Developing Probabilistic Graphical Models for Identifying Text Patterns and Semantics

by

Minhua Huang

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy. The City University of New York

2013

This manuscript has been read and accepted for the
Graduate Faculty in Computer Science in satisfaction
of the dissertation requirement for the degree of Doctor of Philosophy.

Robert M. Haralick

Data — Chair of Examining Committee

Ted Brown

Data — Executive Officer

William Sakas

Martin Chodorow

Calandra Moore

Aleksey Myasnikov

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

Developing Probabilistic Graphical Models for Identifying Text Patterns and Semantics

by

Minhua Huang

Advisor: **Professor Robert M. Haralick**

In this dissertation, we discuss a new probabilistic graphical model called the data sequence dependence model (*DSDM*). This model is derived from the joint probability function for a sequence of class assignments given a sequence of input symbols (words) under fewer and different conditional independence assumptions than the commonly used probabilistic graphical models, such as hidden Markov models (*HMM*s), maximum entropy Markov models (*MEMM*s), and conditional random fields (*CRF*s). Our model accounts for the data sequence dependency rather than the class sequence dependency.

In order to find a sequence of optimal class assignments for a sequence of input symbols, the *HMM* and *CRF* models have to employ dynamic programming. In contrast to these models, our method does not need to employ dynamic programming. Although dynamic programming is an efficient optimization technique, our model leads to an algorithm whose computational complexity is less than dynamic programming and whose performance is just as good or better.

Based on *DSDM*, we develop algorithms for identifying semantics in texts. In this research, semantics consists of three types of text patterns. They are the semantic arguments of a verb, the

sense of a polysemous word, and the noun phrases of a sentence.

In addition, two other probabilistic graphical models are described. They are called the context independence model (*CIM*) and the class sequence dependence model (*CSDM*). These models have the same economic gain function as *DSDM*. However, they are derived under the different conditional independence assumptions.

Finally, statistical testing methodologies are employed to validate these models. For our task of identifying semantic patterns, we compare each pair of the models by testing the null hypothesis that the two models are equally good at identifying semantic patterns against the alternative hypothesis that one model is better able to identify semantic patterns. The resulting $p - value$ shows that *DSDM* is better able to identify semantic patterns than the other models.

ACKNOWLEDGEMENTS

I would like to thank my advisor Professor Robert M. Haralick for his excellent guidance and persistent inspiration throughout the years. His philosophy of thinking in complete thoughts: thoughts that are correct, complete, and unambiguous guides my research. He has led me to have a sold and broad foundation of knowledge on machine learning and pattern recognition and prepared me for my professional career.

I would like to thank Professor Martin Chodorow, Professor William Sakas, Professor Aleksey Myasnikov, and Professor Calandra Moore for serving on my Ph.D committee and for their scientific advises throughout the years and their valuable comments on my dissertation.

I would like to thank my family. I would like to thank my parents for bringing me into the world. I would like to thank my husband for his unconditional love and constant support. I would like to thank my son who has always brought me joy.

DEDICATION

To my husband *Michael Shi* and my son *Ean Shi*

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Text patterns, such as the semantic arguments of a verb, the sense of a polysemous word, and the noun phrases of a sentence, are essential patterns for capturing semantics in texts. For example, the semantic arguments of a verb can be used to answer the questions of who, what, when, where, and why; the sense of a polysemous word can be used to understand the meaning of the word; the noun phrases of a sentence can be used to obtain the basic components of the meaning of the sentence. In this dissertation, we discuss a new probabilistic graphical model called the data sequence dependence model (*DSDM*) that leads to algorithms for identifying these kinds of semantic text patterns.

Probabilistic graphical models, such as hidden Markov models (*HMM*s) [1], maximum entropy Markov models (*MEMM*s) [2], and conditional random fields (*CRF*s) [3], have been used for identifying text patterns in a sentence by researchers over the years. These models are derived from either a joint probability function or a conditional probability function for a sequence of class assignments given a sequence of input symbols (words) under some conditional independence assumptions. These assumptions might not be the best assumptions for capturing such text patterns in a sentence. For example, one of their conditional independence assumptions is that the class identification for the current symbol depends only on the class identification of the previous symbol, not others.

