18 -24.4458</td>
<td>that sanowoltheblockssfell over</td>
<td>3</td>
</tr>
<tr>
<td>19 -33.1854</td>
<td>oh and the kidsaresayin'byeseethey'rewavin'</td>
<td>3, 1</td>
</tr>
<tr>
<td>20 -26.0025</td>
<td>let's tryyhythisisa snap</td>
<td>(noise)</td>
</tr>
<tr>
<td>21 -12.6852</td>
<td>you 'renomatallinterestedin this dragon ' think</td>
<td>2</td>
</tr>
<tr>
<td>22 -35.1325</td>
<td>igotchayougotchagotchya</td>
<td>2, 2</td>
</tr>
<tr>
<td>23 -18.1525</td>
<td>do you re member what wesaidthelast time</td>
<td>4</td>
</tr>
<tr>
<td>24 -25.2219</td>
<td>mightbehavingtrouble oniknees</td>
<td>1, 5</td>
</tr>
<tr>
<td>25 -22.7947</td>
<td>look sto me like you 'remakinhimdance</td>
<td>1</td>
</tr>
<tr>
<td>26 -22.0512</td>
<td>but he's gonna tripohhisshoelacesoriesbows</td>
<td>1</td>
</tr>
<tr>
<td>27 -28.0104</td>
<td>like adolphinit'ssaxhale</td>
<td>1</td>
</tr>
<tr>
<td>28 -35.1325</td>
<td>hehassoomuchhairyoucan'tseehisneck</td>
<td>6</td>
</tr>
<tr>
<td>29 -35.1325</td>
<td>yastrapityagoanditssticks</td>
<td>4, 2</td>
</tr>
<tr>
<td>30 -29.7628</td>
<td>yeah tishitwereaburry</td>
<td>1</td>
</tr>
<tr>
<td>31 -13.5204</td>
<td>you know what it's supposedtobe</td>
<td>2</td>
</tr>
<tr>
<td>32 -21.7612</td>
<td>it's supposedtobe pieceof steak</td>
<td>2</td>
</tr>
<tr>
<td>33 -31.6725</td>
<td>growlshothookmy steak</td>
<td>1</td>
</tr>
<tr>
<td>34 -16.0308</td>
<td>maybe the drayon'diketobinte the high chair</td>
<td>1</td>
</tr>
<tr>
<td>35 -7.4811</td>
<td>ithink hemightbetoom all for this high chair</td>
<td>1</td>
</tr>
<tr>
<td>36 -20.2497</td>
<td>you know Michael'sgonnagoonvacation today</td>
<td>2, 1</td>
</tr>
<tr>
<td>37 -19.2983</td>
<td>it's notakiteit'that'ssteam com ing outof the food</td>
<td>4, 2</td>
</tr>
</tbody>
</table>

Table 7.12: Examples of segmentations with negative DLG
given at the right. Actually, it is not a problem for the learner not to be able to recognise
the low frequency words. It is absolutely normal for all unsupervised lexical learning
algorithms based on co-occurring statistics not to learn most of these “bad” words. What
is really a problem in our DLG-based learning algorithm is that when a “bad” word is
lost in this way, it looks as though this bad word interferes with the recognition of other
words. For example, in Table 7.12, [lets] in line 7 grabs seven words, namely, [ok] and
[get down and get to these], into the same clump; [idea] in line 10 grabs six
other words [oh i got i got an] — whereas in almost all other utterances these six
words are properly recognised by the unsupervised learner, as exemplified by the output
from the 0S+V ’+LR+WS algorithm as below, in comparison with all utterances involving
[i got] and [got an] in the original input corpus in the right column:

<table>
<thead>
<tr>
<th>Learning output</th>
<th>Original segmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>oh got got an idea</td>
<td>oh i got i got an idea#</td>
</tr>
<tr>
<td>i got blocks at home</td>
<td>i got blocks at home#</td>
</tr>
<tr>
<td>oh i got them</td>
<td>oh i got them#</td>
</tr>
<tr>
<td>a hi got your nose</td>
<td>ah i got your nose#</td>
</tr>
<tr>
<td>i got ya finger</td>
<td>i got ya finger#</td>
</tr>
<tr>
<td>wait now i got ta fix it</td>
<td>wait now i gotta fix it#</td>
</tr>
<tr>
<td>i got it</td>
<td>i got it#</td>
</tr>
<tr>
<td>i got you</td>
<td>i got you#</td>
</tr>
<tr>
<td>i got you</td>
<td>i got you#</td>
</tr>
<tr>
<td>i got your hand</td>
<td>i got your hand#</td>
</tr>
<tr>
<td>i got you</td>
<td>i got you#</td>
</tr>
<tr>
<td>i got you lemme open the door</td>
<td>i got you lemme open the door#</td>
</tr>
<tr>
<td>it’s got an a0lb0lc0l</td>
<td>it’s got an a0l b0l c0l#</td>
</tr>
</tbody>
</table>

