Information Preservation and Its Applications to Natural Language Processing

Ph.D. Dissertation

Ruey-Cheng Chen

Dept. of Computer Science and Information Engineering
National Taiwan University
1 Roosevelt Rd. Sec. 4, Taipei 106, Taiwan

rueycheng@turing.csie.ntu.edu.tw

Draft (November 13, 2012; Revised: April 26, 2013)

Acknowledgments

I wish to thank my advisor Jieh Hsiang, for his help and support over the years, and being my favorite academic role model.

I have been fortunate to work with and learn from many brilliant people, including Chun-Yi Chi, Wei-Yen Day, Andrew S. Gordon, Shao-Hang Kao, Chia-Jung Lee, Shuo-Peng Liao, Yi-Chun Lin, Wen-Hsiang Lu, Reid Swanson, Chiung-Min Tsai, and Jenq-Haur Wang. I owe my deepest gratitude to Hsin-Hsi Chen, Pu-Jen Cheng, Lee-Feng Chien, Jane Yung-Jen Hsu, and Chih-Jen Lin, for their great service in my dissertation and proposal committee.

The credit is shared with my family and in-laws. I am especially indebted to my wife, Hsing-Hui, and my son, Karl. Without their patience and insistence, this dissertation would not have been possible.

Abstract

In this dissertation, we motivate a mathematical concept, called information preservation, in the context of probabilistic modeling. Our approach provides a common ground for relating various optimization principles, such as maximum and minimum entropy methods. In this framework, we make explicit an assumption that the model induction is a directed process toward some reference hypothesis. To verify this theory, we conducted extensive empirical studies to unsupervised word segmentation and static index pruning. In unsupervised word segmentation, our approach has significantly boosted the segmentation accuracy of an ordinary compression-based method and achieved comparable performance to several state-of-the-art methods in terms of efficiency and effectiveness. For static index pruning, the proposed information-based measure has achieved state-of-the-art performance, and it has done so more efficiently than the other methods. Our approach to model induction has also led to new discovery, such as a new regularization method for cluster analysis. We expect that this deepened understanding about the induction principles may produce new methodologies towards probabilistic modeling, and eventually lead to breakthrough in natural language processing.

Keywords: information theory; information preservation; induction principle; unsupervised word segmentation; static index pruning; entropy optimization.

摘要

在這篇論文中，我們從機率模型的範疇內推導一個稱作「資訊保存」的數學概念。我們的方法提供了連接數個最佳化原則，例如最大熵及最小熵方法（maximum and minimum entropy methods）的基礎。在這個框架中，我們明確地假設模型推衍是一個目標針對某個參考假說的有向過程。為了檢驗這個理論，我們對無監督式斷詞（unsupervised word segmentation）以及靜態索引刪減（static index pruning）進行了詳盡的實證研究。在無監督式斷詞中，我們的方法顯著地提昇了以壓縮為基礎的方法斷詞精確度，並且在效能與效率表現上達到與目前最佳方法接近的程度。在靜態索引刪減上，我們提出的以資訊為基礎的量度（information-based measure）以比其他方法效率更好的方式達到目前最好的結果。我們的模型推衍方法也取得了新發現，像是分群分析（cluster analysis）中的新校正方法。我們期望這個對推衍原則的深度理解能產生機率模型的新方法論，並且最終邁向自然語言處理上的突破。

關鍵詞： 資訊理論；資訊保存；推衍原則；無監督式斷詞；靜態索引刪減；熵最佳化。

Contents

Acknowledgments **i**

Abstract **iii**

Abstract (in Chinese) **v**

List of Figures **xi**

List of Tables **xiii**

1 Introduction **1**

1.1 Goals and Methodology 1

1.2 Contribution 2

1.3 Outline 3

2 Information Preservation **5**

2.1 Overview 5

2.2 Formal Definition 6

2.3 Relation to Other Approaches 8

2.3.1 Principle of Maximum Entropy 8

2.3.2 Principle of Minimum Cross-Entropy 8

2.3.3 Minimum-Entropy Methods 9

2.4 Examples 10

2.4.1 Feature Selection 10

2.4.2 Regression 11

2.4.3 Cluster Analysis . . . 14
2.5 Concluding Remarks . . . 18

3 Unsupervised Word Segmentation 21
3.1 Overview . . . 21
3.2 Related Work . . . 23
3.2.1 Transitional Probability . . . 23
3.2.2 Mutual Information . . . 24
3.2.3 Hierarchical Bayesian Methods . . . 25
3.2.4 Minimum Description Length Principle . . . 26
3.3 Preliminaries . . . 27
3.4 Regularized Compression . . . 28
3.5 Iterative Approximation . . . 31
3.6 Implementation . . . 32
3.7 Evaluation . . . 34
3.7.1 Setup . . . 34
3.7.2 Parameter Estimation . . . 35
3.7.3 Evaluation on Bernstein-Ratner Corpus . . . 36
3.7.4 Evaluation on Bakeoff-2005 Corpus . . . 37
3.7.5 Effects of Data Size . . . 40
3.8 Concluding Remarks . . . 42

4 Static Index Pruning 45
4.1 Overview . . . 45
4.2 Related Work . . . 47
4.2.1 Term-Centric and Document-Centric Methods . . . 47
4.2.2 Recent Development . . . 48
4.3 Information Preservation . . . 48
4.4 Compositional Approximation . . . 52
4.5 Experiments . . . 54

4.5.1 Setup 54
4.5.2 Retrieval Performance 55
4.5.3 Correlation Analysis 59
4.6 Concluding Remarks 60

5 Discussion and Conclusion 61

5.1 Overview 61
5.2 General Discussion 63
5.2.1 Problem-Oriented Analysis 63
5.2.2 Relation to Principle of Maximum Cross-Entropy 64
5.2.3 Implication to Latent Variable Modeling 65
5.3 Concluding Remarks 65

Bibliography 69

A Supplementary Details 77

A.1 Representation System and Entropy 77
A.2 Rewrite-Update Procedure 79
A.3 Utility-Bias Tradeoff 84
A.3.1 Motivation 84
A.3.2 Perturbation and Utility-Bias Tradeoff 86
A.3.3 Applications 88
A.3.4 Example: Unsupervised Word Segmentation 89
A.3.5 Example: Static Index Pruning 93

List of Figures

2.1 A practical example of linear regression. 12

2.2 Cluster analysis is a way to generalize data into groups. 15

3.1 Performance evaluation for the proposed method on the CityU training set. 41

3.2 Performance evaluation for the proposed method on the MSR training set. 42

4.1 Performance results for all the methods on WT2G. Rows indicate different performance measures (MAP/P@10). Columns indicate different query types (short/long). 55

List of Tables

3.1 The notation used in the development of regularized compression. . . 27

3.2 Performance evaluation for the proposed method across different test corpora. The first row indicates a reference HDP run (Goldwater et al., 2009); the other rows represent the proposed method tested on different test corpora. Columns indicates performance metrics, which correspond to precision, recall, and F-measure at word (P/R/F), boundary (BP/BR/BF), and type (TP/TR/TF) levels. 36

3.3 Performance evaluation on the Bernstein-Ratner corpus. The reported values for each method indicate word precision, recall, F-measure and running time, respectively. The underlined value in each column indicates the top performer under the corresponding metric. . 37

3.4 A short summary about the subsets in the Bakeoff-2005 dataset. The size of each subset is given in number of words (W) and number of unique word types (T). 38

3.5 Performance evaluation on the common training subsets in the Bakeoff-2005 and Bakeoff-2006 datasets. The reported values are token F-measure. The boldface value in each column indicates the top performer for the corresponding set. 39

3.6 Performance evaluation on two random samples from the common sets (CityU and MSR subsets) in the Bakeoff-2005 and Bakeoff-2006 datasets. Note that the third run is an outside test. 40

4.1 The overall performance results on LATimes. We round down all the reported measures to the 3rd digit under the decimal point, and ignore preceding zeroes and decimal points for brevity. Underlined entries indicate the best performance in the corresponding group. Entries that are significantly superior or inferior ($p < 0.05$) to TCP are denoted by superscripts ▲ or ▼, respectively. Analogously, entries that are significantly superior or inferior to PRP are denoted by subscripts $\sharp$ or $\flat$, respectively. 56

4.2 The overall performance results on TREC-8. See Table 4.1 for the description of notation. 57

4.3 The overall performance results on WT2G. See Table 4.1 for the description of notation. 58

4.4 Correlation analysis for the decision measures on the LATimes corpus. The correlation is estimated using Pearson's product-moment correlation coefficient, weighted using term frequencies of index entries. 59

5.1 A brief summary of problems that are solvable by using information preservation. Each row indicates one research problem. The second column indicates the reference hypothesis used in the corresponding information preservation framework. The third column shows relative degree of entropy (low/high) for each problem. Related principles are listed at the last column; abbreviations are used instead of full names. The principles include minimum entropy (MinEnt), maximum entropy (MaxEnt), minimum error (MinErr), and maximum likelihood (MaxLL). 63

Chapter 1

Introduction

1.1 Goals and Methodology

The main theme of this thesis work is a mathematical concept called information preservation. This concept is introduced in the context of probabilistic modeling, as an optimization principle in fitting probabilistic models. It has a very broad application domain and has been shown to be useful in practical settings. Our primary goal here, if not the only, is to convince the readers that this particular theory is interesting and useful so as to maximize the impact.

This is never an easy task, and sitting in a crowded ground along with renowned mathematical ideas, such as maximum entropy and maximum likelihood, has made it even more challenging. Specifically, the intricacy embodied in information preservation adds much complexity to this writing. As its name suggests, the principle is about preserving information carried by fitted probabilistic models, measured by absolute change in entropy. This concept, although being straightforward, is not easy to introduce without careful postulation.

To make it easy to comprehend the underlying concept, we seek to incorporate many practical examples as we give out formal arguments. This application-oriented strategy is applied throughout the work. First, three classic problems in probabilistic modeling is treated in the first part of the thesis, showing that information preservation leads to a unified, simple way of looking at model fitting problems. Then, two natural language processing applications are introduced as proof of concepts. We show that information preservation can be used to solve complicated cases where other mathematical principles do not seem to fit, such as inducing a vocabulary model in unsupervised word segmentation, or reducing a predictive model in static index pruning. Finally, we conclude this work by discussing some possible extensions of information preservation.

1.2 Contribution

The concept of information preservation provides a common ground for maximum and minimum entropy methods. In this unified framework, we highlight that the usual principles applied in model fitting, such as maximization and minimization of entropy, are not arbitrary decisions. An implicit reference hypothesis is there to guide the optimization process, and model fitting seeks to approach that hypothesis in terms of model complexity in a constrained space. Existence of such a reference model is a strong indication on how the search for fitted model shall be conducted. In this respect, our account partly justifies the incompatibility between entropy maximization and minimization. From the view of information preservation, these principles no longer contradict each other; they are just different use cases that find application in different scenarios.

Our study also covers two natural language processing applications, which are unsupervised word segmentation and static index pruning. Our unique view to unsupervised word segmentation is a renovation of conventional compression-based methodology. We introduce the idea of preservation of vocabulary complexity into an ordinary compression-based framework. The renovated approach, regularized compression, has significantly boosted the segmentation accuracy, and has achieved comparable performance as several state-of-the-art methods. Our approach can be easily retargeted to larger text corpora since it is more efficient and less resource-demanding. Our experimental study also reveals one interesting application of this method as to applying parallel segmentation to one large text collection. This application is not addressed in this thesis work, but will be a primary focus in our further exploration.

We have also obtained fruitful results in our experiments in static index pruning. Conventional methods usually solve this problem using impact-based decision criteria based on the idea that impact is directly related to retrieval performance. Our approach provides an alternative view to this problem. We show that information-based measures can achieve competitive performance in a more efficient way. From the experiments, we uncover the possibility of combining multiple decision criteria to prioritize the index entries, which is backed by low correlation found between the proposed approach and the other state-of-the-art methods. Although further exploration for this interesting idea is beyond the scope of this thesis, our result has led to deeper insights in this research problem and has thus paved the way for exciting future efforts.

1.3 Outline

Chapter 2 covers the definition and implications of information preservation. We first motivate this concept from a theoretical aspect, showing that it generalizes the ideas advocated by well-known mathematical principles such as maximum entropy and minimum-entropy methods. Then, we apply information preservation to some well-studied fundamental problems in probabilistic modeling, including feature selection, regression, and cluster analysis. Through these working examples, we show that our approach leads to simple, intuitive solutions in accordance with the known results.

In Chapter 3, we give an overview on the development of unsupervised word segmentation. This is the first application example of information preservation studied in this thesis work. Our approach to unsupervised word segmentation is motivated from a different angle, based on the idea of text compression. We first assume that a solution to unsupervised word segmentation is equivalent to a series of choices made in text compression. These choices can be determined by solving a general optimization problem that corresponds to information preservation. This problem is intractable due to various combinatorial choices involved, so we present a simplified iterative solution as an approximation. Later, we describe a full experimental setup that covers experiments on two standard benchmarks. The experimental result shows that the performance of the our approach is comparable to that of state-of-the-art unsupervised word segmentation methods in terms of F-measure.

We introduce the problem of static index pruning in Chapter 4 as the second application of information preservation. We first briefly go over the past effort in this area. Then we proceed to formulate this problem in the information preservation framework, in which static index pruning is seen as a process of degenerating a predictive model. Under this rationale, the primary goal in static index pruning is to preserve the predictive power of the pruned index. This idea is written out as an information preservation problem, and is solved by using an approximation algorithm. This construction leads to an efficient formula that we can use to rank postings. We test our approach in three standard collections of different size, and compare its performance with two other state-of-the-art methods. The result shows that our approach is competitive to the best performance across all experimental settings.

Some incomplete results are loosely discussed in Chapter 5. We briefly review the problem types that are solvable by information preservation and discuss the advantages of our approach. Finally, in Section 3.8, we summarize the contributions and possible implications of this work to conclude this thesis.

Chapter 2

Information Preservation

2.1 Overview

Much of the recent development in natural language processing is devoted to probabilistic modeling. Nowadays, probabilistic methods have prevailed in almost every subfield in natural language processing. Some notable examples include probabilistic context-free grammar (PCFG) in parsing, max-margin methods in classification, mixture models in clustering, to name a few.

What lies in the core of a probabilistic method is often an assumption, written in probabilistic statements, about certain aspect of language that one tries to model. The probabilistic statements discussed here can be *generative*, describing a generative process that produces the language we have observed, or *discriminative*, describing a decision model that helps differentiate one type of observation from the other.

Probabilistic methods are not merely about assumptions. One needs to find the right model, e.g., parameters, to fit these assumptions, and this is when optimization techniques come into play. Behind any optimization problem, there is always some *optimization principle*, based on which one specifies the objective in mathematical terms. Such a principle usually has a deep root in statistics and physics, and sometimes it can be very philosophical.

Most of the well-known principles carry very simple messages that generalize essential aspects in scientific modeling, which in turns lead to successful applications. Maximum likelihood (ML), for one, is used in almost every scientific area. There are many others, including maximum a posteriori (MAP), a Bayesian-remake of maximum likelihood; maximum entropy (ME), asserting that one shall seek the most general model; and minimum description length (MDL), advocating the famous philosophical concept of Occam's razor.

In this chapter, we present the concept of *information preservation* that we believe encapsulates some of the common wisdom in probabilistic modeling. As we will show later, information preservation covers two well-known principles, maximum entropy and minimum entropy, as its special cases. It also provides a ground for discussions about how information-based optimization can be designed in probabilistic modeling.

2.2 Formal Definition

We define an information preservation problem as follows. Let Θ be the hypothesis space, and $\Theta_C \subset \Theta$ be a subset of hypotheses that satisfy some constraint C. Let X be some random variable and H be some entropy-like information quantity which is well-defined defined for every $\theta \in \Theta$.

Given that $\theta_0 \in \Theta$, the following minimization problem is said to be an *information preservation problem*:

$$\begin{array}{ll} \text{minimize} & |H_\theta(X) - H_{\theta_0}(X)| \\ \text{subject to} & \theta \in \Theta_C. \end{array} \tag{2.1}$$

The general idea is very simple. Given the search space Θ_C and a reference hypothesis θ_0, the best hypothesis is the one that minimizes the absolute change in information, measured by some quantity H. In other words, we want to find an approximation of the reference hypothesis θ_0 in the constrained search space Θ_C; the closeness (or distance) between two hypotheses is measured in terms of the change in entropy.

Information preservation comprises three key aspects, *information*, *hypothesis*, and *approximation*, which are detailed in the following paragraphs.

Information The information of a probabilistic model is measured by the quantity H. The definition of H can be further generalized as long as the extension is measurable for every hypothesis $\theta \in \Theta$. This function can be entropy $H(X)$, joint entropy $H(X, Y, Z)$, or even conditional entropy $H(X|Y)$. To write a problem in the form of information preservation, we need to think about by which quantity the information is best represented.

Hypothesis In an information preservation problem, we search for a hypothesis that is closest to the *reference hypothesis* θ_0 in terms of the information estimate H. Since this is a strong assumption that biases the search process, the choice of reference hypothesis needs to be justified. A good starting point is to consider some trivial hypothesis, as we will shortly visit in the later examples, that can be easily formed based on the observations.

Approximation The optimization problem may not be analytically feasible to solve. As we will see later, information preservation can sometimes fall back to entropy maximization or minimization, none of which is easy to solve when combinatorial choices are involved. Therefore, it is essential to know whether the formulation can be solved approximately using efficient numerical methods.

In the next section, we discuss the relation of information preservation to the other principles.

2.3 Relation to Other Approaches

2.3.1 Principle of Maximum Entropy

Jaynes (1957a; 1957b) proposed the principle of maximum entropy, stating that a distribution that maximizes entropy is the one making the minimal claim beyond the knowledge embedded in the prior data. Therefore, the maximum-entropy choice in model fitting is equivalent to choosing the least informative distribution from an entire family of distributions.

This principle is usually cast as a constrained entropy maximization problem, which in simple continuous cases is solved by convex programming. Consider that $\Theta_C \subset \Theta$ is the set of feasible parameter combinations. The entropy maximization problem is written as follows.

$$\begin{aligned} &\text{maximize} \qquad && H_\theta(X) \\ &\text{subject to} \qquad && \theta \in \Theta_C. \end{aligned} \tag{2.2}$$

It is clear that, when we take θ_0 as the most uninformative model in Θ, the information preservation problem reduces to the maximum entropy problem.

2.3.2 Principle of Minimum Cross-Entropy

Kullback (1959), who also developed the Kullback-Leibler divergence, proposed the principle of minimum cross-entropy, which is sometimes called the principle of minimum discrimination information. It asserts that, as more data is observed, a new distribution shall be fitted and this choice needs to be made as close to the original distribution as possible, measured in terms of Kullback-Leibler divergence.

Let the original distribution be denoted as p and the new one as q. The Kullback-Leibler divergence between p and q (in that order) can be expressed in terms of the entropy of p and the cross entropy of p and q. In the following derivation, we consider only the discrete case. Note that we use $D(p||q)$ to denote the Kullback-

Leibler divergence and $CE(p;q)$ to denote the cross entropy.

$$\begin{aligned} D(p||q) &= \sum_x p(x) \log \frac{p(x)}{q(x)} \\ &= \sum_x p(x) \log p(x) - \sum_x p(x) \log q(x) \\ &= -H(p) + CE(p;q). \end{aligned} \tag{2.3}$$

Since p is a known distribution, minimizing $D(p||q)$ is equivalent to minimizing the cross entropy $CE(p;q)$. This objective is convex since it is a linear combination (or more precisely, convex combination) of the $-\log(x)$ function, which is also convex. Again, this is a convex programming problem. The principle can be written as the following optimization problem in our notation:

$$\begin{aligned} &\text{minimize} && CE(\theta_0;\theta) \\ &\text{subject to} && \theta \in \Theta_C. \end{aligned} \tag{2.4}$$

There is no direct connection between information preservation and the principle of minimum cross-entropy. Nevertheless, they agree on one essential point that closeness between the old and the new distributions shall be highly valued in search for a better fit for the new facts.