Moreover, these graphical models lead to an optimization by threading through a sequence of class assignments to optimize the joint or conditional probability of the sequence of class

assignments given a sequence of symbols. This optimization can be understood as an optimization of expected economic gain [4]. The economic gain function employed by these common models has a gain of one if all class assignments are correct and zero if any assignment is wrong. No partial credit is given for some correct assignments. This criterion leads to difficulties where noise is in the text data. A noisy or perturbed symbol at any position in an input sequence can produce a wrong class category path for a substantial length subsequence.

Furthermore, for these graphical models, the maximum global probability value cannot be determined until the last symbol of the sequence has been reached and the computation must be done by a dynamic programming algorithm. In order to identify a sequence of n symbols among a set of m classes, these models have a time complexity of $O(m^2n)$.

In contrast to these probabilistic graphical models, our model has fewer and different conditional independence assumptions. Our model accounts for the data sequence dependency rather than the class sequence dependency. The class identification of the current symbol is based on information carried by the current symbol, the information between the current symbol and preceding symbol, and information between the current symbol and succeeding symbol. Our economic gain function gives a partial credit for each correct class assignment. When we make a mistake on one symbol in a sequence, it will not effect other correct decisions that have been made or will be made for other symbols.

Furthermore, our method does not need to employ dynamic programming. Although dynamic programming is an efficient optimization technique, our model leads to an algorithm whose computational complexity is less than dynamic programming and whose performance is just as good or better. In order to identify a sequence of n symbols among a set of m classes, our model has a time complexity of $O(mn)$.

Based on the model, we develop algorithms for identifying the noun phrases of a sentence, the sense of an ambiguous word, and the semantic arguments of a verb in a sentence. In our algorithms, an input symbol sequence can be either a sentence or a sequence of context words of an ambiguous word or a path in the parse tree associated with a sentence. Class categories are defined in Sections 7.3.1, 7.4.1, and 7.5.4. To find noun phrases, blocks are found from the assigned class category sequence. Each block is associated with a noun phrase. To find the sense

of a word, the class category with the maximum number of votes is selected from the assigned class category sequence. To find the semantic arguments of a verb, we build a set of subtrees. A semantic argument of the verb is associated with the leaves of a tree.

We also discuss two other probabilistic graphical models. They are called the context independence model (*CIM*) and the class sequence dependence model (*CSDM*). These models have the same economic gain function as *DSDM*. However, they are derived under the different conditional independence assumptions. For *CIM*, the class identification of the current symbol is based on information carried by the current symbol. For *CSDM*, the class identification of the current symbol is based on information carried by the current symbol and the previous class identification.

Experiments conducted on standard data sets show good results. For example, methods derived from *DSDM* achieve an average precision of 92.96% and an average recall of 94.94% for recognizing the semantic arguments of verbs on *WSJ* data from Penn Treebank and PropBank, an average accuracy of 81.12% for recognizing the six sense word *line*, and an average precision of 97.7% and an average recall of 98.8% for recognizing noun phrases on *WSJ* data from Penn Treebank.

Moreover, we use statistical hypothesis testing to compare any pair of models among *DSDM*, *CIM*, and *CSDM*. At the significance level 0.05, we say that the performance achieved by *DSDM* is better than the ones achieved by *CIM* and *CSDM* for identifying noun phrases on *WSJ* data from Penn Treebank. Furthermore, in order to do more comparisons, we implement a hidden Markov model (*HMM*) and test it for identifying noun phrases on *CoNLL*-2000 data. By the $f - measure$ in Table 7.4, we say that *DSDM* is about 2.1% better than the *HMM*. We also test a maximum entropy model (*MEM*) for identifying polysemous words *line*, *interest*, *hard*, and *serve* from the corpus *SENSEVAL 2* by running the Maxent software package implemented in Python [5]. By Table 7.20 and Table 7.22, we say that the accuracies achieved by *DSDM* are far better than the accuracies achieved by the *MEM*.