Therefore, the key to solving this word-clumping problem is to find a principled way
to protect the other words from being corrupted by erroneous recognition of a bad word.
We say “principled way” because a cognitively sound strategy of word segmentation
when there are unknown words. In our learning approach we assume the same DLG
optimisation based approach to word segmentation as to lexical learning.

An ideal strategy is one that can incorporate the advantages of a DLG-based ap-
proach but avoid its disadvantages in dealing with the negative DLG problem. It is
highly possible that word segmentation by human subjects with an existing lexicon is

---

6In the original orthographic text of Bernstein corpus, a0l, b0l and c0l denote single letters A, B
and C, respectively.
Table 7.13: Word type precision and recall of the unsupervised lexical algorithms

<table>
<thead>
<tr>
<th>Learning Algorithms</th>
<th>Learned Word Types</th>
<th>Correct Word Types</th>
<th>Word Type Precision (%)</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>1,922</td>
<td>394</td>
<td>20.50</td>
<td>22.25</td>
</tr>
<tr>
<td>OS + LR+WS</td>
<td>1,046</td>
<td>400</td>
<td>38.34</td>
<td>22.59</td>
</tr>
<tr>
<td>OS + (LR+WS)x2</td>
<td>909</td>
<td>390</td>
<td>42.90</td>
<td>22.02</td>
</tr>
<tr>
<td>OS+V</td>
<td>3,036</td>
<td>601</td>
<td>19.64</td>
<td>33.94</td>
</tr>
<tr>
<td>OS+V+ LR+WS</td>
<td>1,914</td>
<td>568</td>
<td>29.68</td>
<td>32.07</td>
</tr>
<tr>
<td>OS+V+(LR+WS)x2</td>
<td>1,792</td>
<td>557</td>
<td>31.08</td>
<td>31.43</td>
</tr>
<tr>
<td>OS+V’</td>
<td>2,916</td>
<td>604</td>
<td>20.71</td>
<td>34.11</td>
</tr>
<tr>
<td>OS+V’+ LR+WS</td>
<td>1,582</td>
<td>565</td>
<td>35.71</td>
<td>31.90</td>
</tr>
<tr>
<td>OS+V’+(LR+WS)x2</td>
<td>1,492</td>
<td>558</td>
<td>37.42</td>
<td>31.49</td>
</tr>
</tbody>
</table>

an optimisation process with a different goodness measure than the one for their lexical learning. It can be a strategy as simple as maximal match segmentation (MMS): scanning through the input, outputting the longest matched word and then moving on to the next word, and continuing to work this way until the entire input utterance is finished. It is also possible that the MMS strategy is incorporated into the DLG optimisation based segmentation. One possible approach to the incorporation would be, whenever a negative DLG segmentation is encountered, to apply the MMS to save as many good words as possible from the clumping effect with “bad” word(s). According to our statistics, 5.76% of the learning output is in such word clumps. That means, if we could have a strategy that can perform word segmentation on this portion as well as on the rest, our learning algorithms would enhance their learning performance by $5.76\% \times \frac{70\%}{1-5.76\%} = 4.28\%$. This would be a very significant improvement.

The problem of negative DLG segmentation deserves much research effort in our future work.

### 7.5.2 Word Type Precision and Recall

In contrast to the word token precision and recall, which are usually at the level of 70%, the word type precision and recall of our learning algorithms appear low, at the level of slightly higher than 30%, as listed in Table 7.13. The number of word types in the original input corpus is 1771.

As a matter of fact, the word type precision and recall in unsupervised lexical learning are commonly at this level, or even lower. There has been no report on them, except
Figure 7.9: Word coverage rate versus frequency and coverage rank of word

for the word type precision of the MBDP-1 algorithm reported in [32]. MBDP-1’s word type precision starts at about 36% and grows to 54% in its incremental learning on the Bernstein corpus. All other algorithms reported in [32] have a word type precision beneath 30% on average.