2.3.3 Minimum-Entropy Methods

Minimum entropy is not formally recognized as a mathematical principle, but it has already found many applications in combinatorics and machine learning. The general philosophy is to find the most informative model that satisfies the constraint, and is sometimes seen as incompatible with the principle of maximum entropy, which states exactly the opposite.

While the compatibility issue between two theories is beyond our work, we do find a few examples that shows how minimum entropy leads to other principles, such as error minimization.

When setting the reference hypothesis θ_0 at some extreme points, e.g., $H_\theta = 0$, information preservation reduces to minimum entropy. Nevertheless, optimizing toward such extremes may not always be a good justification by itself. In the next section, we will review some typical assumptions regarding setting such reference hypotheses.

2.4 Examples

2.4.1 Feature Selection

Feature selection is a technique commonly used in supervised learning tasks, such as text classification, where the feature dimension is usually very high. In this case, feature selection determines a subset of feature dimensions so that the learning task can be carried out more efficiently.

Let O be an set of labeled instances $\langle (x_1, c_1), (x_2, c_2), \ldots, (x_n, c_n) \rangle$ where each $x_i \in 2^d$ is a d-dimensional feature vector and each $c_i \in 1, \ldots, K$ is a class label. Here, for simplicity, we assume binary features and finite class labels.

Every subset of these features forms a unique partition over the instances. A subset of c binary features generates 2^c different possible combinations, and according to which one can sort the instances uniquely into different groups. That is to say, any two instances with the same combination of these features are assigned to the same group.

Assume that we attach a new variable $a_i \in \mathbb{N}$ (can sometimes be greater than K) to every instance (x_i, c_i). The variables $\{a_1, a_2, \ldots, a_n\}$ collectively represent some partition over the instances, determined by some selected set of features.

Consider the entropy of class labels, $H(C)$, and the conditional entropy of class labels given the partition assignment, $H(C|A)$. Here, C and A are random variables for class label and partition assignment, respectively. These two quantities have the following relation:

$$H(C) = H(C|A) + I(C;A), \tag{2.5}$$

where $I(C;A)$ is the mutual information between class labels and the partition assignment.

Now, let θ_0 denote an *ideal partition* over the instances, with which we assign $a_i = c_i$ for each instance. This partition always exists because we have knowledge about the true class labels. Given this ideal partition, the conditional entropy $H(C|A)$ becomes 0, and therefore, for any partition θ over the instances, we know that $H_{\theta_0}(C|A)$ is always less than or equal to $H_\theta(C|A)$.

We use θ_0 as the reference partition and write down the information preservation problem. Let θ^* be the optimal partition. We have the following equations:

$$\begin{aligned} \theta^* &= \arg\min_\theta |H_\theta(C|A) - H_{\theta_0}(C|A)| \\ &= \arg\min_\theta H_\theta(C|A) \\ &= \arg\min_\theta H_\theta(C) - I_\theta(C;A) \\ &= \arg\max_\theta I_\theta(C;A). \end{aligned} \tag{2.6}$$

The results accords with a commonly-used feature selection method that chooses the optimal feature set by maximizing the mutual information.

2.4.2 Regression

Consider a series of data points $O = \langle (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \rangle$ where each $(x_i, y_i) \in \mathbb{R}^2$. We assume a functional relation $y = f(x)$, between the two components and want to find the best fit f in a family of functions $\mathcal{F}$. This is a classic problem that *regression analysis* seeks to solve.

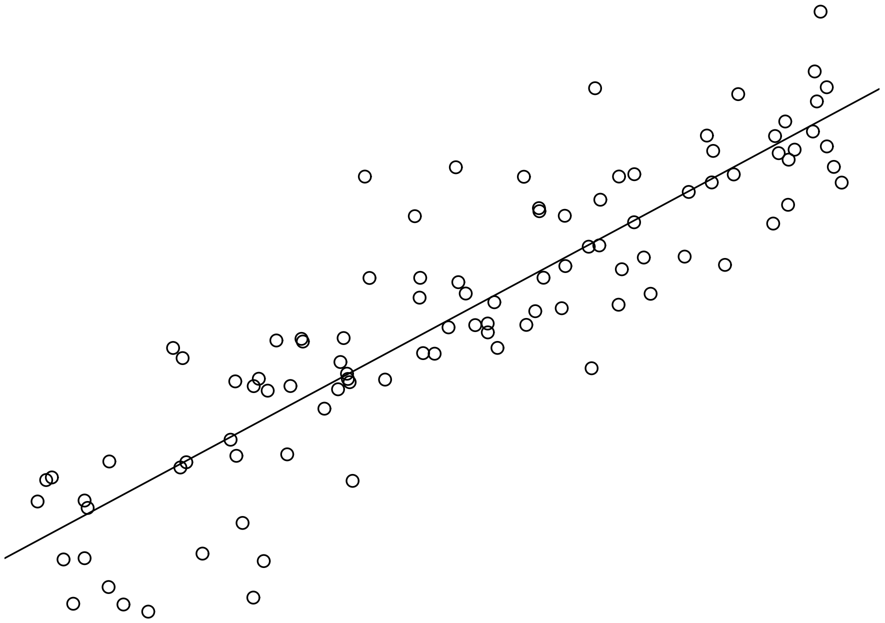

Figure 2.1: A practical example of linear regression.

Let us first write this problem in the typical way. We use random variables X and Y to denote some random data point that we observe. The structural dependence between X and Y is defined as follows. Note that, in this definition, ϵ is a random variable that denotes the error:

$$Y = f(X) + \epsilon,$$
$$\epsilon \sim \mathcal{N}(0, \sigma^2).$$

In regression analysis, the goodness of fit is assessed in some way according to the error made by the structural assumption. For this reason, we need to specify an error distribution. Here, we take an usual assumption that ϵ follows a Gaussian distribution with zero mean and σ^2 variance.

We estimate the entropy of ϵ empirically using the data points that we have observed, O, as samples. Specifically, this sample-based estimate is defined as:

$$\tilde{H}(\epsilon) = -\frac{1}{n} \sum_{i=1}^{n} \log p(\epsilon_i), \tag{2.7}$$

where $\epsilon_i = y_i - f(x_i)$ denotes the error term for the i-th data point. Plugging in the

Gaussian density into Equation (2.7), we arrive at the following equation:

$$\tilde{H}(\epsilon) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\epsilon_i^2}{2\sigma^2} + \frac{1}{2}\log 2\sigma^2\pi\right). \tag{2.8}$$

It can be easily shown that $\tilde{H}(\epsilon)$ is uniquely minimized when $\epsilon_i = 0$ for all $i = 1, \ldots, n$. In other words, when the search space $\mathcal{F}$ is unrestricted, any function that trivially maps the observed values $\{x_i\}$ to their counterparts $\{y_i\}$ minimizes the error entropy. To see why this is the case, we check the first and the second derivatives of $\tilde{H}(\epsilon)$ with respect to $f(x_i)$:

$$\frac{\partial}{\partial f(x_i)}\tilde{H}(\epsilon) = -\frac{1}{n\sigma^2}\epsilon_i, \tag{2.9}$$

$$\frac{\partial^2}{\partial f(x_i)^2}\tilde{H}(\epsilon) = \frac{1}{n\sigma^2}. \tag{2.10}$$

The first derivative vanishes when $\epsilon_i = 0$ for all $i = 1, \ldots, n$. Since the second derivative is always positive, the stationary point is a global minimum.

Now we are ready to apply information preservation to this problem. We chose some trivial function f_0 as the reference model, i.e., for all $i = 1, \ldots, n$, $f_0(x_i) = y_i$. Applying the concept of information preservation, the best model $f^* \in \mathcal{F}$ is the one that minimizes the absolute difference in entropy against the reference model, as in:

$$\begin{aligned} f^* &= \arg\min_{f\in\mathcal{F}} |\tilde{H}_f(\epsilon) - \tilde{H}_{f_0}(\epsilon)| \\ &= \arg\min_{f\in\mathcal{F}} \tilde{H}_f(\epsilon) - \tilde{H}_{f_0}(\epsilon) \\ &= \arg\min_{f\in\mathcal{F}} \tilde{H}_f(\epsilon) \\ &= \arg\min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\epsilon_i^2}{2\sigma^2} + \frac{1}{2}\log 2\sigma^2\pi\right) \\ &= \arg\min_{f\in\mathcal{F}} \sum_{i=1}^{n}\epsilon_i^2. \end{aligned} \tag{2.11}$$

This result is in line with the least squares method that minimizes the sum of squared errors of a fit.

2.4.3 Cluster Analysis

Consider that we have observed n data points $\langle x_1, x_2, \ldots, x_n \rangle$ in some d-dimensional space. We have a structural assumption that these data points belongs to some number of clusters. Our job in *cluster analysis* is to find these clusters and assign each data point to one.

For simplicity, we assume that the number of clusters, K, is known *a priori*. Our goal here is to test a number of hypotheses about the underlying structure, and each hypothesis is expressed as a set of assumed cluster centers, denoted as θ.

We use the notation X and $\mu(X)$ to denote a random data point and its cluster center, respectively. We assign to each X the closest cluster center in θ. In other words, we write $\mu(X)$ as:

$$\mu(X) = \arg\min_{\mu \in \theta} \|X - \mu\|_2,$$

Therefore, we have the following definition for the error distribution:

$$\begin{aligned} X &= \mu(X) + \epsilon, \\ \epsilon &\sim \mathcal{N}(0, \Sigma). \end{aligned} \tag{2.12}$$

Here, ϵ is the displacement between a data point and its cluster center, and it follows a multivariate Gaussian distribution centered at the origin 0, i.e., a zero vector, with covariance Σ, a d by d symmetric and positive-definite matrix.

The probability of observing some data point is written out somewhat differently because the support sets of density functions overlap with each other. In this defi-

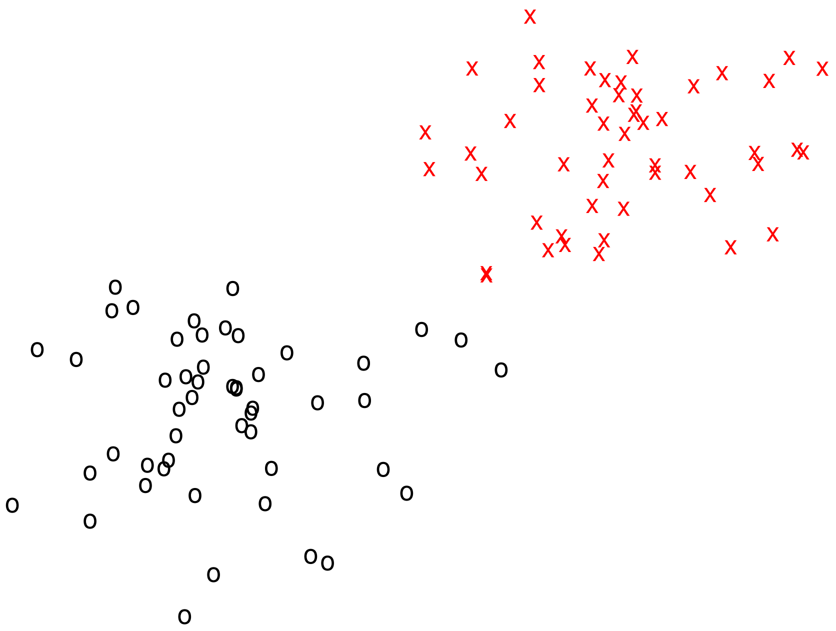

Figure 2.2: Cluster analysis is a way to generalize data into groups.

nition, we introduce a normalization factor $Z(\theta)$:

$$\begin{aligned} p(x|\theta) &= \frac{1}{Z(\theta)} \mathcal{N}(x; \mu(x), \Sigma), \\ Z(\theta) &= \int_{x'} \mathcal{N}(x'; \mu(x'), \Sigma) \; \mathrm{d}x'. \end{aligned} \tag{2.13}$$

Following the idea in the previous section, we estimate the entropy of $p(x|\theta)$ empirically using the sample-based method. The entropy is written as:

$$\begin{aligned} H(X|\theta) &= -\frac{1}{n} \sum_{i=1}^{n} \log p(x|\theta) \\ &= \frac{1}{2n} \sum_{i=1}^{n} (x_i - \mu(x_i))^T \Sigma^{-1} (x_i - \mu(x_i)) + \frac{1}{2} \log |2\pi\Sigma| + \log Z(\theta). \end{aligned} \tag{2.14}$$

This definition of error entropy has two implications. First, the sum of square errors, a penalty function commonly used in cluster analysis, is actually a special case of information preservation. This happens when the likelihood $p(x|\theta)$ is left unnormalized, i.e., setting $Z(\theta)$ to some constant. Second, solving the information preservation problem generally with respect to this equation leads to a regularized version of sum of squared errors.

Let us discuss the first case. When $Z(\theta)$ does not depend on θ, it can be shown that a trivial hypothesis θ_0 that assigns each data point to itself as the cluster center, i.e., $\mu(x_i) = x_i$ for all $i = 1, \ldots, n$, is the global minimizer of this entropy. To see how, let us suppose $Z(\theta) = c$ for some constant c and take the derivative tests with respect to μ:

$$\begin{aligned} H(X|\theta) &= \frac{1}{2n}\sum_{i=1}^{n}(x_i - \mu(x_i))^T\Sigma^{-1}(x_i - \mu(x_i)) + \frac{1}{2}\log|2\pi\Sigma| + \log c, && (2.15)\\ \frac{\partial}{\partial\mu}H(X|\theta) &= \frac{1}{n}\sum_{i:\mu(x_i)=\mu}\Sigma^{-1}(x_i - \mu), && (2.16)\\ \frac{\partial^2}{\partial\mu\partial\mu^T}H(X|\theta) &= \frac{1}{n}\Sigma^{-1}. && (2.17) \end{aligned}$$

We first establish the convexity of $H(X|\theta)$ by writing out its Hessian, which is positive definite. Since the first derivative vanishes at $\theta = \theta_0$, we confirm that θ_0 is a global minimum.

Applying the concept of information preservation, the best hypothesis θ^* is actually a minimizer of weighted sum of squared errors. This result accords with usual assumption in cluster analysis, and suggests that information preservation is a more general method in this aspect.

$$\begin{aligned} \theta^* &= \arg\min_\theta |H(X|\theta) - H(X|\theta_0)|,\\ &= \arg\min_\theta H(X|\theta) - H(X|\theta_0),\\ &= \arg\min_\theta H(X|\theta),\\ &= \arg\min_\theta \sum_{i=1}^{n}(x_i - \mu(x_i))^T\Sigma^{-1}(x_i - \mu(x_i)). \end{aligned} \tag{2.18}$$

Solving the information preservation problem analytically is difficult. This is a difficult problem because $Z(\theta)$ depends on the positions of all the cluster centers, and this entanglement cannot be easily analyzed. Therefore, as a necessary compromise, we suggest solving this problem in the context of medoid-based clustering, meaning that cluster centers have to sit on some data points rather than arbitrary choices. This leads to our second result in this example.

Generally, the exact value of $Z(\theta)$ can be written out as a summation of integrals. For any cluster center μ, let $N(\mu)$ denote the *neighborhood* of μ, which is a set of points that have μ as the cluster center, i.e., $\mu = \arg\min_{\mu'} \|x - \mu'\|_2$ for all $x \in N(\mu)$. With knowledge about $N(\mu)$, we can write $Z(\theta)$ as a summation of the densities contributed by individual cluster centers:

$$Z(\theta) = \sum_{\mu \in \theta} \int_{x \in N(\mu)} \mathcal{N}(x; \mu, \Sigma)\ \mathrm{d}x. \tag{2.19}$$

Solving this equation is still very challenging, since each integral has to be solved numerically with respect to $N(\mu)$, which is not easy to compute in high dimension. To approximate $N(\mu)$, we replace the domain of the innermost integral with a confidence region $C_r(\mu, \Sigma)$ of the Gaussian distribution $\mathcal{N}(\mu, \Sigma)$, i.e., the set of points that lies within some radius r to the mean μ:

$$\begin{aligned} Z(\theta) &= \sum_{\mu \in \theta} \int_{x \in C_r(\mu, \Sigma)} \mathcal{N}(x; \mu, \Sigma)\ \mathrm{d}x \\ &= \sum_{\mu \in \theta} \int_{x \in C_r(0, \Sigma)} \mathcal{N}(x; 0, \Sigma)\ \mathrm{d}x. \end{aligned} \tag{2.20}$$

We use the following equation to determine r for each μ. The idea is to take half of the distance from μ to its nearest neighboring cluster center.

$$r = \frac{1}{2} \min_{\mu' \in \theta} \|\mu - \mu'\|_2. \tag{2.21}$$

It is now feasible to solve the innermost integral in Equation (2.20) numerically.

Considering only medoids as cluster centers, we can compute this approximation to $Z(\theta)$ much more efficiently for any given hypothesis. Nevertheless, further simplification is needed to efficiently explore the search space. Here, we propose using an iterative algorithm to find the solution. This algorithm starts from the trivial hypothesis that has n cluster centers and iteratively removes one cluster center away. That is to say, it iteratively solve the following information preservation problem for $i = 1, \dots, n-K$.

$$\theta^{(0)} = \theta_0, \tag{2.22}$$

$$\theta^{(i)} = \arg\min_{\theta} |H(X|\theta) - H(X|\theta^{(i-1)})| \tag{2.23}$$

$$= \arg\min_{\theta} \left| \frac{1}{2n} \Delta_{\mathrm{WSSE}}(\theta, \theta^{(i-1)}) + \log \frac{Z(\theta)}{Z(\theta^{(i-1)})} \right|, \tag{2.24}$$

where $\Delta_{\mathrm{WSSE}}(\theta, \theta^{(i-1)})$ is the change in sum of squared errors,

$$\sum_{x:\mu^{(i-1)}(x) \neq \mu(x)} \left((x - \mu(x))^T \Sigma^{-1} (x - \mu(x)) - (x - \mu^{(i-1)}(x))^T \Sigma^{-1} (x - \mu^{(i-1)}(x)) \right).$$

The decision criteria in Equation (2.24) is actually a regularized version of weighted sum of squared errors. The term $\delta_{\mathrm{WSSE}}(\theta, \theta^{(i-1)})$ is always positive because removing cluster centers increases errors. The regularization term, $\log Z(\theta) - \log Z(\theta^{(i-1)})$, is negative due to the reduced total densities. Practically, we often want to drop the absolute value, because the regularization term is lower bounded by $-\log 2$.

This regularization method penalizes cases where the removed cluster center is closed to the others. In other words, it favors more spread-out clusters. This property also aligns with a conventional heuristic that suggests maximizing inter-cluster distances.

2.5 Concluding Remarks

In the previous sections, we have briefly reviewed several classic problems in probabilistic modeling. Conventional approaches to these problems, such as error minimization and maximization of mutual information, are shown to be equal to information preservation. Moreover, information preservation leads to a new regularization method to data clustering. This discussion, though not being thorough and rigorous, has suggested that information preservation is a more general and unified optimization strategy in probabilistic modeling.

So far, our exploration is limited to general modeling tasks. In the following chapters, we go on and uncover the potential of information preservation in a broader application domain, namely, natural language processing.

Chapter 3

Unsupervised Word Segmentation

3.1 Overview

Word segmentation is the computational task that aims at identifying word boundaries in a continuous text stream (Goldwater et al., 2009)., and it is essential to natural language processing because many NLP applications are designed to work on this level of abstraction. In language, words are the smallest lexical units that carry coherent semantics or meaning, That is to say, the meaning of individual words can be postulated or interpreted on their own. Therefore, for NLP applications that focus mostly on semantics, words are just the ideal representation to operate on.