1.1 Contributions

Our main contributions include:

1. A new probabilistic graphical model called the data sequence dependence model (*DSDM*) for a sequence of class assignments given a sequence of input symbols [6] [7] [8].
   It has fewer and different conditional independence assumptions compared with other commonly used probabilistic graphical models. It accounts for the data sequence dependency rather than the class sequence dependency. It gives a partial credit for each correct class assignment. It does not need to employ dynamic programming. It requires less operating time for identifying a new sequence of input symbols than competing techniques.

2. An algorithm for identifying the semantic the arguments of a verb [9].
   For each verb in a parse tree $T = (V, E)$ associated with a sentence, the algorithm discovers a path. From the path, a set of roots are extracted. Each root induces a subtree of the parse tree. The leaves of a subtree constitute a semantic argument of the verb. The time complexity of the algorithm is $O(|E|)$ and the memory complexity of the algorithm is $O(|V|^2)$.

3. An algorithm for identifying the sense of a word [10].
   For a sequence of n context symbols associated with a polysemous word, the algorithm assigns a sequence of class categories. Then, it determines the sense of the word by selecting the most frequently assigned class category. The time complexity of the algorithm is $O(nm)$ and the memory complexity of the algorithm is $O(n^2m)$, where m is the number of classes.

4. An algorithm for identifying the noun phrases in a sentence [11].
   For a sequence of n symbols associated with a sentence, the algorithm assigns a sequence of class categories. From the category sequence, the algorithm finds blocks such that each of these blocks is one of the noun phrases. The time complexity of the algorithm is $O(nm)$ and the memory complexity of the algorithm is $O(n^2m)$, where m is the number of classes.

5. Two other probabilistic graphical models [8].
   They are the context independence model (*CIM*) and the class sequence dependence model (*CSDM*). They also give a partial credit for each correct class assignment for a sequence of input symbols. However, the conditional independence assumptions are different from *DSDM*.

6. A method of employing statistical hypothesis testing.

   Papers [12] [14] have discussed statistical permutation tests for NLP tasks. In our case, we compare each pair of models by testing the null hypothesis that two models are equally good at identifying semantic patterns against the alternative hypothesis that one model is better able to identify semantic patterns. The resulting $p - value$ shows which model is better than the other.

1.2 Overview

In Chapter 2, we discuss conditional independence graphs and the fundamental properties of conditional independence graphs. Moreover, we discuss how to obtain the joint probability distributions over all random variables using conditional independence graphs. We show some examples.

In Chapter 3, we introduce a new probabilistic graphical model called the data sequence dependence model (*DSDM*). We also discuss two other probabilistic graphical models. They are the context independent model (*CIM*) and the class sequence dependence model (*CSDM*). We show the conditional independences graphs and mathematical expressions of these models. We discuss time and memory complexities of these models. We describe the properties of *DSDM*.

In Chapter 4 we describe commonly used methods for finding an optimal sequence for probabilistic graphical models. We discuss two types of the hidden Markov models built based on the expected economic gain of the second type. We discuss two dynamic programming algorithms. We give examples to show the procedures of these algorithms.

In Chapter 5 and Chapter 6, we discuss some NLP tasks related semantics. In chapter 5, we discuss some existed classification methods for Word Sense Disambiguation, *NP* Chunking, and Semantic Role Labeling. In Chapter 6, we discuss several techniques for reducing noise in textual data. We discuss two smoothing methods for handling unseen events.

In Chapter 7, based on our model, we develop an algorithm for identifying the semantic arguments of a verb, an algorithm for identifying the sense of a polysemous word, and an algorithm for identifying the noun phrases of a sentence. We discuss the worst case performances and the

worst case space complexities of these algorithms. Moreover, we give experimental results for the validations of these methods.

In Chapter 8, we give our conclusions.