From Table 7.13 we can see that the OS+V’ based algorithms have the best performance, and that when the LR+WS is applied one more time, the precision increases significantly and the corresponding recall decreases slightly. Both our OS+V and OS+V’ based algorithms learn about 1/3 of the word types in the input corpus.

How can such a low word type recall enable the word token precision and recall, as we reported above, at the level of 70%? The answer is given in Figure 7.9: the top 500 word types, either in the frequency or coverage ranking, cover about 90% of the input corpus. Our learning algorithms, e.g., OS+V+LR+WS and OS+V’ +LR+WS, learn about 550 words correctly, most of which are frequent words. It is not surprising that these words cover more than 70% of the input corpus.

Like other statistically based learning algorithms, our algorithms make fewer errors
Figure 7.10: Word type precision and recall in terms of frequency rank of word
in learning high frequency words than in learning low frequency words. The word type
precision and recall of our lexical learning versus word frequency rank are presented in
the two figures in Figure 7.10. The first figure is plotted in terms of the word frequency
in the learning output and the lower in terms of the frequency in the input corpus. The
diamond-dots plot the precision or recall at each frequency rank, and the solid lines
plot the average precision or recall over word types up to a frequency rank. We can see
from the upper figure that the word type precision over frequent words is very high: the
average precision up to the first 100 ranks (out of 147) is above 80%, and the learned
words are not 100% correct only in 12 ranks out of the first half, roughly 75, of all
ranks. We can also see from the lower figure that the word type recall over frequent
words is also very high: the average recall up to the first 100 ranks (out of 147) is above
80%, and there are only 16 ranks in the first half, roughly 75, of all ranks in which the
words are not 100% correctly learned. The low overall word type precision and recall is
determined by the fact that the word type number increases dramatically in the last 10
frequency ranks, where the precision and recall both drop very rapidly.

So, the focus of enhancing the word type precision and recall is on the enhancement
of the precision and recall of learning low frequency words.

7.5.3 Other Problems

In addition to the negative DLG segmentation problem and the problem of low word type
precision and recall, there are also other problems that hinder our learning algorithms
from performing or scoring any better. Some of these problems are inherent in the input
data, e.g., the inconsistency and data noise in the transcripts. Some are related to our
evaluation criteria, e.g., [−ing] and [−ed] are not counted as creditable morphemes in
the learning output, because they are another type of morpheme categorically different
from abbreviated forms of existing words.

Some problems are directly related to the behaviour of the DLG-based leaner, e.g.,
in the Bernstein corpus, the word [balloon], with 46 occurrences (a very high fre-
quency), is correctly recognised 42 times but erroneously divided into [bald][lo][on]
3 times, and a more frequent word [another], with 86 occurrences, is always divided
into [a][mother]7. Many frequent words or noun compounds, e.g., [instead] and
[golfball], in some other child-directed corpora, are also given abnormal segmentations
by the DLG-based learning algorithms. The word [instead], with 25 occurrences,

7Native speakers actually say “that’s a whole mother thing’.
is always segmented into [i][instead], and the compound [golfball], with 19 occurrences, is divided into [golf][ball] 8 times and recognised as a single word 11 times.

We still do not quite understand such unusual behaviour in a DLG-based lexical learner, because according to the DLG optimisation, the learner should choose [another] and [instead] instead of [a][mother] and [i][instead], because [another] and [instead] have the same frequency as [mother] and [instead], respectively, but each has a greater length and therefore a greater (positive) DLG. A lexical learner based on the DLG optimisation should select these longer words instead of the shorter ones. Why does it learn these unexpected words?

These are interesting problems that deserve more research effort. We have a reasonable assumption, namely, the least effort principle, as the starting point for our study. And we have developed an elegant computational theory for the unsupervised lexical learning based on the MDL principle and, accordingly, formulated the DLG goodness measure for selecting word candidates. We also have implemented a number of sound learning programs based on DLG optimisation that can learn most words correctly from the input corpus. However, a DLG lexical learner still has some unexpected behaviours beyond our understanding for the time being. We need to have a thorough understanding of them in order to advance the computational studies on human cognitive mechanisms for lexical learning based on our current achievements with the LDG optimisation approach.