In this chapter, we focus on unsupervised word segmentation, which aims to mitigate the word segmentation problem without using any training data. This technique has been vigorously studied by cognitive scientists and regional linguists, and has thus become an intersection of theory and application.

In cognitive science, unsupervised word segmentation is often related to lexical acquisition, a study regarding how infants acquire language in their early ages. It is generally believed that infants learn word boundaries largely without supervision, even though a number of visual or gestural cues have been shown useful in this process, so knowledge about practical methods is of great value to cognitivists in developing their theories.

Regional language groups also express high interest in unsupervised word segmentation, since in certain Asian languages, such as Japanese and Chinese, recognition of words is a non-trivial task. These notable exceptions do not practice the "whitespace separation" convention, which is a norm to many other languages such as English, and therefore words of an utterance in these languages are usually written out without separation, making further application even more difficult. Advanced supervised approaches using conditional random-fields have achieved promising result in many related regional task, but their adoption is generally limited by the costly and labor-intensive process of preparing training data. Unsupervised methods, on the contrary, do not have this burden and remain an economical alternative for most of the NLP projects.

In the following sections, we introduce a compression-based framework for unsupervised word segmentation, assuming that true words in a text can be uncovered via text compression. Here, we point out that efficiency, as usually assumed in data compression, is not the only factor to consider. Vocabulary complexity, which is the entropy rate of the resulting lexicon, needs to be controlled to some extent. As its complexity increases, more effort is required to harness the vocabulary. To leverage this issue, we suggest applying the principle of information preservation. We create a new formulation to word segmentation and write it out as an optimization problem, which is called *regularized compression*.

The rest of this chapter is structured as follows. We briefly summarize related work on unsupervised word segmentation in Section 3.2. In Sections 3.3 and 3.4, we introduce the proposed formulation. The iterative algorithm and other technical details for solving the optimization problem are covered in Sections 3.5 and 3.6.

In Section 3.7, we describe the evaluation procedure and discuss the experimental results. Finally, we present concluding remarks in Section 3.8.

3.2 Related Work

There is a rich body of literature in unsupervised word segmentation. Many early work focused on boundary detection, using cohesion measures to assess the association strength between tokens. Further development stressed on language generation, casting word segmentation as a model inference problem; recently, this approach has gained success when more sophisticated nonparametric Bayesian methods were introduced to model the intricate language generation process. The latest research trend is to use minimum description length principle to optimize the performance of existing methods. In the following subsections, we review a number of popular methods in this area and briefly discuss their strengths and weaknesses.

3.2.1 Transitional Probability

In linguistics, one of the earliest and the most influential idea for boundary detection is the *transitional probability.* This idea is uncovered in 1955 by Harris (Harris, 1955), who exploited the connection between token uncertainty and the presence of a boundary. In his seminal work, he stated that the uncertainty of tokens coming after a sequence helps determine whether a given position is at a boundary. Since then, many researchers has taken this idea and given their own formulations regarding the token uncertainly at a boundary.

One reasonable way to to model the token uncertainty is through *entropy.* Starting from late 1990s, many research efforts in Chinese text analysis have begun to follow this trail (Chang and Su, 1997; Huang and Powers, 2003). The definition of *branching entropy* that we know of today is given in 2006 by Jin and Tanaka-Ishii (Tanaka-Ishii, 2005; Jin and Ishii, 2006). Formally, the branching entropy of a word w is defined as:

$$H(X|X_n = w) = -\sum_{x \in \chi} P(x|w) \log P(x|w),$$

where χ is the set of all the possible characters. Note that this formulation is different from the conditional entropy of X given X_n.

There are other ways to model token uncertainty. In 2004, Feng et al. proposed the *accessor variety* that estimates the uncertainty based on the frequency rather than the probability (Feng et al., 2004). In his definition, we write accessor variety as:

$$\min\{\mathrm{AV}_L(w), \mathrm{AV}_R(w)\},$$

where the left/right accessor variety AV_L and AV_R is defined as the number of distinct tokens that precede/follow w, respectively.

3.2.2 Mutual Information

In 1990, Sproat and Shih discovered the use of mutual information in detecting two-character groups (i.e., bigrams) in the Chinese text (Sproat and Shih, 1990). Their work amplified the idea of using an association measure to assess how likely a string is indeed a word. This idea is straightforward enough: The segmentor first determines the most probable bigram in an input phrase by using mutual information, places two boundaries around the discovered bigram, and recursively process the remaining sub-phrases.

To date, there have been many association measures based on mutual information (Chien, 1997; Sun et al., 1998) introduced to the word segmentation problem.

3.2.3 Hierarchical Bayesian Methods

The past few years have seen many nonparametric Bayesian methods developed to model natural languages. Many such applications were applied to word segmentation and have collectively reshaped the entire research field. In cognitive science, unsupervised word segmentation is often related to language acquisition, an active area in which researchers explore how infants acquire spoken languages in an unsupervised fashion in their very early years. This connection had been brought to earth in 1950s in behavioral study.

The perspective that cognitivists take toward unsupervised word segmentation is usually a *generative* one, in which an underlying probabilistic structure is assumed for governing the generation of word sequences. To uncover the boundaries, one learns the latent model parameters from the text stream by applying inference techniques based on maximum-likelihood (ML) or maximum a-posteriori (MAP) principle, and then the most likely segmentation boundaries is discovered by using Viterbi-like algorithms.

Hierarchical Bayesian methods were first introduced to complement conventional probabilistic methods to facilitate context-aware word generation. Goldwater et al. (2006) used hierarchical Dirichlet processes (HDP) to induce contextual word models. Specifically, they expressed the contextual dependencies as:

$$\Pr(w_1 \ldots w_n \$) = P(w_1|\$) \left[\prod_{i=2}^{n} P(w_i|w_{i-1}) \right] P(\$|w_n).$$

The underlying generative process assumed by HDP is as follows. Note that P_0 is uniform across different word lengths.

$$\begin{aligned} G &\sim \mathrm{DP}(\alpha_0, P_0) \\ W_i|W_{i-1} = l &\sim \mathrm{DP}(\alpha_1, G) \quad \forall l \end{aligned}$$

This approach was a significant improvement over conventional probabilistic methods, and has inspired further explorations into more advanced hierarchical modeling techniques. Such examples include the nested Pitman-Yor process (Mochihashi et al., 2009), a sophisticated installment for hierarchical modeling at both word and character levels, and adaptor grammars (Johnson and Goldwater, 2009), a framework that generalize the idea behind HDP to probabilistic context-free grammars. The generative description of the nested Pitman-Yor Process (NPYP) is summarized as follows. Note that P_0 is generated from yet another character-level HPYP.

$$\begin{aligned} G &\sim \mathrm{PY}(d, \theta, P_0) \\ G_l &\sim \mathrm{PY}(d, \theta, G) \quad \forall l \\ G_{l,k} &\sim \mathrm{PY}(d, \theta, G_l) \quad \forall l, k \\ W_i | W_{i-1} = l &\sim G_l \\ W_i | W_{i-1} = l, W_{i-2} = k &\sim G_{l,k} \end{aligned}$$

Detailed descriptions regarding adaptor grammars is referred to Johnson et al. (2009)

3.2.4 Minimum Description Length Principle

The minimum description length (MDL) principle, originally developed in the context of information theory, was adopted in Bayesian statistics as a principled model selection method (Rissanen, 1978). Its connection to lexical acquisition was first uncovered in behavioral studies, and early applications focused mostly on applying MDL to induce word segmentation that results in compact lexicons (Kit and Wilks, 1999; Yu, 2000; Argamon et al., 2004).

Kit and Wilks proposed a compression-based approach, called *description length gain*, in 1999. In their formulation (Zhao and Kit, 2008), the description length gain is defined as:

$$L(C) - L(C[r \to w] \oplus r),$$

$C = \langle c_1, \dots, c_N \rangle$	Character sequence
$W = \langle w_1, \dots, w_M \rangle$	Word sequence, $M < N$
$U = \{u_0 = 0, u_1, \dots, u_K\}$	Positions of utterance boundaries
A_C	Character alphabet
A_W	Word alphabet (i.e., lexicon)
$f_{\langle c_i, \dots, c_j \rangle}$	n-gram frequency

Table 3.1: The notation used in the development of regularized compression.

where $C[r \rightarrow w]$ is the resulting string after replacing all occurrence of w in C with r. Note that $\oplus$ denotes the concatenation operator, and the description length $L(C)$ is obtained from the Shannon-Fano code of C. Specifically, $L(C) = |C| \times H(V(C))$, where $V(C)$ denotes the character vocabulary of C.

More recent approaches (Zhikov et al., 2010; Hewlett and Cohen, 2011) used MDL in combination with existing algorithms, such as branching entropy (Tanaka-Ishii, 2005; Jin and Ishii, 2006) and bootstrap voting experts (Hewlett and Cohen, 2009), to determine the best segmentation parameters. On various benchmarks, MDL-powered algorithms have achieved state-of-the-art performance, sometimes even surpassing that of the most sophisticated hierarchical modeling methods.

3.3 Preliminaries

In this section, we describe the notation and the basic assumptions in our work. In developing the notation, we try to be consistent with the mathematical convention, but we find that, in the procedure that we are about to motivate, certain mathematical aspects (e.g., sequence and set) regarding one underlying object cannot be easily expressed without clutter. For clarity, we intentionally abuse the notation in some cases, using the same expression to represent different mathematical viewpoints when the meaning is clear. A short summary about the notation is given in Table 3.1.

The input to our word segmentation algorithm is a set of unsegmented texts, each of which represents an utterance. Consider that this set is of K elements, and all the utterances consist totally of N characters. For brevity, we denote the input as a sequence of characters $C = \langle c_1, \ldots, c_N \rangle$, as if conceptually concatenating all the utterances into one string. Analogously, a segmented output is defined as a sequence of words $W = \langle w_1, \ldots, w_M \rangle$, for some $M \leq N$.

We represent the positions of all the utterance boundaries in C as a sequence of integers $U = \langle u_0 = 0, u_1, \ldots, u_K = N \rangle$, where $u_0 < u_1 < \ldots < u_K$. Therefore, to find the k-th utterance in C, we look at the characters between positions $u_{k-1}+1$ and u_k. In other words, the k-th utterance is the subsequence $\langle c_{u_{k-1}+1}, \ldots, c_{u_k} \rangle$.

A valid word segmentation W of C has to satisfy two properties. First, W represents the same piece of text as C does. Second, W *respects* the utterance boundaries U, meaning that any word $w_i \in W$ does not span over two utterances. In mathematical terms, there exists a sequence of boundaries $B = \langle b_0 = 0, b_1, \ldots, b_M = N \rangle$ such that: (i) $b_0 < b_1 < \ldots < b_M$, (ii) $c_{b_{i-1}+1} \ldots c_{b_i} = w_i$ for $1 \leq i \leq M$, and (iii) $U \subset B$.

The unique elements in a sequence implicitly define an alphabet set, or *lexicon*. This property is not limited to character or word sequences. Hereafter, for any sequence S, we denote its alphabet as A_S. As we shall find out later, sometimes we need to refer to a sequence that is either a mix of characters or words, or of the same type that we do not care. We use a generic term *token sequence* to this type of sequence.

3.4 Regularized Compression

Word segmentation results from compressing a sequence of characters. By compression, we refer to a series of steps that we can iteratively apply to the character sequence C C to produce the final result W. In each of these steps, called *compression steps*, we replace some subsequence $\langle c_i, c_{i+1}, \ldots, c_j \rangle$ in C with a string $w = c_i c_{i+1} \ldots c_j$ (word). It is clear that, for every possible W, such compression steps always exist.

Since the subsequence $\langle c_i, c_{i+1}, \ldots, c_j \rangle$ may occur multiple times in C, for simplicity, we further assume that, in each step, all the occurrences of subsequence $\langle c_i, c_{i+1}, \ldots, c_j \rangle$ are compressed at once. That is to say, we replace every subsequence in the set

$$\{\langle c_k, \ldots, c_l \rangle | c_k = c_i, c_{k+1} = c_{i+1}, \ldots, c_l = c_j\}$$

with the string w. Therefore, the compression step can be denoted as a *rewrite rule*, as in:

$$w \rightarrow \langle c_i, c_{i+1}, \ldots c_j \rangle.$$

This assumption greatly simplifies the modeling, since we do not need to worry about the compression order of these occurrences, while it is no longer clear whether going from any C to any W using compression is feasible. In the following discussion, we stick with this approximation and show that it does not lead to disastrous results. Relaxing this constraint is beyond the scope of this work and remains an open issue.

Here, we first review some basic properties of a compression step. Generally, applying a compression step to C has the following effects:

1. The total number of tokens in C decreases, and its alphabet set A_C is expanded to include a new token $w = c_i \ldots c_j$. Specifically, the size of C, denoted as $|C|$, is reduced by $f_{\langle c_i, \ldots, c_j \rangle}(j - i)$. When a compression step reduces more tokens, this step is said to be more *coherent*. It means that the subsequence to be replaced appears more frequently in the text, and thus is more likely to be a collocation;

2. Asymptotically, the vocabulary complexity, measured in terms of entropy rate, always increases. We can estimate the entropy rate empirically by seeing a token sequence as a series of outcomes drawn from some underlying stochastic process. The absolute change in entropy rate is called *deviation*.

The first property seems like a natural consequence. It says that the effort to describe the same piece of information, in terms of the number of tokens, get reduced at the expense of expanding the vocabulary. The second property is less obvious: It means that compression has a side effect of redistributing probability masses among observations, thereby causing deviation to entropy rate. Here, we describe a stronger result that the deviation is always positive, i.e., entropy always increases. A formal argument to this is given in Appendix A.1.

Since every compression step involves a choice of subsequence to be replaced, we can say that word segmentation is basically a combinatorial optimization problem. Understanding the effects gains us some insight to develop reasonable choices. In the following definitions, we seek to quantify the effects of cohesion and deviation. Specifically, they are defined with respect to two token sequences S_1 and S_2, such that S_2 is the result of applying some series of compression steps to S_1:

$$\text{cohesion} \equiv -|S_2|/|S_1|, \tag{3.1}$$

$$\text{deviation} \equiv |\tilde{H}(S_2) - \tilde{H}(S_1)|. \tag{3.2}$$

Note that, for some token sequence S, $\tilde{H}(S)$ denote the empirical entropy rates for the random variable $\mathcal{S} \in A_{\mathbf{S}}$, in which the probability $p(\mathcal{S})$ is estimated using the relative frequencies in sequence S.

We suggest that each choice shall be optimized toward high cohesion and low deviation. This is easily justified, because high cohesion leads to choices that cover more collocations, and low deviation translates to less effort for language users to harness the new vocabulary. Conceptually, to go from sequence S to some new sequence S' using only k compression steps, we consider the following procedure:

$$\begin{aligned} &\text{minimize} && (-\text{cohesion}(S, S'), \text{deviation}(S, S')) \\ &\text{subject to} && S' \text{ respects } U \\ & && r_1, \ldots, r_k \text{ are valid steps to go from } S \text{ to } S' \end{aligned} \tag{3.3}$$

Note that the objective is in vector form, and therefore it has to be scalarized by using a convex combination of its components. Specifically, by valid compression steps, we means that $r_1, \ldots, r_k$ satisfy:

$$\begin{aligned} &S^{(0)} = S, \\ &S^{(k)} = S', \\ &S^{(i-1)} \xrightarrow{r_i} S^{(i)} \text{ for } i = 1, \ldots, k. \end{aligned}$$

Using this procedure, we are able to find the best sequence, in terms of the aforementioned criteria, that is reachable in k compression steps from S. Nevertheless, this formulation is not very useful in practice, because in word segmentation, we want to find the best sequence within some predefined compression rate ρ. Compression rate is actually the average word length of the sequence, and we expect that matches the average word length of the language.

Rewriting the procedure to reflect this need can nevertheless trivialize our first decision criteria. When we impose a constraint on the compression rate, the cohesion will always be the same. Despite this issue, search in the entire space remains challenging because there are exponentially-many valid choices to consider. These issues are addressed in the next section using an iterative approximation method.

3.5 Iterative Approximation

Acknowledging that exponentially many feasible sequences need to be checked, we propose an alternative formulation in a restricted solution space. The idea is, instead of optimizing for segmentations, we search for *segmentation generators*, i.e., a set of functions that generate segmentations from the input. The generators we consider here is the *ordered rulesets.*

An ordered ruleset $R = \langle r_1, r_2, \ldots, r_k \rangle$ is a sequence of translation rules, i.e., compression steps. Each of these rules takes the following form:

$$w \to \langle c_1, c_2, \ldots, c_n, \rangle$$

where the right-hand side denotes some n-token subsequence to be replaced, and the left-hand side denotes the new token to be introduced. Applying an ordered ruleset R to a token sequence is equivalent to iteratively applying the translation rules $r_1, r_2, \ldots, r_k$ in strict order.

This notion of ordered rulesets allows one to explore the search space efficiently using a greedy inclusion algorithm. The idea is to maintain a globally best ruleset B that covers the best translation rules we have discovered so far, and then iteratively expand B by discovering new best rule and adding it to ruleset.

The following procedure repeats several times until the compression rate reaches some predefined value ρ. In each iteration, the best translation rule is determined by solving a modified version of Equation (3.3), which is written as follows:

$$\begin{aligned}
&\text{(In iteration } i\text{)} & & \\
&\text{minimize} & & -\alpha \times \text{cohesion}(S^{(i-1)}, S^{(i)}) + \text{deviation}(S^{(i-1)}, S^{(i)})) \\
&\text{subject to} & & S^{(i)} \text{ respects } U \\
& & & r \text{ is a valid compression step}
\end{aligned} \tag{3.4}$$

Note that the alternative formulation is largely a greedy version of Equation (3.3). The vector-form objective is scalarized by using a trade-off parameter α. A brief sketch of the algorithm is given in Algorithm 1.

3.6 Implementation

Additional care needs to be taken in implementation. The simplest way to collect n-gram counts for computing the objective in Equation (3.4) is to run multiple scans

Algorithm 1 The proposed word segmentation algorithm

```
Require: $\rho$
  $S \leftarrow C$
  for all $t = 1, 2, \ldots$ do
    if compression rate is less than or equal to $\rho$ then
      Leave the loop
    end if
    Solve Equation (3.4) to find the step $r$
    Rewrite $S$ in-place using $r$
  end for
  return $S$
```

over the entire sequence. Our experience suggests that using an indexing structure that keeps track of token positions can be more efficient. This is especially important when updating the affected n-gram counts in each iteration. Since replacing one occurrence for any subsequence affects only its surrounding n-grams, the total number of such affected n-gram occurrences in one iteration is linear in the number of occurrences for the replaced subsequence. Using an indexing structure in this case has the advantage to reduce seek time.

A detailed description about the implementation is discussed in Appendix A.2. Note that, however, the overall running time remains in the same complexity class regardless of the deployment of an indexing structure. The time complexity for this algorithm is $O(TN)$, where T is the number of iterations and N is the length of the input sequence.

Although it is theoretically appealing to create an n-gram search algorithm, in this preliminary study we used a simple bigram-based implementation for efficiency. We considered only bigrams in creating translation rules, expecting that the discovered bigrams can grow into trigrams or higher-order n-grams in the subsequent iterations. To allow unmerged tokens (i.e., characters that was supposed to be in one n-gram but eventually left out due to bigram implementation) being merged into the discovered bigram, we also required that that one of the two participating tokens at the right-hand side of any translation rule has to be an unmerged token. This has a side effect to exclude generation of collocation-based words[1]. It can be an issue in certain standards; on the test corpora we used, this kind of problems is not obvious.