Chapter 2

Probabilistic Graphical Models

In this section, we discuss probabilistic graphical models. We discuss conditional independence graphs. We discuss the fundamental properties of conditional independence graphs. Moreover, we discuss how to obtain joint probability distributions over all random variables using conditional independence graphs.

2.1 Fundamentals

Conditional Independence

Let A, B, and C be random variables. Let P be a probability distribution on these random variables. A is conditional independent of B given C, written as $A \perp\!\!\!\perp B \mid C$ if and only if $P(AB|C) = P(A|C)P(B|C)$.

$$A \perp\!\!\!\perp B \mid C \leftrightarrow P(AB|C) = P(A|C)P(B|C)$$

Probabilistic Graphical Model

A multi-dimensional probability distribution P on N random variables determines an undirected graph $G = (V, E)$. Each of the nodes represents one random variable. Node i and node j are not connected by an edge if and only if variable i is conditionally independent of variable j given the remaining variables, written as

$$i \perp\!\!\!\perp j \mid V - \{i, j\}$$

Conditional Independence Graph

In this case, G is called the conditional independence graph.

When the conditional independence graph is triangulated (see Section A.2.10), then the joint probability function can be expressed as a product of joint probability functions over cliques of theses variables divided by a product of joint probability functions for variables in the separators (see Section A.2.11).

Directed acyclic graphs *DAG*s can also be used to express conditional independences of P. This happens when a node A is conditionally independent of its non-descendants (see Section A.2.8) given its parents. *DAG*s are better able to express causal relationships between random variables. A conditional probability is associated with each node i and its parents $\pi(i)$: $P(i|\pi(i))$. The joint probability function over all the random variables is a product of conditional probability functions over the subsets of these variables.

Moreover, a directed graph can be converted into an undirected graph and visa versa [15].

2.1.1 Semi-Graphoid

Definition

Let S be a set. Let $G = \{ (A, B, C) \mid A \cap B = \phi, A \cap C = \phi, B \cap C = \phi, A, B, C \subseteq S \}$. G is called a Semi-Graphoid if it satisfies the properties of 1, 2, 3, and 4.

1. Symmetry: $(A, B, C) \in G \leftrightarrow (B, A, C) \in G$
2. Decomposition: $(A \cup B, C, D) \in G \rightarrow (B, C, D) \in G$
3. Weak Union: $(A \cup B, C, D) \in G \rightarrow (B, C, D \cup A) \in G$
4. Contraction: $(A, B, D) \in G$ *and* $(A, C, D \cup B) \in G \rightarrow (A, B \cup C, D) \in G$

Theorem

Let I be an index set containing the indexes of all the random variables. Let P be probability distribution. Let $H = \{ (A, B, C) \mid A \perp\!\!\!\perp B \mid C \}$ with respect to P, where $A \cap B = \phi, A \cap C = \phi, B \cap C =$

$\phi, A, B, C \subseteq I$. Then, H is a Semi-Graphoid.

To show that H is a Semi-Graphoid, we need to show:

- $A \perp\!\!\!\perp B \mid C \leftrightarrow B \perp\!\!\!\perp A \mid C$
- $A \cup B \perp\!\!\!\perp C \mid D \rightarrow B \perp\!\!\!\perp C \mid D$
- $A \cup B \perp\!\!\!\perp C \mid D \rightarrow B \perp\!\!\!\perp C \mid D \cup A$
- $(A \perp\!\!\!\perp B \mid D)$ *and* $(A \perp\!\!\!\perp C \mid D \cup B) \rightarrow A \perp\!\!\!\perp B \cup C \mid D$

These four properties are proved by the proposition 1 - 4.

Proposition 1. $A \perp\!\!\!\perp B \mid C \leftrightarrow B \perp\!\!\!\perp A \mid C$

Proof.