7.6 Summary

In this chapter we have presented the unsupervised lexical learning algorithms based on the DLG optimisation over input utterances with a Viterbi algorithm, following the MDL principle. The representation formalism for the learning is trivially simple: each lexical item is represented as a string, with one parameter, namely, its frequency – each lexical item’s LDG is calculated in terms of its frequency. The Viterbi algorithm searches for the segmentation of an utterance that gives the greatest sum of DLG over its segments.

The lexical learning process in our computational approach consists of three phases (or learning modules), namely, DLG-based optimal segmentation of input utterances into lexical candidates, lexical refinement to divide the word-clump candidates into individual words, and then word segmentation in terms of lexical items acquired in the previous two phases. This lexical learning model is consistent with human infants’ behaviours in
CHAPTER 7. LEARNING ALGORITHMS AND EVALUATION

lexical learning: they recognise many word clumps as individual lexical items and later divide them into individual words when they are exposed to more language evidence supporting the decomposability of the clumps. In our approach each of three phases involves an application of the Viterbi algorithm for DLG optimisation with a different word space.

We have developed twelve unsupervised lexical learning algorithms each with a different combination of learning modules, parameters and constraints, and also developed a comprehensive evaluation approach based on seven evaluation measures to systematically examine their learning performance on the orthographic texts of the Bernstein child-directed speech corpus. The evaluation measures are word precision and recall, word boundary precision and recall, word type precision and recall, and correct character ratio. This is the most comprehensive evaluation approach ever applied in the field of computational lexical learning.

The top performance of our DLG-based unsupervised learning of words and of words and bound morphemes is achieved, respectively, by two typical unsupervised lexical learning algorithms involving the three phases, namely, the OS+V+LR+WS and OS+V’+LR+WS. The best performance in learning words is 75% precision, 71% recall and, accordingly, $F = 73\%$. This performance compares favourably with the state-of-the-art performance at 71% precision, 72% recall and $F = 71.5\%$, achieved by Brent’s MBDP-1 algorithm on the same child-directed speech corpus. Our best performance in learning words and bound morphemes is 80% precision, 76% recall and $F = 78\%$ -- this $F$ score is 9% higher than MBDP-1’s $F$ score on words.

In addition to the comprehensive evaluation, we have also analysed a number of problems that the DLG-based lexical learning approach encounters, including the negative DLG segmentation problem and the low precision and recall on word type. The analysis points to a direction for future work.
Chapter 8

Conclusions

This thesis has presented a computational approach to unsupervised lexical learning, including its assumptions, theoretical framework, goodness measure, learning algorithms and the implementation of the algorithms. In this last chapter we will conclude the thesis with respect to what we have achieved, and outline some future work.

The starting point of our study is the assumption that language learning follows the least effort principle that has been observed in many aspects of language phenomena. We hypothesise that in lexical learning a learner follows this principle to infer a minimal-cost representation for the input data, and the chunks in such representation are consequently the lexical items in the language. In our computational approach, we assume that the cost is measured in terms of bits in the representation.

With respect to these hypotheses we have developed an elegant theoretical framework for unsupervised lexical learning based on Solomonoff’s inductive inference theory and Rissanen’s MDL principle. We formulated the lexical learning problem as an inductive inference process to infer as many regularities as possible from the input data, and the goodness of a piece of regularity is measured by how much it can compress the input data. Within this learning-via-compression approach, learning lexical items is to infer regularities that can be represented with some pre-defined lexical representation formalism.

For the purpose of implementing a learning algorithm with this learning-via-compression approach, we formulated the DLG goodness measure for how good a string in the input is to be selected as a lexical candidate, in terms of the number of bits that the extraction of the string from the input as a lexical item can compress the data. We also have developed the Virtual Corpus system for deriving and counting n-grams in large-scale corpora, as a supporting tool to enable an efficient calculation of the DLG value for any
string in the input.

Based on the DLG calculation, we formulated a Viterbi algorithm for optimal segmentation of an input utterance into segments as lexical candidates that has a maximal sum of DLG over the segments. The essence of the Viterbi algorithm with the DLG goodness measure is to allow competition among regularities and let the best combination of regularities win. Based on this optimal segmentation, we developed a number of unsupervised lexical learning algorithms that comprise three learning phases that a human learner is known to undergo: inference of lexical candidates from input utterances, lexical refinement to derive finer-grained lexical items from the candidates, and word segmentation using the refined lexicon.