Another constraint that we added to the implementation is to limit the choice of bigrams to those having more frequency counts. Generally, the number of occurrence for any candidate bigram being considered in the search space has to be greater or equal to some predefined threshold. In practice, we found little difference in performance for specifying any integer between 3 and 7 as the threshold; in this paper, we stick to 3.

3.7 Evaluation

3.7.1 Setup

We conducted a series of experiments to investigate the effectiveness of the proposed segmentation method under different language settings and segmentation standards. In the first and the second experiments, we focus on drawing comparison between our method and state-of-the-art approaches. The third experiment focuses on the influence of data size to segmentation accuracy.

Segmentation performance is assessed using standard metrics, such as precision, recall, and F-measure. Generally, these measures are reported only at word level; in some cases where further analysis is called for, we report boundary-level and type-level measures as well. We used the evaluation script in the official HDP package to calculate these numbers.

The reference methods considered in the comparative study and their abbreviations are listed as follows:

- HDP: Hierarchical Dirichlet process (Goldwater et al., 2009);
- NPY: Nested Pitman-Yor process (Mochihashi et al., 2009);

[1] Fictional examples include “homework” or “cellphone”.

- AG: Adaptor grammars (Johnson and Goldwater, 2009);
- Ent-MDL: Branching entropy + MDL (Zhikov et al., 2010);
- BVE-MDL: Bootstrap voting experts + MDL (Hewlett and Cohen, 2011);
- DLG: Description length gain (Zhao and Kit, 2008).

The proposed method is denoted as RC; it is also denoted as RC-MDL in a few cases where MDL is used for parameter estimation.

3.7.2 Parameter Estimation

There are two free parameters α and ρ in our model. The parameter α specifies the degree to which we favors high-frequency collocations when solving Equation (3.4). Experimentation suggests that α can be sensitive when set too low[2]. Practically, we recommend optimizing α based on grid search on development data, or the MDL principle. The formula for calculating description length is not shown here; see Zhikov et al. (2010), Hewlett and Cohen (2011), and Rissanen (1978) for details.

The expected compression rate ρ determines when to stop the segmentor. It is related to the expected word length: When the compression rate $|C|/|W|$ reaches ρ and the segmentor is about to stop, $1/\rho$ is the average word length in the segmentation. In this sense, it seems ρ is somehow connected to the language of concern. We expect that optimal values learned on one data set may thus generalize on the other sets of the same language. Throughout the experiments, we estimated this value based on development data.

[2] Informally speaking, when $\alpha < \tilde{H}(C)$. The analysis is not covered in this preliminary study.

(a) Bernstein-Ratner

	P	R	F	BP	BR	BF	TP	TR	TF
HDP	.75	.70	.72	.90	.81	.85	.64	.55	.59
RC-MDL	.77	.82	.79	.85	.92	.89	.57	.48	.50

(b) Bakeoff 2005

	P	R	F	BP	BR	BF	TP	TR	TF
RC, CityU training	.75	.79	.77	.89	.93	.91	.63	.35	.45
RC, MSR training	.73	.82	.77	.86	.96	.91	.70	.26	.38

Table 3.2: Performance evaluation for the proposed method across different test corpora. The first row indicates a reference HDP run (Goldwater et al., 2009); the other rows represent the proposed method tested on different test corpora. Columns indicates performance metrics, which correspond to precision, recall, and F-measure at word (P/R/F), boundary (BP/BR/BF), and type (TP/TR/TF) levels.

3.7.3 Evaluation on Bernstein-Ratner Corpus

We conducted the first experiment on the Bernstein-Ratner corpus (Bernstein-Ratner, 1987), a standard benchmark for English phonetic segmentation. We used the version derived by Michael Brent, which is made available in the CHILDES database (Brent and Cartwright, 1996; MacWhinney and Snow, 1990). The corpus comprises 9,790 utterances, which amount to 95,809 words in total. Its relatively small size allows experimentation with the most computational-intensive Bayesian models.

Parameter estimation for the proposed method has been a challenge due to the lack of appropriate development data. We first obtained a rough estimate for the compression rate ρ via human inspection into the first 10 lines of the corpus (these 10 lines were later excluded in evaluation) and used that estimate to set up the termination condition. Since the first 10 lines are too small to reveal any useful segmentation cues other than the word/token ration of interest, we considered this setting ("almost unsupervised") a reasonable compromise. In this experiment, ρ is set to 0.37; the trade-off parameter α is set to 8.3, optimized using MDL principle in a two-pass grid search (the first pass over $\{1, 2, \ldots, 20\}$ and the second over $\{8.0, 8.1, \ldots, 10.0\}$).

	P	R	F	Time
HDP	.752	.696	.723	–
NPY, bigram	.748	.767	.757	17 min.
AG	–	–	<u>.890</u>	–
Ent-MDL	.763	.745	.754	2.6 sec.
BVE-MDL	<u>.793</u>	.734	.762	2.6 sec.
RC-MDL	.771	<u>.819</u>	.794	.9 sec.

Table 3.3: Performance evaluation on the Bernstein-Ratner corpus. The reported values for each method indicate word precision, recall, F-measure and running time, respectively. The underlined value in each column indicates the top performer under the corresponding metric.

A detailed performance result for the proposed method is described in Table 3.2. A reference run for HDP is included for comparison. The proposed method achieved satisfactory result at word and boundary levels. Nevertheless, low type-level numbers (in contrast to those for HDP) together with high boundary recall suggested that we might have experienced over-segmentation.

Table 3.3 covers the same result with less details in order to compare with other reference methods. All the reported measures for reference methods are directly taken from the literature. The result shows that AG achieved the best performance in F-measure (other metrics are not reported), surpassing all the other methods by a large margin (10 percent). Among the other methods, our method paired with MDL achieved comparable performance as the others in precision; it does slightly better than the others in recall (5 percent) and F-measure (2.5 percent). Furthermore, our algorithm also seems to be competitive in terms of computational efficiency. On this benchmark it demanded only minimal memory low as 4MB and finished the segmentation run in 0.9 second, even less than the reported running time for both MDL-based algorithms.

3.7.4 Evaluation on Bakeoff-2005 Corpus

The second benchmark that we adopted is the SIGHAN Bakeoff-2005 dataset (Emerson, 2005) for Chinese word segmentation. The corpus has four separates subsets

Corpus	Training (W/T)	Test (W/T)
AS	5.45M / 141K	122K / 19K
PKU	1.1M / 55K	104K / 13K
CityU	1.46M / 69K	41K / 9K
MSR	2.37M / 88K	107K / 13K

Table 3.4: A short summary about the subsets in the Bakeoff-2005 dataset. The size of each subset is given in number of words (W) and number of unique word types (T).

prepared by different research groups; it is among the largest word segmentation benchmarks available. Table 3.4 briefly summarizes the statistics regarding this dataset.

We decided to compare our algorithm with description length gain (DLG), for that it seems to deliver best segmentation accuracy among other unsupervised approaches ever reported on this benchmark (Zhao and Kit, 2008). Since the reported values for DLG were obtained on another closed dataset Bakeoff-2006 (Levow, 2006), we followed a similar experimental setup as suggested in the literature (Mochihashi et al., 2009): We compared both methods only on the training sets for the common subsets CityU and MSR. Note that this experimental setup departed slightly from that of Mochihashi et al. in that all the comparisons were strictly made on the training sets. The approach is more straightforward than the suggested sampling-based method.

Other baseline methods that we considered include HDP, Ent-MDL, and BVE-MDL, for their representativeness in segmentation performance and ease of implementation. The HDP implementation we used is a modified version of the official HDP package[3]; we patched the package to make it work with Unicode-encoded Chinese characters. For Ent-MDL and BVE-MDL, we used the software package[4] distributed by Hewlett and Cohen (2011). We estimated the parameters using the AS training set as the development data. We set α to 6 based on a grid search. The expected compression rate ρ that we learned from the development data is 0.65.

[3] http://homepages.inf.ed.ac.uk/sgwater/

[4] http://code.google.com/p/voting-experts

	CityU	MSR
DLG, ensemble	.684	.665
Ent-MDL, $n_{max} = 3$	.798	.795
RC, $r = .65$	.770	.774

Table 3.5: Performance evaluation on the common training subsets in the Bakeoff-2005 and Bakeoff-2006 datasets. The reported values are token F-measure. The boldface value in each column indicates the top performer for the corresponding set.

In Table 3.2, we give a detailed listing of various performance measures for the proposed method. Segmentation performance seems moderate at both word and boundary levels. Nevertheless, high type precision and low type recall on both CityU and MSR training corpora signaled that our algorithm failed to discover most word types. This issue, we suspect, was caused by exclusion of low-frequency candidate bigrams, as discussed in Section 3.6.

Table 3.5 summarizes the result for word segmentation conducted on the CityU and MSR subsets of Bakeoff-2005. Due to practical computational limits, we were not able to run HDP and BVE-MDL on any complete subset. The result shows that our algorithm outperforms DLG by 8 to 10 percents in F-measure, while Ent-MDL still performs slightly better, achieving the top performance among all the experimental runs on both subsets.

To compare with HDP, we conducted another test run on top of a random sample of 1,000 lines from each subset. We chose 1,000 lines because HDP can easily consume more than 4GB of main memory on any larger sample. We adopted standard settings for HDP: $\alpha_0 = 3,000$, $\alpha_1 = 300$, and $p_b = 0.2$. In each trial run, we ran the Gibbs sampler for 20,000 iterations using simulated annealing (Goldwater et al., 2009). We obtained 10 samples from the Gibbs sampler and used the average performance in comparison. It took slightly more than 50 hours to collect one trial run on one subset.

The evaluation result is summarized in Table 3.6. We ran our algorithm to the desired compression ratio $r = 0.65$ on this small sample. The result shows that the performance of regularized compression is inferior to that of HDP by 9 to 13 percents

	CityU-1k	MSR-1k
HDP, 10 sample average	.591	.623
RC, $r = .65$	.505	.492
RC, $r = .65$/punc.	.599	.591

Table 3.6: Performance evaluation on two random samples from the common sets (CityU and MSR subsets) in the Bakeoff-2005 and Bakeoff-2006 datasets. Note that the third run is an outside test.

in F-measure for both sets. To investigate why, we looked into the segmentation output. We observed that, in the regularized compression output, most of the punctuation marks were incorrectly aligned to their neighboring words, owing to the short of frequency counts in this small sample. The HDP, however, does not seem to suffer from this issue.

We devised a simple post-processing step, in which each punctuation mark was forced segmented from the surrounding text. Another outside test was conducted to see how well the algorithm works using heuristics derived from minimal domain knowledge. The additional run is denoted as RC/punc. The result is shown in Table 3.6. From the result, we found that the combined approach works slightly better than HDP in one corpus, but not in the other.

3.7.5 Effects of Data Size

We employed the third experiment to study the influence of corpora size to segmentation accuracy. Since the proposed method relies on empirical estimates for entropy rate to decide the word boundaries, we were interested in learning about how it responds to relatively low and high volume input.

This experiment was conducted on CityU and MSR training sets. On each corpus, we took the first k% of data (in terms of utterances) and tested the proposed method against that subset; this test was repeated several times with different values for k. In this experiment, we chose the value for k from the set $\{2, 4, 6, 8, 10, 20, 30, \ldots, 90, 100\}$. The performance is evaluated using word, boundary, and type F-measures.

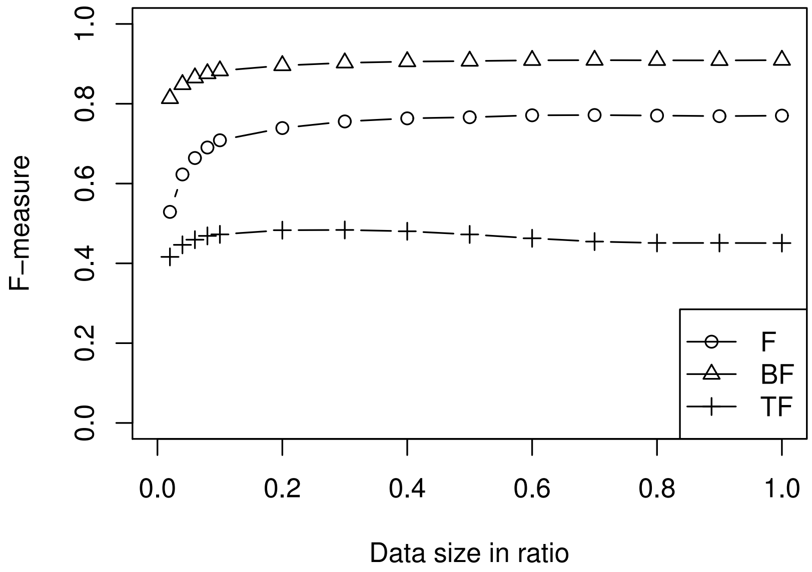

Figure 3.1: Performance evaluation for the proposed method on the CityU training set.

Figures 3.1 and 3.2 show the experiment results. Both figures revealed similar patterns for segmentation performance at different volume levels. Word F-measures for both corpora begin at roughly 0.52, climb up rapidly to 0.73 as the volume grows from 2% to 20%, and finally settle on some value around 0.77. Boundary F-measures for both corpora show a similar trend—a less steep increase before 20% from 0.80 to 0.89 followed by a plateau at around 0.93. Here, the result seems to suggest that estimating token entropy rate using less than 20% of data might be insufficient for this type of text corpora. Furthermore, since performance is saturated at such an early stage, it seems feasible to split the entire dataset into a number of folds (e.g., 5, in this case) and solve each fold individually in parallel. This technique may greatly enhance the run-time efficiency of the segmentor.

The patterns we observed for type F-measure tells another story. On both corpora, type F-measures do not seem to improve as data volume increases. On CityU corpora, type F-measure gradually increased from 0.42 to 0.48 and then slowly falling back to 0.45. On MSR corpora, type F-measure peaked at 0.45 when receiving

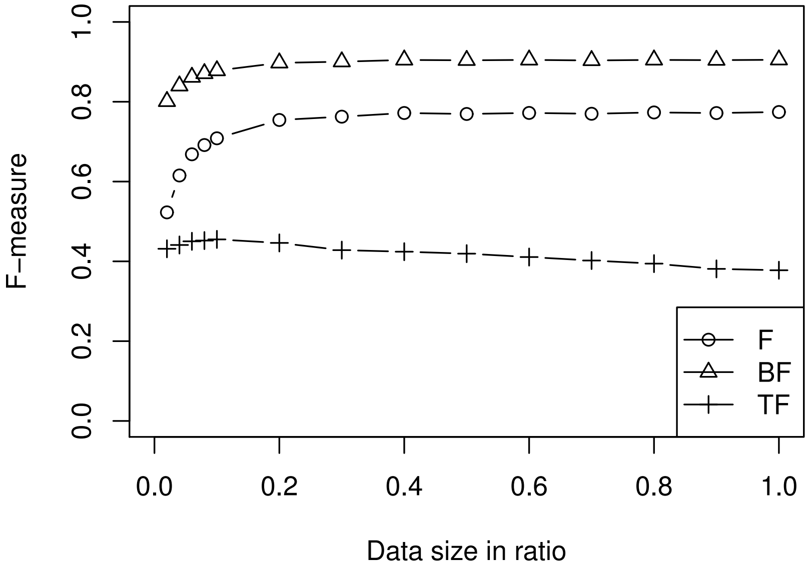

Figure 3.2: Performance evaluation for the proposed method on the MSR training set.

10% of data; after that it started decreasing, going all the way down to 0.37, even lower than the number 0.43 it received at the beginning. Our guess is that, at some early point (20%), the proposed method started to under-segment the text. We suspect that there is some deep connection between performance saturation and under-segmentation, since from the result they both begin at roughly the same level. Further investigation in this respect is needed to give out definitive explanations.

3.8 Concluding Remarks

Preliminary experimental results suggest that the regularized compression method, even only with partial evidence, seems as effective as the state-of-the-art methods in different language settings. When paired with MDL criteria, regularized compression is comparable to hierarchical Bayesian methods and MDL-based algorithms in terms of segmentation accuracy and computational efficiency. Furthermore, regularized compression is less memory-demanding than the other approaches; thus, it scales more easily to large corpora for carrying out certain tasks such as segmenting historical texts written in ancient languages, or preprocessing a large dataset for subsequent manual annotation.

We have identified a number of limitations of regular compression. First, the choice of candidate n-grams does not cover *hapax legomena*, i.e., words that occur only once in the corpus. At present, precluding these low-frequency n-grams seems to be a necessary compromise due to our limited understanding about the dynamics behind regular compression. Second, regularized compression does not work well with low volume data, since on smaller dataset the distribution of frequency counts is less precise. Third, the algorithm may stop identifying new word types at some point. We suspect that this is related to the choice of n-gram, since in our implementation no two existing "words" can be aggregated into one. These issues shall be addressed in our future work.

Chapter 4

Static Index Pruning

4.1 Overview

In information retrieval, static index pruning refers to a task that reduces the index size during or immediately after index construction. Doing so better utilizes the disk space required to store the index and improves the overall query throughput (Altingovde et al., 2009). This concept was brought to earth in the groundbreaking work of Carmel et al. (2001). Static index pruning was motivated in the context of Web search, by the growing need to efficiently operate retrieval on top of tremendous amount of information. To date, it has gathered much attention for its implication on the Web-scale text collections (Blanco and Barreiro, 2010; Büttcher and Clarke, 2006).

In a typical retrieval system, we use an inverted index to keep track of the term-to-document mapping in the underlying text collection, and an index entry, or a *posting*, records one such term-document association. Keeping the complete term-to-document mapping in the index is usually suggested since it preserves the document coverage for any query term, but it also poses a serious data management problem as the underlying collection grows.

Static index pruning seeks a balance between efficiency and effectiveness in such a situation. By permanently removing some subset of index entries from a production index, efficiency, in terms of disk usage and query throughput, comes at the sacrifice of retrieval accuracy. This leads to an interesting research question: When we reduce the index size down to some fixed ratio, how do we minimize the loss in retrieval effectiveness?

Many previous efforts consider the impact of individual entries. *Impact* is the individual contribution of a term-document pair to the final retrieval score. The idea is to order index entries according to *impact*, and then remove low-impact entries from the index, since they are less as influential in retrieval as high-impact ones.

This approach has been shown successful in practice. Carmel et al. (2001) used retrieval scores, such as tf-idf or BM25, to assess the impact of individual postings. Büttcher and Clarke (2006) measured the impact of one posting (t, d) based on the contribution of term t to the Kullback-Leibler divergence score between document d and the entire collection. Blanco and Barreiro (2010) developed a decision criterion based on odd-ratio of relevance in probability ranking principle (PRP) (Robertson, 1997).

Alternative decision criteria other than impact have also been investigated. Examples include entropy, informativeness, and discriminative value (Blanco and Barreiro, 2007; Zheng and Cox, 2009); some of them have been shown useful in specific query scenarios.

In this chapter, we revisit this problem in the view of information preservation. The insight is that an inverted index is essentially a nonparametric predictive model $p(d|t)$, with which one estimates the likelihood of some document d being relevant to some query term t, and pruning this model permanently removes the connections between some terms and some documents, thereby causing a loss in predictive power. We propose using the conditional entropy $H(D|T)$ to quantify the predictive power and suggest minimizing the loss in static index pruning by considering the contribution of individual index entries to the conditional entropy as the decision criteria.

4.2 Related Work

The research on static index pruning is advocated by the two most popular work, *term-centric approach* and *document-centric pruning*, which differ slightly in the pruning strategy they take. Both methods have shown promising results in efficiency and effectiveness on standard benchmarks. In the following paragraphs, we brief describe these methods and discuss their strengths and weaknesses.