On the left side, $A \perp\!\!\!\perp B \mid C \rightarrow P(AB|C) = P(A|C)P(B|C)$

$$A \perp\!\!\!\perp B \mid C \quad \rightarrow \quad P(AB \mid C) = P(A|C)P((B|C) \rightarrow$$
$$P(AB|C) = P(B|C)P(A|C) \rightarrow B \perp\!\!\!\perp A \mid C$$

On the right side, $B \perp\!\!\!\perp A \mid C \rightarrow P(BA|C) = P(B|C)P(A|C)$

$$B \perp\!\!\!\perp A \mid C \quad \rightarrow \quad P(BA \mid C) = P(B \mid C)P(A \mid C) \rightarrow$$
$$P(BA \mid C) = P(A \mid C)P(B \mid C) \rightarrow A \perp\!\!\!\perp B \mid C$$

$$\therefore A \perp\!\!\!\perp B \mid C \leftrightarrow B \perp\!\!\!\perp A \mid C$$

Hence, the Proposition 1 holds.

Proposition 2. $A \cup B \perp\!\!\!\perp C \mid D \rightarrow B \perp\!\!\!\perp C \mid D$

Proof.

On the left side, $A \cup B \perp\!\!\!\perp C \mid D \rightarrow P(ABC|D) = P(AB|D)P(C|D)$

$$A \cup B \perp\!\!\!\perp C \mid D \rightarrow \qquad P(ABC|D) = P(AB|D)P(C|D) \rightarrow$$
$$\sum_A P(ABC|D) = \sum_A P(AB|D)P(C|D) \rightarrow$$

$$P(BC|D) = P(B|D)P(C|D)$$

$$\therefore B \perp\!\!\!\perp C \mid D$$

Therefore, the Proposition 2 holds.

Proposition 3. $A \cup B \perp\!\!\!\perp C \mid D \rightarrow B \perp\!\!\!\perp C \mid D \cup A$

Proof.

To show $B \perp\!\!\!\perp C \mid D \cup A$, we need to show $P(BC|DA) = P(B|DA)P(C|DA)$ by applying $A \cup B \perp\!\!\!\perp C \mid D$.

$$\begin{aligned}
& P(BC|AD) = \frac{P(ABCD)}{P(AD)} = \frac{P(ABC|D)P(D)}{P(AD)} \\
= \quad & \frac{P(AB|D)P(C|D)P(D)}{p(AD)} \qquad \textit{since } A \cup B \perp\!\!\!\perp C \mid D \rightarrow P(ABC|D) = P(AB|D)P(C|D) \\
= \quad & \frac{P(ABD)P(C|D)}{P(AD)} = P(B|AD)P(C|D) \\
= \quad & P(B|AD)\frac{P(ABCD)}{P(AB|CD)P(D)} \qquad \textit{since } P(ABCD) = P(AB|CD)P(C|D)P(D) \rightarrow P(C|D) = \frac{P(ABCD)}{P(AB|CD)P(D)} \\
= \quad & P(B|AD)\frac{P(B|ACD)P(ACD)}{P(AB|D)P(D)} \qquad \textit{since } A \cup B \perp\!\!\!\perp C \mid D \rightarrow P(AB|CD) = P(AB|D) \\
= \quad & P(B|AD)\frac{P(B|AD)P(ACD)}{P(AB|D)P(D)} \qquad \textit{since } A \cup B \perp\!\!\!\perp C \mid D \rightarrow P(B|ACD) = P(B|AD) \\
= \quad & P(B|AD)\frac{P(ABD)P(ACD)P(D)}{P(ABD)P(AD)P(D)} = P(B|AD)P(C|AD) = P(B|DA)P(C|DA) \\
& \therefore B \perp\!\!\!\perp C \mid D \cup A
\end{aligned}$$

Therefore, the Proposition 3 holds.

Proposition 4. $(A \perp\!\!\!\perp B \mid D)$ *and* $(A \perp\!\!\!\perp C \mid D \cup B) \rightarrow A \perp\!\!\!\perp B \cup C \mid D$

Proof.