We also have developed a systematic evaluation approach for lexical learning, with a number of evaluation measures, including word precision and recall, word boundary precision and recall, word type precision and recall, and the correct character ratio. These measures provide the most comprehensive evaluation of lexical learning output that we ever have had in the field.

Experimental results on child-directed speech data from the CHILDES collection showed that validity of the learning approach and the effectiveness of the learning strategy implemented in our learning algorithms. We have learning algorithms that outperform the state-of-the-art learning algorithm in the field, with respect to the scores in the performance evaluation.

Comparing to other approaches to unsupervised lexical learning, our approach is outstanding not only in its learning performance but also in its simplicity. Our approach is simple in both the representation it uses and the computation involved. The representation formalism is only a deterministic regular grammar, and the computation involves mainly n-gram counting and DLG calculation based on the result of n-gram counting.

Our approach has illustrated that string counting enables lexical learning, in the sense that the idea of learning-via-compression can be implemented based on string (i.e., n-gram) counting: the string counting enables DLG calculation and the DLG calculation enables learning via compression. In this sense, we demonstrated a learning-via-counting approach to lexical learning, although this does not necessarily mean that the learning solely relies on the counting. Rather, it reveals the importance of frequency information in language learning, although linguistic structures are not necessarily directly determined by the frequency information. It is more likely that some more comprehensive
CHAPTER 8. CONCLUSIONS

goodness measure, e.g., the DLG measure, based on the frequency information may play the determining role in inferring linguistic structures.

The DLG goodness measure for lexical learning is one of the greatest achievements in our study. The Viterbi algorithm that we adopted as a dynamic programming technique for the implementation of our learning approach does not have any significant advantages over the algorithms adopted in many other researchers’ work. The superiority of our learning algorithms has to be explained by the advantage of the DLG goodness measure that the Viterbi algorithm follows for the optimal segmentation of an input utterance.

The three-phase learning model behind our learning algorithms is also a crucial factor that underlies the outstanding performance of our learning approach and also verifies its cognitive significance. In particular, the lexical refinement process outperforms the EM algorithm to an amazing extent, indicating that our learning model is close in principle to human cognitive mechanisms for lexical learning.

In summary, we have developed a start-of-the-art computational approach to unsupervised lexical learning. Its underlying principle is the principle of simplicity. Its outstanding learning performance has confirmed the validity of the learning theory, the soundness of the formulation of the learning approach and the effectiveness of the learning strategy implemented in the learning algorithm.

8.1 Directions of Future Work

There are many directions in which one could pursue future work on the basis of what we have achieved. One direction is concerned with the elaboration of the learning approach aimed at improving learning performance. As discussed in last chapter, the first problem to be solved in this direction is the negative segmentation problem. Another problem is the low word type precision and recall problem. This problem is caused by the low precision and recall on low frequency words. According to our analysis, the high and mediate frequency words have a precision and recall at the level of 80%. Can this advantage in our learning approach be facilitated to alleviate the problem of low word type precision and recall?

Another direction of future work is concerned with the extension of the learning approach. The current learning approach that we are undertaking is different from the online incremental learning that human learner actually perform in their language acquisition. We have had an elegant learning theory and a sound goodness measure. Based on these, we can implement an online incremental learning algorithm, to examine fur-
ther the validity and significance of our learning approach in relation to human learning. Also, the psychological reality of DLG measure or some equivalent measure in lexical learning, in contrast to transitional probability, is also an interesting issue to explore.

The next direction of future work is concerned with the application of our learning approach to some practical language processing problems. We took a learning-via-compression approach. This approach is closely related to dictionary-based compression methods and is highly suitable for text compression, mostly due to the demand for speed. Text compression has never implemented any compression algorithm involving optimisation on compression effect. In our approach we consider learning as an optimisation problem, and we have developed an efficient algorithm for DLG optimisation to achieve the maximal compression effect to find the best segmentation for the input. If we incorporate the DLG optimisation into text compression, what performance can we achieve?

Other possible extensions and applications of our learning approach are related to practical NLP tasks, including phrase learning from POS tagged corpora, chunking as a subtask for sentence analysis, word segmentation, proper name and unknown word identification, grammar induction, etc. These extensions will make our learning approach have practical significance for NLP technology development.
Bibliography


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


[193] C. Kit. Speeding up the Virtual Corpus approach to deriving and retrieving n-grams for any n from large-scale corpora. Manuscript, Department of Computer Science, University of Sheffield, August 1995.


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