4.2.1 Term-Centric and Document-Centric Methods

The approach is proposed by Carmel et al. (Carmel et al., 2001), and is so named because it focuses on reducing the posting list for each term in the index. The idea is to keep only a subset of the postings, called the top answers, for each term t. Generally, this is done by first sorting all the postings for term t according to some score function $A(t, d)$, and then permanently removing away those entries not in the top set. Carmel et al. gave two interpretation about the top answers, one defined by a fixed number k, indicating the top-k entries, and the other by a ratio $0 \leq \delta \leq 1$, indicating the proportion the top set takes in the corresponding posting list.

In contrast to the term-centric approach, the document-centric pruning seeks to reduce the posting list for each document. Büttcher and Clarke (Büttcher and Clarke, 2006) motivated the idea by removing away less-important entries for each document. They approached this idea from the language modeling point of view by considering the contribution for each term t to the Kullback-Leibler divergence score $KLD(d, C)$ between the document d and the collection model C. By sorting all the postings for document d according to this quantity, the document-centric pruning method keeps only the top $\lambda|d|$ entries in the index and discards everything else. Note that $0 \leq \lambda \leq 1$ is some given pruning ratio.

4.2.2 Recent Development

Blanco and Barreiro (2010) developed a decision criterion based on the probability ranking principle (PRP) (Robertson, 1997). The idea is to take every term in the index as a single-word query and calculate the odd-ratio of relevance:

$$p(r|t,d)/p(\overline{r}|t,d).$$

For any term-document pair, the lower this quantity, the more likely the term is irrelevant to the document. Therefore, to prune the index, one shall sort all the entries according to this score and remove the low-scoring ends. A parameter, ϵ, is used to serve as the cutting threshold.

Recently, Altingovde et al. (Altingovde et al., 2012) proposed an interesting query-view-based technique for static pruning. This technique works orthogonally with other methods. It can be used to pre-screens the index entries identifying important entries. These entries are marked as "do-not-prune". The subsequent (true) pruning method that takes over respects these marks when doing its job. Generally, query-view method relies on external resources, i.e., query logs, to help uncovering important index entries. By definition, the query view of a query is the the top document returned to the user. For an l-term query, every document in its query view is said to be associated with any of these constituent l terms. By repeatedly running the queries collected from external sources, one can easily obtain a huge set of associated term-document pairs.

4.3 Information Preservation

Information retrieval is a practice about ranking documents in response to information needs. To achieve optimal performance, documents shall be retrieved in order of the decreasing probability of relevance (Robertson, 1997). This notion of relevance forms the foundation of modern information retrieval. Modeling relevance has become the most essential aspect in development of retrieval methods.

One of the major functions that retrieval models provide is to assess the probability of some document d being relevant to some given query q, written as $p(d|q)$. This is conceptually true for every retrieval model, but does not necessitate explicit modeling of $p(d|q)$. In some retrieval model, the probability never appear but the output, i.e., retrieval scores, are nevertheless rank-equivalent to $p(d|q)$.

Since a query consists of a number of query terms, in practice, this probability is postulated as some combination, e.g., a product, of individual probabilities $p(d|t)$ that assess term relevance. These probabilities are evaluated according to the information we store in the index.

The model $p(d|t)$ is actually a nonparametric *predictive model*, in a sense that prediction is made over the choice of documents with respect to the textual input from users. Nonparametric models do have parameters, while the number of parameters is not fixed. In our case of $p(d|t)$, the parameter set is a set of tuples O stored in the index, in the following form. Here, $f_{t,d}$ denotes the frequency of term t inside document d:

$$O = \{(t, d, f_{t,d}) | t \in T, d \in D, f_{t,d} > 0\}. \tag{4.1}$$

To model the choice in static index pruning, we introduce a conceptual construct into the parameter space. We add a binary indicator a to the end of each term-document tuple. When $a = 1$, we say that the tuple is *active*, meaning that it stays in the index. When $a = 0$, the tuple is removed away. In an unpruned index, all the entries are active. The resulting parameter set is now written as:

$$\theta = \{(t, d, f_{t,d}, a) | (t, d, f_{t,d}) \in O\}. \tag{4.2}$$

Accordingly, the unpruned model θ_0 is defined as:

$$\theta_0 = \{(t, d, f_{t,d}, 1) | (t, d, f_{t,d}) \in O\}. \tag{4.3}$$

In static index pruning, we concern about how to preserve as much predictive power of $p(d|t)$ when a considerable amount of information is discarded. Generally, predictive power is measured in terms of entropy, and it is natural to consider the conditional entropy $H(D|T)$ alone. Nevertheless, since predictive power also depends on activeness of index entries, we suggest using another conditional entropy $H(D|T, A)$ to assess the predictive power of a pruned model. This conditional entropy is a summary statistic regarding how difficult it is to predict the right outcome D (document) given the predictor T (term) and A (activeness).

We first establish that the following relation holds for every choice of parameter θ with respect to O:

$$H_{\theta_0}(D|T) = H_\theta(D|T). \tag{4.4}$$

The subscripts denotes the parameter set in use when estimating the conditional entropy. Since activeness is the only difference between different settings of parameters and it is not involved in the evaluation of $H(D|T)$, this equation trivially holds.

Next, we show that $H_\theta(D|T, A) \leq H_{\theta_0}(D|T, A)$ for every θ:

$$\begin{aligned} H_\theta(D|T, A) &\leq H_\theta(D|T) \\ &= H_{\theta_0}(D|T) \\ &= H_{\theta_0}(D|T, A) + I_{\theta_0}(D|T; A) \\ &= H_{\theta_0}(D|T, A). \end{aligned} \tag{4.5}$$

It is clear that $I_{\theta_0}(D|T; A) = 0$ because every index entry in θ_0 is by default active. This equation suggest that splitting the index into subsets, i.e., the active and inactive sets, only reduces the overall predictive power.

In fact, a pruned index operates at a even lower level because the submodel for the inactive set provides no information at all. Let $\hat{H}_\theta(D|T,A)$ denote the true predictive power exhibited by a pruned index. In the following equation, we show that $\hat{H}_\theta(D|T,A) \leq H_\theta(D|T,A)$:

$$\begin{aligned}\hat{H}_\theta(D|T,A) &= p_\theta(A=1)\hat{H}_\theta(D|T,A=1) + p_\theta(A=0)\hat{H}_\theta(D|T,A=0)\\ &= p_\theta(A=1)H_\theta(D|T,A=1)\\ &\leq p_\theta(A=1)H_\theta(D|T,A=1) + p_\theta(A=0)H_\theta(D|T,A=0)\\ &= H_\theta(D|T,A).\end{aligned} \tag{4.6}$$

By writing it as an information preservation problem, we minimize the absolute change in entropy. In this derivation, we use the true entropy $\hat{H}_\theta(D|T,A)$ rather than $H_\theta(D|T,A)$. Let θ^* denote the best model and ρ denote a predefined prune ratio. We have the following equations:

$$\begin{aligned}\theta^* &= \mathop{\arg\min}_{\theta: p_\theta(A=0)\geq\rho} |\hat{H}_\theta(D|T,A) - H_{\theta_0}(D|T,A)|\\ &= \mathop{\arg\min}_{\theta: p_\theta(A=0)\geq\rho} H_{\theta_0}(D|T,A) - \hat{H}_\theta(D|T,A)\\ &= \mathop{\arg\max}_{\theta: p_\theta(A=0)\geq\rho} \hat{H}_\theta(D|T,A)\\ &= \mathop{\arg\max}_{\theta: p_\theta(A=0)\geq\rho} H_\theta(D|T,A=1),\end{aligned} \tag{4.7}$$

and an equivalent optimization problem is written as follows:

$$\begin{aligned}&\text{maximize} && H_\theta(D|T,A=1)\\ &\text{subject to} && p_\theta(A=0)\geq\rho\end{aligned} \tag{4.8}$$

4.4 Compositional Approximation

The information preservation formulation is actually a combinatorial optimization problem. Solving this problem exact needs to consider exponentially-many possible choices, which is infeasible in practice.

In this section, we show that an approximate solution can be found by *decomposition.* The key assumption is that (i) the objective is decomposable and (ii) it can be written as the summation of individual contributions, $h_{d|t}$, of a subset of term-document pair $O_\theta \subset O$. The mathematical definition of this assumption is written as the following two equations:

$$H_{\theta_0}(D|T, A = 1) = \sum_{O} h_{d|t}, \tag{4.9}$$

$$H_{\theta}(D|T, A = 1) = \sum_{O_\theta \subset O} h_{d|t}. \tag{4.10}$$

Since our goal is to find a subset that maximizes the objective, it suffices to rewrite the optimization problem as follows:

$$\begin{array}{ll} \text{maximize} & \sum_{O_A \subset O} h_{d|t} \\ \text{subject to} & (1 - |O_A|)/|O| \geq \rho \end{array} \tag{4.11}$$

We first show that it is possible to write the conditional entropy $H_{\theta_0}(D|T, A = 1)$ as a summation over all the term-document pairs in the index. For brevity, in the following derivation, we drop the subscript θ_0 and the activeness condition and use the notation $H(D|T)$ to denote the conditional entropy of the unpruned model.

Generally, it is written as:

$$H(D|T) = \sum_{t \in T} p(t) \left(- \sum_{d \in D} p(d|t) \log p(d|t) \right), \tag{4.12}$$

where $p(t)$ denotes the probability of term t being used in queries, and $p(d|t)$ is the predictive model that assesses the relevance between document d and term t.

The distribution $p(t)$ is independent of the retrieval model in use. To estimate $p(t)$, we simply assume that it is uniformly distributed. Note that this estimate can be further improved by using session logs. Now, we go ahead and rewrite $H(D|T)$ as a summation of *uncertainties* $h_{d|t}$ contributed by individual term-document pairs to the model:

$$H(D|T) = \frac{1}{|T|} \sum_{t \in T} \sum_{d \in D} h_{d|t}, \tag{4.13}$$

where $h_{d|t}$ is defined as follows:

$$h_{d|t} = -\frac{p(t|d)p(d)}{\sum_{d'} p(t|d')p(d')} \log \frac{p(t|d)p(d)}{\sum_{d'} p(t|d')p(d')}. \tag{4.14}$$

Consider any two term-document pairs (t, d) and (t', d') such that $h_{d|t} < h_{d'|t'}$. The formulation implies that predicting d from t requires less information than predicting d' from t'. That is to say, we are more certain about the connection from t to d, and, in this case, we argue that removing (t, d) from the index has less effect on the overall predictive power than removing (t', d'). By setting a cutting threshold ϵ, it is now straightforward to scan over the entire index and discard any entry whose uncertainty $h_{d|t}$ is strictly lower than ϵ. In other words, discarding these entries, i.e., $h_{d|t} < \epsilon$, guarantees to minimize overall information loss and to retain the most predictive power with respect to a specific choice of ϵ.

Based on this idea, we propose a simple index pruning algorithm. It takes a threshold value ϵ as input, then it scans over the entire index and discard any index entry whose uncertainty $h_{d|t}$ is strictly less than ϵ. Algorithm 2 summarizes the proposed procedures.

Algorithm 2 The proposed index pruning algorithm based on information preservation.

```
Require: $\epsilon$
  for all $t \in T$ do
    for all $d \in$ postings($t$) do
      Compute $h_{d|t}$ using Equation (4.14)
      if $h_{d|t} < \epsilon$ then
        Remove $d$ from postings($t$)
      end if
    end for
  end for
```

4.5 Experiments

4.5.1 Setup

We chose two baseline approaches to compare our methods with: top-k term-centric pruning (denoted as TCP) and probability ranking principle (denoted as PRP). We implemented both methods using the Indri API[1]. Our implementation does not update the document length values after pruning. For TCP, we set $k = 10$ to maximize the precision for the top 10 documents (Carmel et al., 2001) and used BM25 as the score function. For PRP, we set $\lambda = 0.6$ for query likelihood estimation, and applied the suggested approximations to estimate the rest of the probabilities (Blanco and Barreiro, 2010). These probability estimates are summarized as follows.

$$\begin{aligned} p(t|D) &= (1-\lambda)p_{\mathrm{ML}}(t|D) + \lambda p(t|C), && (4.15)\\ p(r|D) &= \frac{1}{2} + \frac{1}{10}\tanh\frac{dl - \overline{X_d}}{S_d}, && (4.16)\\ p(t|\overline{r}) &= p(t|C). && (4.17) \end{aligned}$$

For the proposed method, a similar setting as in the baselines was adopted to minimize the effect of retrieval method. We used Equation (4.15) to estimate the query likelihood $p(t|d)$ (setting $\lambda = 0.6$). For estimating document prior $p(d)$, we experimented with two approaches, which are *tangent approximation* as in Equation (4.16)

[1] `http://www.lemurproject.org/indri.php`

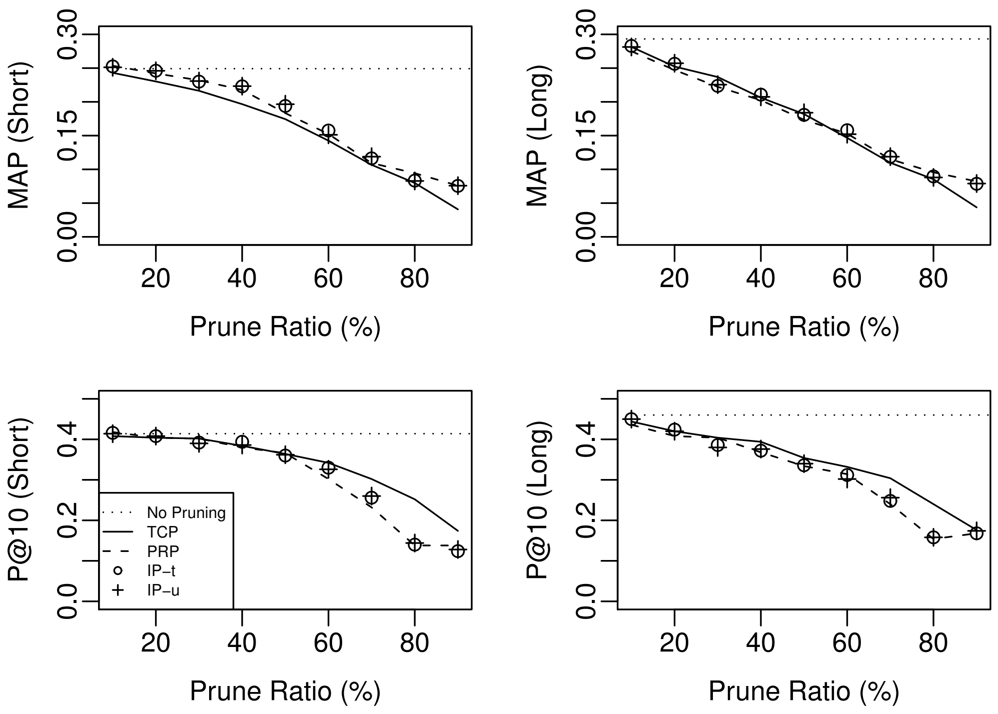

Figure 4.1: Performance results for all the methods on WT2G. Rows indicate different performance measures (MAP/P@10). Columns indicate different query types (short/long).

(denoted as IP-t) and *uniform prior*, i.e., $p(d) = 1/|D|$ (denoted as IP-u).

We managed to control the prune ratio at different levels (e.g., 10%, 20%, . . . , 90%.) For PRP and IP-based methods, the prune ratio depends on a global threshold ϵ. To prune the index to the right size, we sample the decision scores from the entire index to estimate the percentiles, and then use the estimates to find the right threshold value. For TCP, we manually adjust the parameter ϵ to approach the designated prune ratio. In our experiments, the error is controlled to roughly ±0.2% in prune ratio.

4.5.2 Retrieval Performance

We conducted a series of experiments on the LATimes, TREC-8, and WT2G corpora, using TREC topics 401-450 as queries. We tested two different query types,

Short Query (MAP/P@10 at 0%: 210/250)

MAP	10%	20%	30%	40%	50%	60%	70%	80%	90%
TCP	209	206	201	193	177	$\underline{168}$	143	$\underline{124}$	073
PRP	$\underline{211}$	209	$\underline{207}$	201$^\blacktriangle$	190	158	141	113	$\underline{098}$
IP-t	210	210	$\underline{207}$	$\underline{203}^\blacktriangle$	188	161	148	109	097
IP-u	210	$\underline{210}$	207	203$^\blacktriangle$	$\underline{191}^\blacktriangle$	164	$\underline{151}$	104	090

P@10	10%	20%	30%	40%	50%	60%	70%	80%	90%
TCP	252	244	246	238	228	218	$\underline{204}$	$\underline{194}$	128
PRP	$\underline{254}$	$\underline{256}$	254	248	234	212	172	124$^\blacktriangledown$	130
IP-t	250	$\underline{256}^\blacktriangle$	$\underline{258}$	$\underline{254}$	234	218	168	110$^\blacktriangledown$	130
IP-u	250	$\underline{256}^\blacktriangle$	$\underline{258}$	250	$\underline{236}$	$\underline{224}$	184	116$^\blacktriangledown$	$\underline{138}$

Long Query (MAP/P@10 at 0%: 235/260)

MAP	10%	20%	30%	40%	50%	60%	70%	80%	90%
TCP	232	$\underline{230}$	$\underline{217}$	206	174	$\underline{171}$	$\underline{153}$	116	075
PRP	228	221	206	202	$\underline{193}$	160	141	$\underline{119}$	$\underline{106}$
IP-t	232	221	215$_\sharp$	$\underline{207}$	187	160	144	116	103
IP-u	$\underline{234}$	221	216	204	188	168	149	113	098

P@10	10%	20%	30%	40%	50%	60%	70%	80%	90%
TCP	262	$\underline{264}$	256	242	238	$\underline{234}$	$\underline{212}$	$\underline{188}$	$\underline{142}$
PRP	$\underline{264}$	256	242	$\underline{252}$	$\underline{248}$	228	174	132$^\blacktriangledown$	132
IP-t	258	254	254	244	244	222	164	122$^\blacktriangledown$	130
IP-u	256	256	$\underline{258}$	244	242	232	182	118$^\blacktriangledown$	136

Table 4.1: The overall performance results on LATimes. We round down all the reported measures to the 3rd digit under the decimal point, and ignore preceding zeroes and decimal points for brevity. Underlined entries indicate the best performance in the corresponding group. Entries that are significantly superior or inferior ($p < 0.05$) to TCP are denoted by superscripts $\blacktriangle$ or $\blacktriangledown$, respectively. Analogously, entries that are significantly superior or inferior to PRP are denoted by subscripts $\sharp$ or $\flat$, respectively.

short (using title) and long (using title and description). We used BM25 to retrieve documents in all the experimental runs. Performance is evaluated using mean average-precision (MAP) and precision-at-10 (P@10).

Performance result is given in both figural and tabular formats. Figure 4.1 shows the evaluation result on WT2G[2]. The result is summarized in four plots, each indicating different combination of query types and performance measures. Each method is plotted as a curve or a series of points according to the measured performance (y-axis) at some prune ratio (x-axis). Tables 4.1, 4.2, and 4.3 covers the detailed

[2] Results on the other two corpora show similar trends and are therefore omitted here.

Short Query (MAP/P@10 at 0%: 228/436)

MAP	10%	20%	30%	40%	50%	60%	70%	80%	90%
TCP	223	213	204	191	176	154	126	094	055
PRP	226	221	215	201	181	157	143	$\underline{147}^{\blacktriangle}$	$103^{\blacktriangle}$
IP-t	$\underline{227}_{\sharp}^{\blacktriangle}$	$\underline{223}^{\blacktriangle}$	$215^{\blacktriangle}$	202	$186_{\sharp}$	160	$\underline{147}$	$\underline{147}^{\blacktriangle}$	$\underline{106}^{\blacktriangle}$
IP-u	$227^{\blacktriangle}$	$223^{\blacktriangle}$	$\underline{216}^{\blacktriangle}$	$\underline{203}$	$\underline{187}_{\sharp}$	$\underline{163}$	143	$145^{\blacktriangle}$	$\underline{106}^{\blacktriangle}$

P@10	10%	20%	30%	40%	50%	60%	70%	80%	90%
TCP	436	434	428	430	$\underline{432}$	$\underline{388}$	$\underline{338}$	$\underline{288}$	188
PRP	438	$\underline{442}$	$\underline{456}^{\blacktriangle}$	432	414	378	296	276	202
IP-t	436	440	$444_{\flat}$	442	422	$\underline{388}$	298	$\underline{288}$	$\underline{210}$
IP-u	$\underline{440}$	$\underline{442}$	$442_{\flat}$	$\underline{444}_{\sharp}$	424	$\underline{388}$	302	$\underline{288}$	202

Long Query (MAP/P@10 at 0%: 256/478)

MAP	10%	20%	30%	40%	50%	60%	70%	80%	90%
TCP	249	$\underline{239}$	$\underline{230}$	$\underline{209}$	$\underline{188}$	$\underline{166}$	136	103	064
PRP	$\underline{251}$	239	221	204	179	161	143	$\underline{147}^{\blacktriangle}$	$120^{\blacktriangle}$
IP-t	251	238	222	207	185	161	$\underline{143}$	$144^{\blacktriangle}$	$123^{\blacktriangle}$
IP-u	250	238	223	208	185	164	142	$141^{\blacktriangle}$	$\underline{124}^{\blacktriangle}$

P@10	10%	20%	30%	40%	50%	60%	70%	80%	90%
TCP	476	478	$\underline{480}$	$\underline{456}$	$\underline{464}$	$\underline{436}$	$\underline{376}$	$\underline{322}$	188
PRP	$\underline{490}$	486	472	440	$408^{\blacktriangledown}$	376	324	294	232
IP-t	478	$\underline{488}$	460	454	$410^{\blacktriangledown}$	388	344	300	$\underline{238}$
IP-u	482	484	462	452	$410^{\blacktriangledown}$	388	342	286	228

Table 4.2: The overall performance results on TREC-8. See Table 4.1 for the description of notation.

results on all the corpora and stresses more on performance differences. Statistical significance in this respect is assessed using two-tailed paired t-test for $p < 0.05$. We use superscripts ($\blacktriangle$ and $\blacktriangledown$) and subscripts ($\sharp$ and $\flat$) to highlight these entries.

The result shows that the performance for IP-based methods is generally comparable to that for PRP. No consistent pattern is observed across all settings to assess one method is better than the others. Significant difference in either MAP or P@10 between IP-based methods and PRP is detected for 10 out of 54 experimental runs, among which IP-based methods are shown superior to PRP in 8 runs (denoted as $\sharp$). PRP significantly outperforms only for short queries on TREC-8 at 30% and long queries on WT2G at 80% (denoted as $\flat$), but the latter result is inconsistent across performance measures.

Short Query (MAP/P@10 at 0%: 249/414)

MAP	10%	20%	30%	40%	50%	60%	70%	80%	90%
TCP	243	230	216	197	174	142	107	080	041
PRP	$\underline{254}^{\blacktriangle}$	$242^{\blacktriangle}$	$\underline{232}$	218	183	152	109	$\underline{094}$	$\underline{076}^{\blacktriangle}$
IP-t	$253^{\blacktriangle}$	$\underline{246}^{\blacktriangle}$	230	$223^{\blacktriangle}$	194	$\underline{158}$	$116_{\sharp}$	083	$075^{\blacktriangle}$
IP-u	$251^{\blacktriangle}$	$246^{\blacktriangle}$	231	$\underline{223}^{\blacktriangle}$	$\underline{197}$	151	$\underline{119}_{\sharp}$	083	$076^{\blacktriangle}$

P@10	10%	20%	30%	40%	50%	60%	70%	80%	90%
TCP	408	404	$\underline{402}$	384	364	$\underline{342}$	$\underline{302}$	$\underline{252}$	$\underline{174}$
PRP	$\underline{418}$	404	$\underline{402}$	380	$\underline{366}$	302	$232^{\blacktriangledown}$	$138^{\blacktriangledown}$	138
IP-t	416	$\underline{408}$	392	$\underline{394}$	360	$330_{\sharp}$	256	$140^{\blacktriangledown}$	$124^{\blacktriangledown}$
IP-u	414	$\underline{408}$	390	386	362	326	$260_{\sharp}$	$144^{\blacktriangledown}$	$128^{\blacktriangledown}$

Long Query (MAP/P@10 at 0%: 293/460)

MAP	10%	20%	30%	40%	50%	60%	70%	80%	90%
TCP	281	252	$\underline{237}$	206	182	147	110	085	044
PRP	275	247	222	202	173	153	115	$\underline{096}$	$\underline{082}^{\blacktriangle}$
IP-t	$\underline{283}_{\sharp}$	$256_{\sharp}$	224	$\underline{211}$	181	$\underline{158}$	119	089	$079^{\blacktriangle}$
IP-u	281	$\underline{257}_{\sharp}$	226	207	$\underline{184}$	152	$\underline{119}$	$088_{\flat}$	$079^{\blacktriangle}$

P@10	10%	20%	30%	40%	50%	60%	70%	80%	90%
TCP	444	420	$\underline{404}$	$\underline{394}$	$\underline{354}$	$\underline{332}$	$\underline{304}$	$\underline{240}$	$\underline{176}$
PRP	436	408	$\underline{404}$	368	334	314	$240^{\blacktriangledown}$	$154^{\blacktriangledown}$	168
IP-t	$\underline{450}$	$\underline{424}$	386	372	336	312	$248^{\blacktriangledown}$	$158^{\blacktriangledown}$	168
IP-u	$\underline{450}$	420	380	376	340	302	$256^{\blacktriangledown}$	$158^{\blacktriangledown}$	174

Table 4.3: The overall performance results on WT2G. See Table 4.1 for the description of notation.

It is interesting to note that TCP is generally doing slightly worse than the rest of methods in MAP but slightly better in P@10, which suggests that IP-based method favors more on recall. This trend is observed across different corpora and experimental settings, and is more amplified in the short query cases. Comparing the performance for IP-based methods with that for TCP in all 54 runs, we find that IP-based methods significantly outperform TCP in 14 (denoted as ▲), and TCP significantly outperforms IP-based methods in 6 (denoted as ▼). The case we have observed for long queries on WT2G at 90% prune ratio is difficult to interpret: TCP performs significantly worse in MAP but does better in P@10.

	TCP	PRP	IP-t	IP-u
TCP	–	0.332	0.665	0.661
PRP	0.332	–	0.282	0.281
IP-t	0.665	0.282	–	0.998
IP-u	0.661	0.281	0.998	–

Table 4.4: Correlation analysis for the decision measures on the LATimes corpus. The correlation is estimated using Pearson's product-moment correlation coefficient, weighted using term frequencies of index entries.

4.5.3 Correlation Analysis

Our experimental result gives rise to an interesting question that whether different pruning methods lead to different prioritization over index entries. To investigate the effect of pruning methods in this respect, we conducted a simple correlation analysis on the LATimes corpus. For each index entry, we retrieved the decision scores produced by all four algorithms and compiled them into a tuple. We collected totally 36,497,224 such tuples. For each pair of methods, we computed Pearson's product-moment correlation coefficient, weighted using term frequencies of index entries.

The result, which is summarized in Table 4.4, shows that the decision scores produced by two IP-based methods are strongly correlated (0.998). In this case, we conclude that uniform prior is more favorable than tangent approximation in real-world settings, since the former is easier to compute. IP-based methods also show medium correlation (0.661 and 0.665) with TCP, which is slightly stronger than that (0.332) with PRP. We want to point out here that, since the decision score used in TCP corresponds to BM25, IP-based scores leans more toward BM25 in terms of the effect on index entry prioritization. This can be useful in some other information retrieval applications.

4.6 Concluding Remarks

In this paper, we develop the notion of information preservation in the context of information retrieval, and use this idea to motivate a new decision criteria for static index pruning. In the experiments conducted on three different test corpora, the proposed method shows consistent, competitive performance to state-of-the-art methods. So far, there is only minor evidence to interpret the performance differences between the proposed approach and the reference methods for specific cases. We expect this to be made clear with further experimentation.

Our approach has a few advantages in terms of efficiency. First, term-centric pruning has an overhead in computing the cutting threshold for each term, since a sorting algorithm is involved to order the postings in terms of their impact values. Second, computation for the PRP measure depends on three different probability estimates, while the proposed IP-u measure (uniform prior) relies on only the query likelihood. This difference can be far more amplified in Web-scale settings.

There are many ways to extend this work. One possible direction that we have in mind is to combine weakly-correlated measures in static index pruning. Given the correlation analysis result, we believe that doing so is feasible and can be beneficial. Moreover, this study also provides an alternative viewpoint toward prioritization of index entries. Impact and uncertainty are intrinsically two different concepts, while in this very application our result somehow closes the gap in between. This connection may lead to a new postulation toward retrieval theory. We hope that our efforts will invite further investigation into these interesting issues.

Chapter 5

Discussion and Conclusion

5.1 Overview

Before concluding this work, we will loosely go over some technical issues regarding information preservation. Detailed treatment for these topics is beyond the scope of this study, while it points to a promising direction that further enhances our knowledge about this optimization strategy.

So far, we have studied several successful applications in probabilistic modeling and natural language processing empowered by information preservation. In the further analysis, we revisit these problems and focus more on the core attributes. In the following paragraph, we briefly summarize these problems.

Partition We want to find one *partitioner* in a restricted hypothesis space such that, when applied to data points, it produces the best approximation to some known, ideal partition. Feature selection is one such example. This type of learning is related to entropy minimization. Since the exact solution involves combinatorial choices, we usually solve this type of problem using iterative approximation methods.

Regression We search for one function in a constrained space that fits the observed data best. Usually the aim in this type of optimization is to minimize the sum of squared errors or to maximize predictive likelihood. This has been shown to be equal to minimum entropy, and efficient algorithms are already known in some typical examples such as linear regression, where the search space is restricted to linear functions.

Clustering This type of problem is related to unsupervised learning. It seeks to induce a partition over data points such that the intra-cluster cohesion is maximized, meaning that errors are minimized inside a cluster. In Chapter 2, we have shown that one special case of clustering, i.e., spectral clustering, is related to minimum entropy optimization.

Segmentation In a segmentation problem, we want to build a high-level representation over a sequence of tokens. One typical example is unsupervised word segmentation, in which we seek to learn a lexicon of words from text corpora. In Chapter 3, we have placed a constraint on the complexity of induced lexicon, and solved the problem by using a regularization method. This problem is only loosely related to minimum entropy optimization with our regularization assumption.

Degradation We seek to induced a degraded nonparametric predictive model, by reducing the amount of stored information. In Chapter 4, we have studied one special case of this type of problem, showing that it is related to entropy maximization. When combinatorial choices are involved, as in static index pruning, entropy maximization is usually solved using heuristic-based iterative methods.

Problem	Reference (θ_0)	$H_{\theta_0}(\cdot)$	Related Principles
Partition	Perfect partition	Low	MinEnt
Regression	Pointwise function	Low	MinEnt, MinErr, MaxLL
Clustering	Pointwise partition	Low	MinEnt, MinErr, MaxLL
Segmentation	Token representation	–	–
Degradation	Full model	High	MaxEnt
Histogram	Full model	High	MaxEnt

Table 5.1: A brief summary of problems that are solvable by using information preservation. Each row indicates one research problem. The second column indicates the reference hypothesis used in the corresponding information preservation framework. The third column shows relative degree of entropy (low/high) for each problem. Related principles are listed at the last column; abbreviations are used instead of full names. The principles include minimum entropy (MinEnt), maximum entropy (MaxEnt), minimum error (MinErr), and maximum likelihood (MaxLL).

5.2 General Discussion

5.2.1 Problem-Oriented Analysis

Table 5.1 briefly summarizes the solutions for these methods in the proposed framework. For each problem, we list the reference hypothesis, the relative degree of entropy of the reference hypothesis (with respect to that of the rest of hypotheses), and the related principles.

There are quite a few points that we are trying to make. The first one is that finding a usable reference hypothesis is easy. In the examples that we have shown, these hypotheses are either (i) ideal models, as in the partition problem, or (ii) data models, which can be trivially formed using the observed data. Case (i) aligns with our goal in learning the ideal models. Case (ii) is a lot more interesting because it covers many different purposes, including error minimization (as in regression and clustering), complexity regularization (as in segmentation), and uncertainty maximization (as in degradation and histogram.) For each of these problem, we can easily come up with a hypothesis that we need the subsequent optimization to approach. Since these criteria are designed to avoid information change in model fitting, it is straightforward to translate them to information preservation.

The second point is that information preservation is a more general concept than entropy maximization or minimization. This is exemplified by our approach to unsupervised word segmentation. In solving this problem, we make an assumption in the definition of compression steps that greatly simplifies the computation, i.e., replacing the occurrences of some subsequence all at once. This assumption leads to an asymptotic argument, stating that compression process increases vocabulary entropy, so as to justify our approximation method. Nevertheless, it does not mean information preservation problem can only be solved by reduction to entropy minimization. It is still possible to solve information preservation problem generally, though it can be far less efficient to do so, according to our limited understanding about the underlying numerical work.

These argument bring us to our last point that modeling the problem from an alternative angle sometimes reveals new insights. In clustering, we use a probability distribution that is not commonly seen in the literature. This choice is made not because it makes the problem easier but it simplifies the derivation. In fact, we have made this problem more difficult to solve than with usual approaches; as a result, we have uncovered a new regularization approach in our hunt for the evidence of preserved information. This may not be the case if we attempted only to reduce the problem to some form solvable by entropy minimization.

5.2.2 Relation to Principle of Maximum Cross-Entropy

Both information preservation and maximum cross-entropy stress on finding the probabilistic model that is closest to some reference distribution. They only differ in the definition of closeness. Thus, it is interesting to apply information preservation to the problems solvable by maximum cross-entropy, and vice versa. In doing so, we may gain deeper understanding in the true merits of both postulations.

One advantage of information preservation is that absolute change in entropy is straightforward to compute. In maximum cross-entropy, the equivalent “distance” term is expressed using Kullback-Leibler divergence, which imposes a stronger assumption over the two distributions to be compared. Let p and q denote two such distributions and let $X \in \mathcal{X}$ be the random variable of interest. The Kullback-Leibler divergence between p and q, denoted as $D(p||q)$, is not defined when there exists some $x \in \mathcal{X}$ such that $q(x) = 0$ but $p(x) > 0$. Entropy-based distance measures, such as ours, do not suffer from this issue.

5.2.3 Implication to Latent Variable Modeling

Part of our effort is devoted to extending the idea of information preservation to latent variable modeling. In one of the special cases, where the dependence relation between observed and latent variables appears to be deterministic, we find that conventional inference algorithms, such as expectation-maximization (EM) or Markov Chain Monte Carlo (MCMC), may fail to work. These methods rely on exploitation of the correlation between observed and latent variables to find the best latent model, while in the dependence case the correlation may no longer be useful in guiding the search.

To deal with this problem, we develop an inference method, called *utility-bias tradeoff*, based on the iterative approximation techniques that we have used in various information preservation problems. The proposed framework complements the conventional approaches. Specifically, it relies on two quantities, utility and bias, to guide the search for the best latent model. Since this result is incomplete and does not entirely fit into the big picture of this thesis work, we move the details out of the main text. Interested readers are referred to Appendix A.3.

5.3 Concluding Remarks

The foremost contribution of this thesis is the development of information preservation. This concept provides an unified way for modeling different optimization strategies. The proposed principle can be applied to some classic learning problems in probabilistic modeling, such as regression and cluster analysis, and two other natural language applications, such as unsupervised word segmentation and static index pruning. The latter case demonstrates that our method is suitable for solving complicated cases where other mathematical principles do not fit.

Our approach provides a common ground for relating various optimization principles, such as maximum and minimum entropy methods. In our framework, the optimization process is directed toward approximation for a reference hypothesis, an essential concept that may have been implicitly implied in conventional methods. Making this concept explicit improves our understanding about how the entropy-based optimization criteria work. It also resolves the incompatibility issue between entropy maximization and minimization, since in the view of information preservation, the two principles differ only in the target to approximate.

Our experimental study in unsupervised word segmentation and static index pruning has created new methodologies toward these problems. For unsupervised word segmentation, our regularization approach has significantly boosted the segmentation accuracy of an ordinary compression method, and achieved comparable performance to several state-of-the-art methods in terms of efficiency and effectiveness. For static index pruning, our approach suggests a new way of prioritizing index entries. The proposed information-based measure has achieved state-of-the-art performance in this task, and it has done so more efficiently than the other methods.

Many interesting issues have been uncovered and remained open in this thesis work. In the cluster analysis problem, our approach leads to a new regularization method that has not been discussed before; estimation of the normalization factor of the error distribution is also of theoretical value. Similarly, a numerical approximation specifically-tailored for information preservation will have huge impact to most of the problems that we have discussed in the thesis work. We also expect seeing more applications of information preservation to maximum cross-entropy problems, since the former may serve as an economical approximation to the latter. We believe that these directions will lead to fruitful results.

There are some related natural language problems that we look forward to applying information preservation to, such as text summarization (as a sentence pruning problem) and named-entity recognition (as a specialized segmentation problem). Success in these essential tasks will broaden the impact of this approach. Moreover, we expect this thesis work to create new conversations and studies as to the mathematical principles that underpin probabilistic methods. Deeper understanding about these principles may produce new applications or new methodologies towards probabilistic modeling, and eventually lead to breakthrough in natural language processing.

Bibliography

Ismail S. Altingovde, Rifat Ozcan, and Özgür Ulusoy. A practitioner's guide for static index pruning. In Mohand Boughanem, Catherine Berrut, Josiane Mothe, and Chantal Soule-Dupuy, editors, *Advances in Information Retrieval*, volume 5478 of *Lecture Notes in Computer Science*, chapter 65, pages 675–679. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-00957-0. doi: 10.1007/978-3-642-00958-7_65. URL http://dx.doi.org/10.1007/978-3-642-00958-7_65.

Ismail S. Altingovde, Rifat Ozcan, and Özgür Ulusoy. Static index pruning in web search engines: Combining term and document popularities with query views. *ACM Transactions on Information Systems*, 30(1), March 2012. ISSN 1046-8188. doi: 10.1145/2094072.2094074. URL http://dx.doi.org/10.1145/2094072.2094074.

Shlomo Argamon, Navot Akiva, Amihood Amir, and Oren Kapah. Efficient unsupervised recursive word segmentation using minimum description length. In *Proceedings of the 20th international conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics. doi: 10.3115/1220355.1220507. URL http://dx.doi.org/10.3115/1220355.1220507.

Nan Bernstein-Ratner. The phonology of parent child speech. *Children's language*, 6:159–174, 1987.

Roi Blanco and Álvaro Barreiro. Static pruning of terms in inverted files. In

Giambattista Amati, Claudio Carpineto, and Giovanni Romano, editors, *Advances in Information Retrieval*, volume 4425 of *Lecture Notes in Computer Science*, chapter 9, pages 64–75. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 978-3-540-71494-1. doi: 10.1007/978-3-540-71496-5_9. URL http://dx.doi.org/10.1007/978-3-540-71496-5_9.

Roi Blanco and Alvaro Barreiro. Probabilistic static pruning of inverted files. *ACM Transactions on Information Systems*, 28(1), January 2010. ISSN 1046-8188. doi: 10.1145/1658377.1658378. URL http://dx.doi.org/10.1145/1658377.1658378.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003. URL http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf.

Michael R. Brent and Timothy A. Cartwright. Distributional regularity and phonotactic constraints are useful for segmentation. In *Cognition*, pages 93–125, 1996. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.33.1129.

Stefan Büttcher and Charles L. A. Clarke. A document-centric approach to static index pruning in text retrieval systems. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, CIKM '06, pages 182–189, New York, NY, USA, 2006. ACM. ISBN 1-59593-433-2. doi: 10.1145/1183614.1183644. URL http://dx.doi.org/10.1145/1183614.1183644.

David Carmel, Doron Cohen, Ronald Fagin, Eitan Farchi, Michael Herscovici, Yoelle S. Maarek, and Aya Soffer. Static index pruning for information retrieval systems. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 43–50, New York, NY, USA, 2001. ACM. ISBN 1-58113-331-6. doi: 10.1145/383952.383958. URL http://dx.doi.org/10.1145/383952.383958.

Jing-Shin Chang and Keh-Yih Su. An unsupervised iterative method for chinese new lexicon extraction. In *International Journal of Computational Linguistics & Chinese Language Processing*, pages 97–148, 1997. URL http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.26.6659&rep=rep1&type=pdf.

Lee-Feng Chien. PAT-tree-based keyword extraction for chinese information retrieval. *SIGIR Forum*, 31(SI):50–58, 1997. ISSN 0163-5840. doi: 10.1145/258525.258534. URL http://dx.doi.org/10.1145/258525.258534.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. ISSN 00359246. doi: 10.2307/2984875. URL http://web.mit.edu/6.435/www/Dempster77.pdf.

Thomas Emerson. The second international chinese word segmentation bakeoff. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, volume 133. Jeju Island, Korea, 2005.

Haodi Feng, Kang Chen, Xiaotie Deng, and Weimin Zheng. Accessor variety criteria for chinese word extraction. *Comput. Linguist.*, 30:75–93, March 2004. ISSN 0891-2017. doi: 10.1162/089120104773633394. URL http://dx.doi.org/10.1162/089120104773633394.

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, November 1984. doi: 10.1080/02664769300000058. URL http://dx.doi.org/10.1080/02664769300000058.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 673–680, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. doi: 10.3115/

1220175.1220260. URL http://dx.doi.org/10.3115/1220175.1220260.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. A bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1):21–54, July 2009. ISSN 00100277. doi: 10.1016/j.cognition.2009.03.008. URL http://dx.doi.org/10.1016/j.cognition.2009.03.008.

Zellig S. Harris. From phoneme to morpheme. *Language*, 31(2):190–222, 1955. ISSN 00978507. doi: 10.2307/411036. URL http://dx.doi.org/10.2307/411036.

Daniel Hewlett and Paul Cohen. Bootstrap voting experts. In *Proceedings of the 21st international jont conference on Artifical intelligence*, IJCAI'09, pages 1071–1076, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc. URL http://portal.acm.org/citation.cfm?id=1661616.

Daniel Hewlett and Paul Cohen. Fully unsupervised word segmentation with BVE and MDL. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, pages 540–545, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-88-6. URL http://portal.acm.org/citation.cfm?id=2002843.

Jin H. Huang and David Powers. Chinese word segmentation based on contextual entropy. In *Proceedings of the 17th Asian Pacific Conference on Language, Information and Computation*, pages 152–158. Citeseer, 2003. URL http://aclweb.org/anthology/Y/Y03/Y03-1017.pdf.

E. T. Jaynes. Information theory and statistical mechanics. *Physical Review Online Archive (Prola)*, 106(4):620–630, May 1957a. doi: 10.1103/PhysRev.106.620. URL http://dx.doi.org/10.1103/PhysRev.106.620.

E. T. Jaynes. Information theory and statistical mechanics. II. *Physical Review Online Archive (Prola)*, 108(2):171–190, October 1957b. doi: 10.1103/PhysRev.108.171. URL http://dx.doi.org/10.1103/PhysRev.108.171.

Zhihui Jin and Kumiko T. Ishii. Unsupervised segmentation of chinese text by use of branching entropy. In *Proceedings of the COLING/ACL on Main conference poster sessions*, COLING-ACL '06, pages 428–435, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. URL `http://portal.acm.org/citation.cfm?id=1273129`.

Mark Johnson and Sharon Goldwater. Improving nonparameteric bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 317–325, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-41-1. URL `http://portal.acm.org/citation.cfm?id=1620800`.

Chunyu Kit and Yorick Wilks. Unsupervised learning of word boundary with description length gain. In *CoNLL-99*, pages 1–6, Bergen, Norway, 1999.

Soloman Kullback. *Information Theory and Statistics*. Wiley, New York, 1959.

Gina-Anne Levow. The third international chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, volume 117. Sydney: July, 2006.

S. Lloyd. Least squares quantization in PCM. *Information Theory, IEEE Transactions on*, 28(2):129–137, March 1982. ISSN 0018-9448. doi: 10.1109/TIT.1982.1056489. URL `http://dx.doi.org/10.1109/TIT.1982.1056489`.

J. B. Macqueen. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

Brian MacWhinney and Catherine Snow. The child language data exchange system:

an update. *Journal of child language*, 17(2):457–472, June 1990. ISSN 0305-0009. URL http://view.ncbi.nlm.nih.gov/pubmed/2380278.

Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. Bayesian unsupervised word segmentation with nested Pitman-Yor language modeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 100–108, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-45-9. URL http://portal.acm.org/citation.cfm?id=1687894.

Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, September 1978. ISSN 00051098. doi: 10.1016/0005-1098(78)90005-5. URL http://dx.doi.org/10.1016/0005-1098(78)90005-5.

Stephen Robertson. The probability ranking principle in IR. In Karen S. Jones and Peter Willett, editors, *Reading in Information Retrieval*, chapter The probability ranking principle in IR, pages 281–286. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997. ISBN 1-55860-454-5. URL http://portal.acm.org/citation.cfm?id=275701.

Richard W. Sproat and Chilin Shih. A statistical method for finding word boundaries in Chinese text. *Computer Processing of Chinese and Oriental Languages*, 4(4): 336–351, 1990.

Maosong Sun, Dayang Shen, and Benjamin K. Thou. Chinese word segmentation without using lexicon and hand-crafted training data. In *Proceedings of the 17th international conference on Computational linguistics - Volume 2*, COLING '98, pages 1265–1271, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics. doi: 10.3115/980432.980775. URL http://dx.doi.org/10.3115/980432.980775.

Kumiko Tanaka-Ishii. Entropy as an indicator of context boundaries: An experiment using a web search engine. In Robert Dale, Kam-Fai Wong, Jian Su, and Oi Kwong, editors, *Natural Language Processing – IJCNLP 2005*, volume 3651 of *Lecture Notes in Computer Science*, chapter 9, pages 93–105. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-29172-5. doi: 10.1007/11562214_9. URL `http://dx.doi.org/10.1007/11562214_9`.

Hua Yu. Unsupervised word induction using MDL criterion. In *Proceedings of the International Symposium of Chinese Spoken Language Processing*, Beijin, China, 2000.

Hai Zhao and Chunyu Kit. An empirical comparison of goodness measures for unsupervised chinese word segmentation with a unified framework. In *The Third International Joint Conference on Natural Language Processing (IJCNLP-2008)*, 2008. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.135.6154`.

Lei Zheng and Ingemar J. Cox. Entropy-Based static index pruning. In Mohand Boughanem, Catherine Berrut, Josiane Mothe, and Chantal Soule-Dupuy, editors, *Advances in Information Retrieval*, volume 5478 of *Lecture Notes in Computer Science*, chapter 72, pages 713–718. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-00957-0. doi: 10.1007/978-3-642-00958-7_72. URL `http://dx.doi.org/10.1007/978-3-642-00958-7_72`.

Valentin Zhikov, Hiroya Takamura, and Manabu Okumura. An efficient algorithm for unsupervised word segmentation with branching entropy and MDL. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 832–842, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL `http://portal.acm.org/citation.cfm?id=1870739`.

Appendix A

Supplementary Details

A.1 Representation System and Entropy

We define a *representation system* for a set of concepts. A representation system is defined as a 3-tuple $R = (C, A, g)$, where C denotes the set of concepts, A denotes the alphabet, and g denotes the ruleset. The definitions for the three components are given below.

- Concepts: Let C be the set of *concepts*, in which each concept refers to a semantic units that we use to describe concrete ideas. There is no direct connection between a concept and any language construct in written or spoken form. We further assume that any form of higher-level knowledge can be expressed as a sequence of concepts.

- Alphabet: We also need to define a symbol system to carry information, for that exchange of knowledge takes place in a concrete form, e.g., as a piece of text or speech. This set of symbols is called the *alphabet*, which is denoted as A. The alphabet has to be finite but not necessarily closed. Any concept, in order to be understood, has to be represented as a non-empty sequence of symbols in A. Usually we call such a sequence a *word*.

- Ruleset: We assume that an set of rules g exists so as to maps a concept to a word. In fact, g acts as an injective function from C to A^+; we choose the ruleset definition just to keep this notion flexible. The definition about g is shared among all the language users. To communicate ideas, therefore, one employs g to express concepts in her mind as a passage (i.e., a sequence of words) and passes it out; the recipient then recovers the sequence of concepts by consulting the g to interpret the passage. Note that the interpretation to a passage may not be unique when more than one concepts can map to the same word.

For any set of concepts C, one forms a *trivial representation system* R_0 by letting the alphabet be of the as same size as that of C and the ruleset as an identity mapping. In mathematical terms, $R_0 = (C, C, g_0)$ where $g_0(c) = c$ for all $c \in C$. In other words, this representation system corresponds to a language system, in which the number of symbols is as many as that of concepts.

On one hand, this representation system is efficient, because it takes only one symbol to represent any concept. On the other hand, this system is also very verbose, because its alphabet is unbearably large.

Recall that the empirical entropy for X with respect to a sequence of N observations is defined as:

$$\begin{aligned}\tilde{H}(X) &= -\sum_{x \in \chi} \tilde{p}(x) \log \tilde{p}(x) \\ &= -\sum_{x \in \chi} \frac{n_x}{N} \log \frac{n_x}{N} \\ &= \log N - \frac{1}{N} \sum_{x \in \chi} n_x \log n_x.\end{aligned} \tag{A.1}$$

Lemma 1. *Let X be a random variable over a set of tokens χ. Suppose that we observe a sequence of tokens drawn from this distribution and n_x be the number of occurrences for any $x \in \chi$ in the sequence. For any $z \in \chi$ in a sequence of N observations that is sufficiently long such that n_z/N approaches 0, the empirical* *entropy for X decreases if we replace each occurrence of z in $\mathbf{x}$ with two tokens a and b where $a, b \in \chi$.*

Proof. Let $\mathbf{x}$ denote the original sequence and $\mathbf{x}'$ the new sequence after the replacement. We write the difference between the empirical entropy for $\mathbf{x}'$ and that for $\mathbf{x}$ as follows.

$$\begin{aligned}
&\tilde{H}(X)_{\mathbf{x}'} - \tilde{H}(X)_{\mathbf{x}} = \\
&\quad \log(1 + \frac{n_z}{N}) \\
&\quad + \frac{n_z}{N(N+n_z)} \sum_{x \in \chi} n_x \log n_x \\
&\quad + \frac{1}{N+n_z} \bigg[n_z \log n_z + n_a \log n_a + n_b \log n_b \bigg] \\
&\quad - \frac{1}{N+n_z} \bigg[(n_z + n_a) \log(n_z + n_a) - (n_z + n_b) \log(n_z + n_b) \bigg].
\end{aligned} \tag{A.2}$$

We show that the last two terms converge to 0 less rapidly than the first two by multiplying both sides by $N + f(z)$ and pass n_z/N to 0. As a result, the right hand side diminishes except the last two terms; now the equation reads:

$$\begin{aligned}
&\lim_{n_z/N \to 0} (N + n_z)(\tilde{H}(X)_{\mathbf{x}'} - \tilde{H}(X)_{\mathbf{x}}) = \\
&\quad \bigg[n_z \log n_z + n_a \log n_a - (n_z + n_a) \log(n_z + n_a) \bigg] \\
&\quad + \bigg[n_b \log n_b - (n_z + n_b) \log(n_z + n_b) \bigg].
\end{aligned} \tag{A.3}$$

The difference is obviously less than 0. □

A.2 Rewrite-Update Procedure

The proposed algorithm relies on an efficient implementation in Steps 2a and 2b to achieve satisfactory performance. To explore this issue, we make a simplifying assumption here that all the translation rule considered is of the form:

$$w \to xy,$$

where $w \in \mathcal{W}$ and $x, y \in \mathcal{C}$. Note that this assumption is made for ease of discussion; it is possible to tailor the aforementioned algorithm in a more general respect.

In the following paragraphs, we motivate the need for developing a rewrite-update procedure to further enhance the performance:

- In order to solve the optimization problem in Step 2b, we need to iterate through all the possible bigram sequences, gather required statistical quantities, and compute the objective value for each sequence. Specifically, to compute entropy we need access to unigram and bigram frequencies; these quantities, however, do not stay constant throughout iterations.

- To alter the sequence in Step 2b, we need faster access to reach the desired positions in which the proposal xy occur. An usual solution is to employ an indexing structure to book-keep the set of positions that a specific bigram occurs in the text stream. In this case, we need something more than a static indexing structure for doing this job, since in each iteration new tokens (and new bigrams, accordingly) are introduced into the sequence.

It is immediately clear that the major challenge resides in data management. Static data store does not fit into this scenario since tiny changes are introduced and applied to the sequence in every iteration, and to reflect that change back to the data becomes the key to computational efficiency. In the first place, it may seem that a two-pass scan through the sequence is inevitable. We notice that, however, the number of changes to the statistical quantities is linear in the number of occurrences of the subsequence xy. In other words, only a limited number of tokens and bigrams are affected by the change we introduce in Step 2b.

Consider the following snippets of the sequence. Let a denote the token that precedes

x and b the token that follows y. The original sequence is as:

$$\ldots \quad a \quad \underline{x \quad y} \quad b \quad \ldots$$

In Step 2b, we introduce a new token z to replace this occurrence of bigram xy. The resulting sequence becomes:

$$\ldots \quad a \quad \underline{z} \quad b \quad \ldots$$

We can then divide the changes needed to reflect this change into the following four classes.

1. Decrease the unigram frequencies for x and y by 1, respectively. Remove the corresponding positions in the posting lists for x and y.

2. Decrease the bigram frequencies for ax, xy, and yb by 1, respectively. Revise the corresponding posting lists for these bigrams as well.

3. Increase the unigram frequency for z by 1 and add add this position to the posting list for z.

4. Increase the bigram frequency for az and zb by 1 and revise the corresponding posting lists for those bigrams.

Based on this observation, we can update the affected statistical quantities efficiently in each iteration. To facilitate this idea, we need to set up additional data structure when the algorithm starts. These extra initialization steps are detailed as follows:

- We store the input sequence $\mathbf{c}$ in an array of fixed-size integers; this array is used in forthcoming iterations to keep track of the latest token sequence.

- We create posting lists for all the unigrams in the first place. The posting lists also hold the unigram frequencies.

- We keep track of bigram counts with a hashtable.

The last item seems a bit unusual because we do not use posting lists to keep track of the bigram positions. Note that the bigram positions can be recovered by using the unigram posting lists and the sequence array alone. We reveal more details in the following paragraphs.

In the first half of each iteration, we iterate through bigram set and solve the optimization problem, as depicted in Step 2a. This is as simple as to look in the hashtable and iterate through each entry. For each bigram xy, the objective depends only on its bigram frequency and the unigram frequencies for x and for y; it takes $O(1)$ time to look up these quantities from the hashtable and from the unigram posting lists. The total amount of time to determine the proposal is thus linear in the number of possible bigrams in the current sequence.

In the second half, we replace every occurrence for the chosen bigram xy with a new token z and update the statistical quantities accordingly. In the original algorithm, this task is described as Step 2b. Since this procedure is slightly more complicated, we split the replacement-update job into three sub-steps for ease of discussion: decrement, update, and increment. The sole purpose for designing the algorithm this way is to simplify the logic. Without this arrangement, certain bigram combinations, such as the first and the second token being the same (e.g., xx), can be tricky to deal with.

We use an additional array of booleans to keep track of the change we make to the sequence. Conceptually, one can think of it as the *change flags* associated to all the tokens the input sequence.

The first thing we do is to gain access to the posting list for xy. Since we do not keep track of bigram postings, we recover this list, denoted as PL_{xy}, from the posting lists for x and for y, denoted as PL_x and PL_y, respectively. Using a merge-like algorithm, the time complexity is linear to the size of PL_x and PL_y. Once this is done, we proceed to the following sub-steps, which are detailed in the following paragraphs.

- Decrement: We iterate through PL_{xy} and visit each occurrence of bigram xy sequentially. Initially, all the change flags are set to 0. Let the position for each occurrence of bigram xy be p_x and p_y, we scan backward and forward in the text sequence to find the preceding and the following token position p_a and p_b. Let the token at position p_a be a and that at position p_b be b.

  For each of the three bigrams ax, xy, and yb, we check the change flags in their corresponding positions to see if we have already decreased the count. For any bigram, if both of its two positions have the change flags flipped, we proceed to the next; otherwise, we decrease its bigram counts by 1 and flip the change flags to 1 for both tokens.

- Update: We iterate through PL_{xy} and visit each occurrence accordingly. For each occurrence of bigram xy, we check and see if the tokens at the corresponding positions has been replaced. If not, we go ahead and replace the two tokens xy with $z0$. Note that 0 is the special padding symbol that we use to keep the array organized without insertion/deletion. This is illustrated as follows:

  $$\begin{array}{c} \cdots - \alpha - x - y - \beta - \cdots \\ \cdots - \alpha - z - 0 - \beta - \cdots \end{array}$$

  Once the tokens are successfully replaced, we remove the position for token x from PL_x and the position for token y from PL_y. We create a new posting list for z if necessary, and then add the new position for token z into its posting list. The unigram frequencies for x, y, and z are updated accordingly.

- Increment: We iterate through PL_z and visit each occurrence of unigram z sequentially. Again, all the change flags are set to 0 initially. We follow an analogous approach as in the decrement step. For each occurrence of unigram z, we find the preceding and the following token positions p_a and p_b by scanning backward and forward in the text sequence. Let the token at position p_a be a

and that at position p_b be b.

For each of the two bigrams az and zb, we check if it has been processed by looking at the corresponding change flags. If both flags for a bigram are flipped, we proceed to the next; otherwise, we increase its bigram count by 1 and flip both change flags.

The overall time complexity for this algorithm is $O(TN)$, where T is the number of iterations and N is the length of the input sequence.

A.3 Utility-Bias Tradeoff

A.3.1 Motivation

Latent variable modeling is a statistical technique that we use to model the latent information behind the data we observe. Such an application for modeling latent data is central to various natural language tasks, such as topic modeling (Blei et al., 2003) or data clustering (Macqueen, 1967; Lloyd, 1982), in which the information about the subject of interest is not directly observable at the surface level.

To introduce this concept, consider that we have a series of observations $X_1, X_2, \ldots, X_N$ drawn from a stochastic process $X \in \mathcal{X}$, and, with good reasons, we decide to model another related but unobserved stochastic process $Z \in \mathcal{Z}$. Since we do not have access to the latent data, it is not possible to directly specify a model for Z.

The idea of latent variable modeling is to specify an underlying generative structure for X and for Z, and then establish an inference scheme for Z based on exploitation of the structure. Formally speaking, the generative structure includes (i) the definitions for $\mathcal{X}$ and $\mathcal{Z}$, and (ii) the families of density functions for Z and for $X|Z$ (or the other way around, depending on how the two are related).

Let $O = (X_1, X_2, \ldots, X_N)$ denote the observed data and $L = (Z_1, Z_2, \ldots)$ denote the latent information. To infer L, we usually apply an iterative algorithm, such as expectation-maximization (EM) (Dempster et al., 1977) or Gibbs sampling (Geman and Geman, 1984), to induce the latent model based on maximum likelihood principle. This is to say, inference is done through search for a model θ that maximizes the joint likelihood $\Pr(O, L)$. We summarize this concept in the following optimization problem.

$$\begin{array}{ll} \text{maximize} & p(O, L|\theta) \\ \text{subject to} & L \in \mathcal{Z}^*, \theta \in \Theta \end{array} \tag{A.4}$$

For ease of computation, dependence between Z and X is usually assumed in order to dissect the joint likelihood, but sometimes we may fail to establish the generative argument, because it is not always so obvious to relate both variables probabilistically.

Here, we want to discuss an interesting case that the dependence between Z and X is in fact deterministic. By *deterministic dependence*, we mean that the observed and the latent variables are related in one of the following two ways:

1. Observations for X is functionally dependent on that of Z, i.e., $O = f(L)$ for some deterministic function $f : \mathcal{Z}^* \to \mathcal{X}^*$; or
2. Observations for Z is functionally dependent on that of X, i.e., $L = g(O)$ for some deterministic function $g : \mathcal{X}^* \to \mathcal{Z}^*$.

We argue that, in this case, the latent model derived by maximizing the joint likelihood is suboptimal, since, in either formulation, one set of variables fully depend on the other and the joint likelihood $\Pr(O, L)$ thus falls back to a more primitive form $\Pr(O)$ or $\Pr(L)$. This is usually not what we want from latent modeling, and the inference algorithms, which rely on exploitation of the underlying correlation to induce the right model, do not seem to work well in this case. As a consequence, inference needs to be approached using other strategies.

This is the scenario that we use to motivate the utility-bias tradeoff framework. In the following development, we demonstrate how to transform the inference problem shown in Equation (A.4) for the deterministic dependence case into another one that does not involve joint data likelihood maximization.

A.3.2 Perturbation and Utility-Bias Tradeoff

The inference strategy that we propose to solve the deterministic dependence cases is called *utility-bias tradeoff.* The idea is to associate two quantities, *utility* and *bias*, to every possible combination (O, L), and form a multi-criterion problem with respect to all the possible L, in which (i) utility is maximized, and (ii) bias is minimized. This approach is formalized in the following equation:

$$\begin{array}{ll} \text{maximize} & (\text{utility}(O, L), -\text{bias}(O, L)) \\ \text{subject to} & L \in \mathcal{Z}^* \end{array} \tag{A.5}$$

The descriptions about utility and bias are given in the following paragraphs:

- Utility indicates the value that we gain by constructing L with respect to the observations O. It is designed in nature to be maximized, and it is suggested to define this quantity at the application level, based on our preference over all the possible outcomes for the latent variables[1].

- Bias indicates the degree to which the constructed latent model for Z deviates from the true model for X. Intuitively speaking, it reflects the information gained or lost in construction of the latent model. Generally, we define this quantity as $|H(Z) - H(X)|$, where $H(Z)$ and $H(X)$ indicate the entropy rates for the stochastic processes Z and X, respectively. Note that the entropy rates for Z and for X are estimated on top of L and O, respectively.

We notice that, when the two stochastic processes Z ans X are related to each other

[1] From the description, it may seem that utility is somehow related to the prior for Z. We argue that utility shall be defined in a less restrictive sense, since the knowledge required for defining utility does not necessarily lead to a full construction of a probability distribution.

via a deterministic function, it suffices to optimize over a hypothesis space $\mathcal{H}$, each element of which is defined as a function $h : \mathcal{X}^* \to \mathcal{Z}^*$. This technique is called *perturbation*, by which we mean that we perturb the observations O with a function h to derive the latent information L.

Note that there is an additional requirement for applying perturbation to the case where X is functionally dependent on Z: every function h in the hypothesis space $\mathcal{H}$ has to be a bijective function, for which an inverse h^{-1} is defined. With this amendment, optimizing with respect to $\mathcal{H}$ is equivalent to that with respect to $\mathcal{H}^{-1}$, thereby satisfying the aforementioned dependence assumption. This is a necessary adjustment for theoretical soundness, while implementing this amendment also has made inference in this case more difficult than in the other.

A general utility-bias recipe for the deterministic dependence case is written as:

$$\begin{array}{ll} \text{maximize} & (\text{utility}(O, h), -\text{bias}(O, h)) \\ \text{subject to} & h \in \mathcal{H} \end{array} \tag{A.6}$$

We apply *scalarization* to solve this optimization problem. Scalarization transforms a k-dimensional objective $g = (g_1, g_2, \ldots, g_k)^T$ defined in a multi-criterion problem into a scalar form. To do that, we specify a *tradeoff parameter* $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_k)^T$ such that $\sum \lambda_i = 1$ and define the new objective g' as the dot product of λ and g, i.e., $g' = \lambda^T g$. Since in our case the objective is 2-dimensional, it suffices to parametrize the tradeoff using only one scalar $0 \leq \lambda \leq 1$.

Putting it all together, we write out the scalarized version of the utility-bias tradeoff problem as follows:

$$\begin{array}{ll} \text{maximize} & \lambda \cdot \text{utility}(O, h) - (1 - \lambda) \cdot \text{bias}(O, h)) \\ \text{subject to} & h \in \mathcal{H} \end{array} \tag{A.7}$$

Finally, we solve the original problem by search for the corresponding Pareto-optimal solution with respect to λ in the scalarized formulation.

A.3.3 Applications

In this subsection, we briefly sketch two general applications of utility-bias tradeoff. These examples are tailored to cover both deterministic dependence cases as discussed in the previous subsections.

Decoding Problem Consider that we communicate messages on a public channel. In a decoding problem, we receive an encoded message $r \in \mathcal{X}^*$, which we believe is a transcribed version from a non-encoded, latent message $s \in \mathcal{Z}^*$; we want to to induce s and the corresponding translation lexicon $l : \mathcal{Z}^* \to \mathcal{X}^*$ that satisfies $l(s) = r$ based on the observed message r.

Based on the description, there is a straightforward utility-bias construction. The key to successfully develop a solution is, however, a clever design of the utility. In fact, there are more than one way to do it, and we generally rely on application-specific heuristics to develop this quantity.

This problem is closely related to natural language processing tasks, such as unsupervised word segmentation or topic modeling. Note that the latent message s in this example is also referred to as *concepts* or *latent semantics* in the literature.

Model Degeneration Problem Consider that we have a nonparametric model for X built upon a series of observations $X_1, X_2, \ldots, X_N$, which takes a considerable amount of disk space to store. In a model degeneration problem, we seek to selectively discard observations, in light of reducing the model size while retaining model predictability as much as possible. Obviously, there is a tradeoff between the amount of information preserved and the amount of disk space spent, and we seek to balance the both.

Let r denotes the observations $X_1, X_2, \ldots, X_N$. We want to induce s, which is a subsequence of r, and a corresponding degeneration function $d : \mathcal{X}^* \to \mathcal{X}^*$ such that $d(r) = s$. To mathematically motivate "degeneration", we requires that, for any sequence $a \in \mathcal{X}^*$, $|d(a)| \le |a|$. Since one of our goals is to reduce the model size, it seems reasonable to relate the utility to the number of observations discarded with respect to r and d, as in:

$$\text{utility}(r, d) = |r| - |d(r)|. \tag{A.8}$$

A.3.4 Example: Unsupervised Word Segmentation

For any language of concern, let $\mathcal{C}$ denote the set of tokens or characters, i.e., the alphabet, and $\mathcal{W}$ denote the set of words. Generally, every $w \in \mathcal{W}$ is a string over $\mathcal{C}$. Consider that we have observed a set of utterances $\{O_1, O_2, \ldots, O_M\}$ in this language, in which each utterance O_k is a string over $\mathcal{C}$. For simplicity, we write $O = O_1 O_2 \ldots O_M$ as a concatenation of all the utterances by assuming that each utterance is separated from each other using some delimiting token.

Suppose that O as a sequence of observations $(C_1, C_2, \ldots, C_N)$ drawn from a stochastic process $C \in \mathcal{C}$. In unsupervised word segmentation, we seek to induce a sequence of words $(W_1, W_2, \ldots, W_L)$, in which each W_j is a draw from a latent stochastic process $W \in \mathcal{W}$. In this respect, unsupervised word segmentation is connected to the decoding problem discussed in Section A.3.3 and therefore has a corresponding utility-bias solution. In the following paragraphs, we motivates the definitions for a utility-bias construction to unsupervised word segmentation.

Hypothesis An ideal construction of the hypothesis space should cover all the possible ways to segment the observations O, but practically it is infeasible to implement this strategy. Instead, we propose defining the hypothesis space $\mathcal{H}$ as a set of *ordered rulesets*:

$$\mathcal{H} = \{h_R \mid R \text{ is an ordered ruleset}\}, \tag{A.9}$$

in which each hypothesis $h_R : \mathcal{C}^* \to \mathcal{W}*$ with respect to some ordered ruleset R is a function that reduces a sequence of words from a sequence of tokens by applying each translation rule in R in order.

An ordered ruleset R is a sequence of translation rules. Generally, every translation rule in R takes the form

$$w \to c_1c_2c_3\ldots c_n$$

for some n, where $w \in \mathcal{W}$ represents a word and each $c_i \in \mathcal{C}$ represents a token. Applying this translation rule to a string over $\mathcal{C}^*$ is equivalent to replacing all the occurrences of $c_1c_2\ldots c_n$ in the string to a corresponding word w.

In this light, a hypothesis h_R is well-defined since we requires that the translation rules in R be applied in order. For completeness, any token c in the string that is not covered by any rule during the reduction is automatically reduced afterward by using an implicit rule $c \to c$, since c can also be a member of $\mathcal{W}$.

Utility We define the utility in terms of an estimate g for the *lexical cohesion* that we gain by generalizing O into $h(O)$. In linguistics, cohesion is the connection that puts a piece of text together, one broader sense of which is the meaning carried by the text. When this connection is made in the lexical context, it refers to repetitive uses or collocation of a text. It seems reasonable here, in this light, to assume highly-cohesive subsequences are more likely to be true words.

The lexical cohesion is usually defined with respect to an entire word type $w \in \mathcal{W}$ instead of individual occurrences of the word type. In the word segmentation problem, we consider the following three kinds of estimates for lexical cohesion for any word type w:

1. Frequency, by which we define $g(w) = n_w \times (|w| - 1)$ where n_w denotes the

number of occurrences for w and $|w|$ denotes the number of tokens in w.

2. Pointwise mutual information, by which we define $g(w)$ as the pointwise mutual information for w.

3. Branching entropy, by which we define $g(w)$ as the branching entropy for w.

To motivate the definition for the utility using some lexical cohesion g, it suffices to consider only on the unique word types in the translated sequence. In this respect, we define the utility with respect to some hypothesis h and O as:

$$\text{utility}(h, O) = \sum_{w \in \text{WT}(h(O))} g(w), \tag{A.10}$$

where $\text{WT}(h(O))$ denotes the unique words in $h(O)$.

Bias The bias is defined as the absolute value of the difference between the entropy rates for C and for W, as in:

$$\text{bias}(h, O) = |\tilde{H}_{h(O)}(W) - \tilde{H}_O(C)|, \tag{A.11}$$

where $\tilde{H}_{h(O)}(W)$ and $\tilde{H}_O(C)$ denote the empirical entropy rate for W, estimated on $h(O)$, and the empirical entropy rate for C, estimated on O, respectively.

This definition attempts to quantify the change on the model predictability after the perturbation, since it affects not only the observations but the support and density of the underlying probabilistic process. It can be shown that this definition has a connection to the difference between the perplexity estimates for the language models constructed based on the word representation (i.e., $h(O)$) and the token representation (i.e., O), respectively.

Combining Equations (A.10) and (A.11), we write out the scalarized utility-bias solution based on Equation (A.7) as follows:

$$\begin{aligned} \text{maximize} \quad & \lambda \sum_{w \in \mathrm{WT}(h(O))} g(w) + (1-\lambda)|\tilde{H}_{h(O)}(W) - \tilde{H}_O(C)| \\ \text{subject to} \quad & h \subseteq \mathcal{H} \end{aligned} \tag{A.12}$$

Iterative Approximation Globally optimizing Equation (A.12) is infeasible since we need to consider all the possible translation rulesets. To alleviate this problem, we develop an iterative algorithm that allows us to explore the search space more efficiently.

The search strategy we take is called *greedy inclusion*. Since each hypothesis in $\mathcal{H}$ is defined with respect to some ordered ruleset R, we maintain the best ruleset B that we have obtained so far and greedily expand B by adding new rules at the end of the set. Formally, in each iteration, we seek to maximize the objective with respect to all the possible rules r by forming the corresponding solution $B \cup \{r\}$; once the best rule is found, we add it to the end of B. This procedure is repeated many times until the terminal condition is satisfied.

This algorithm can be made more efficient if we make two changes: (1) at the end of each iteration, we rewrite the sequence O by applying the rule r, right before updating the ruleset B, and (2) in each iteration, we consider optimizing over only all the possible new rule r. This is legit in a greedy inclusion algorithm, since, in each iteration, we effectively form a partial solution based on all the existing rules in B and then optimize over all the possible new rules r with respect to this partial solution.

A brief sketch of the algorithm is given as follows:

1. Let B be an empty ordered ruleset, and let $O^{(0)}$ be the original sequence of tokens.

2. Repeat the following steps for each $i \in \mathcal{N}$, starting from $i = 1$, until the terminal condition is satisfied.

(a) Find a rule r in the form of $w \to c_1 c_2 \dots c_n$ for some n, where $w \in \mathcal{W}$ and each $c_i \in \mathcal{C}$, such that Equation (A.12) is maximized with respect to $h = h_{\{r\}}$ and $O = O^{(i-1)}$.

(b) Form a new token sequence $O^{(i)}$ by applying the rule r to $O^{(i-1)}$.

(c) Add r to the end of B.

3. Output B.

In later subsections, we show that there exists an efficient implementation for this algorithm in a simplified case when we consider only bigram translation rules.

A.3.5 Example: Static Index Pruning

Let $\mathcal{T}$ denote the set of terms and $\mathcal{D}$ denote the set of documents in a retrieval system. Consider that we have established a retrieval model based on a series of observations $O = \{O_1, O_2, \dots, O_N\}$, in which each $O_k = (T_k, D_k) \in \mathcal{T} \times \mathcal{D}$ represents a posting for term T_k and document D_k, i.e., term T_k appears in document D_k. In static index pruning, we seek to reduce the model size for space efficiency by keeping only a subset of the observations in the index, denoted as $L \subseteq O$, and discarding all the other less important entries. It is clear that, in this respect, static index pruning is related to a model degeneration problem as discussed in Section A.3.3.

We complete the utility-bias definition in Equation (A.7) by specifying the hypothesis space, the utility, and the bias as follows.

Hypothesis We define a hypothesis space $\mathcal{H}$ that covers all the possible ways for selecting a subset of observations to keep, i.e., $\mathcal{H} = \{h_S \mid S \subseteq \mathcal{T} \times \mathcal{D}\}$, in which each hypothesis function $h_S : \mathcal{P}(\mathcal{T} \times \mathcal{D}) \to \mathcal{P}(\mathcal{T} \times \mathcal{D})$ is defined as:

$$h_S(O) = O \cap S. \tag{A.13}$$

Utility The utility with respect to some hypothesis h and observations O is given as the number of postings discarded by the selection, as in:

$$\text{utility}(O, h) = |O - h(O)|. \tag{A.14}$$

Bias The ideal way to define the bias with respect to some hypothesis h and observations O is to quantify how the change introduced to the model affects the predictability. We propose developing this quantity based on the conditional entropy $H(D|T)$. Formally speaking, we define

$$\text{bias}(O, h) = |\tilde{H}_O(D|T) - \tilde{H}_{h(O)}(D|T)|, \tag{A.15}$$

where $\tilde{H}_O(D|T)$ and $\tilde{H}_{h(O)}(D|T)$ denote the conditional entropy $H(D|T)$ estimated based on the original observations O and on the reduced set $h(O)$, respectively.

This definition can be further simplified by considering the *pruned observations* $r(O, h) = O - h(O)$ with respect to some h and O. First of all, the utility is the size of the pruned set as:

$$\text{utility}(O, h) = r(O, h). \tag{A.16}$$

Next, we derive a simplified equation for the bias. Assume that $p(d)$ is uniform for all d, i.e., $p(d) = 1/|\mathcal{D}|$ for all $d \in \mathcal{D}$. The bias with respect to some h and O is written as:

$$\begin{aligned}
\text{bias}(O, h) &= |\tilde{H}_O(D|T) - \tilde{H}_{h(O)}(D|T)| \\
&\approx -\sum_{(t,d)\in r(O,h)} p(t, d) \log p(d|t) \\
&= -\sum_{(t,d)\in r(O,h)} p(d)p(t|d) \log \frac{p(d)p(t|d)}{\sum_{d'\in\mathcal{D}} p(d')p(t|d')} \\
&= -\frac{1}{|\mathcal{D}|} \sum_{(t,d)\in r(O,h)} p(t|d) \log \frac{p(t|d)}{\sum_{d'\in\mathcal{D}} p(t|d')}
\end{aligned} \tag{A.17}$$

In the second line, we assume that $\tilde{H}_{h(O)}(D|T)$ can be estimated based on the current probabilistic model p and the set $h(O)$ that we choose to keep. We also remove the absolute value since each component in the summation is positive, i.e., $-\log p(d|t) \geq 0$. The third line follows by decomposing the joint probability $p(t, d)$ and applying Bayes' Theorem to $p(d|t)$. The last line follows the uniformity assumption for $p(d)$.

The probability $p(t|d)$ has an interpretation in language modeling. It denotes the likelihood of drawing a term t from the language model M_d created for document d. For brevity, we denote the probability $p(t|d)$ as a score $S_{t,d}$ for term t and document d, since $p(t|d)$ is usually estimated via various smoothing methods that heavily depend on the implementation. We also write the summation $\sum_{d' \in \mathcal{D}} p(t|d')$ as a sum of scores S_t for term t over all the documents, i.e., $S_t = \sum_{d \in \mathcal{D}} S_{t,d}$.

In a retrieval system based on language modeling, these two scores $S_{t,d}$ and S_t are generally accessible. Now, rewrite the previous equation for the bias accordingly:

$$\text{bias}(O, h) = -\frac{1}{|\mathcal{D}|} \sum_{(t,d) \in r(O,h)} S_{t,d}(\log S_{t,d} - \log S_t) \tag{A.18}$$

Combining Equations (A.16) and (A.18), we write out a corresponding utility-bias solution as in Equation (A.7):

$$\begin{aligned} \text{maximize} \quad & \lambda \cdot |R| + (1-\lambda)\frac{1}{|\mathcal{D}|} \sum_{(t,d) \in R} S_{t,d}(\log S_{t,d} - \log S_t) \\ \text{subject to} \quad & R \subseteq O \end{aligned} \tag{A.19}$$

Iterative Approximation Optimization over all the possible $R \subseteq O$ as described in Equation A.19 is generally intractable since the number of such sets are exponentially many. To deal with this issue, we propose using a greedy inclusion algorithm to approximate the search.

The idea is maintain the best solution B that we have obtain so far, and iteratively expand this set. In each iteration, we seek to maximize the objective value with respect to all the unselected postings $O_k \in O - B$ by forming a new solution $B \cup \{O_k\}$; the best such posting is later added into B. This procedure is repeated until the pruning ratio reaches a predefined threshold.

This algorithm is summarized in the following paragraph:

1. Let $B = \emptyset$.
2. Repeat the following steps until the pruning ratio $|B|/|O|$ reaches a predefined threshold.
    (a) Find a posting $(t, d) \in O - B$ such that Equation (A.19) is maximized with respect to $R = B \cup \{(t, d)\}$.

    (b) Add (t, d) into B.
3. Output B.