To show $A \perp\!\!\!\perp B \cup C \mid D$, we need to show $P(ABC|D) = P(A|D)P(BC|D)$ by applying $(A \perp\!\!\!\perp B \mid D)$ and $(A \perp\!\!\!\perp C \mid D \cup B)$.

$$\begin{aligned}
& P(ABC|D) = \frac{P(ABCD)}{P(D)} = \frac{P(AC|BD)P(BD)}{P(D)} \\
= \quad & \frac{P(A|BD)P(C|BD)P(BD)}{P(D)} \qquad \textit{since } A \perp\!\!\!\perp C \mid B \cup D \rightarrow p(AC|BD) = p(A|BD)p(C|BD) \\
= \quad & \frac{P(A|BD)P(CBD)}{P(D)} = \frac{P(A|BD)P(CB|D)P(D)}{P(D)} = P(CB|D)P(A|BD)
\end{aligned}$$

$$
\begin{aligned}
&= P(CB|D)\frac{P(ABD)}{P(BD)} = P(CB|D)\frac{P(AB|D)P(D)}{P(BD)} \\
&= P(CB|D)\frac{P(A|D)P(B|D)P(D)}{P(BD)} \qquad \textit{since } A \perp\!\!\!\perp B \mid D \rightarrow P(AB|D) = P(A|D)P(B|D) \\
&= P(CB|D)\frac{P(A|D)P(BD)}{P(BD)} = P(CB|D)P(A|D) = P(A|D)P(BC|D) \\
&\therefore A \perp\!\!\!\perp B \cup C \mid D
\end{aligned}
$$

Therefore, the Proposition 4 holds.

2.1.2 Graphoid

Definition

Let S be a set. Let $G = \{ (A, B, C) \mid A \cap B = \phi, A \cap C = \phi, B \cap C = \phi, A, B, C \subseteq S \}$. G is called a Graphoid if it satisfies the properties of 1, 2, 3, and 4 of a Semi-Graphoid and the intersection property.

1. Intersection: $(A, B, D \cup C) \in G$ *and* $(A, C, D \cup B) \in G \rightarrow (A, B \cup C, D) \in G$

Theorem

Let I be an index set containing the indexes of all the random variables. Let P be probability distribution. Let $H = \{ (A, B, C) \mid A \perp\!\!\!\perp B \mid C \}$ with respect to P, where $A \cap B = \phi, A \cap C = \phi, B \cap C = \phi, A, B, C, D \subseteq I, P(ABCD) > 0$. Then, H is a Graphoid.

To show H is a Graphoid, we need to show:

- $A \perp\!\!\!\perp B \mid C \leftrightarrow B \perp\!\!\!\perp A \mid C$
- $A \cup B \perp\!\!\!\perp C \mid D \rightarrow B \perp\!\!\!\perp C \mid D$
- $A \cup B \perp\!\!\!\perp C \mid D \rightarrow B \perp\!\!\!\perp C \mid D \cup A$
- $(A \perp\!\!\!\perp B \mid D)$ *and* $(A \perp\!\!\!\perp C \mid D \cup B) \rightarrow A \perp\!\!\!\perp B \cup C \mid D$
- $(A \perp\!\!\!\perp B \mid D \cup C)$ *and* $(A \perp\!\!\!\perp C \mid B \cup D) \rightarrow A \perp\!\!\!\perp B \cup C \mid D$

We have shown the first four properties by the propositions of 1, 2, 3, and 4. The last property is proved by the proposition 5.

Proposition 5. $(A \perp\!\!\!\perp B \mid D \cup C)$ *and* $(A \perp\!\!\!\perp C \mid B \cup D) \Rightarrow A \perp\!\!\!\perp B \cup C \mid D$

Proof.

First, we show that $P(A|BD) = P(A|D)$ when $P(ABCD) > 0$.

$$
\begin{aligned}
P(A|BD) &= \frac{P(ABCD)}{P(C|ABD)P(BD)} \qquad \textit{since } P(ABCD) = P(C|ABD)P(A|BD)P(BD) \\
&= \frac{P(ABCD)}{P(C|BD)P(BD)} \qquad \textit{since } A \perp\!\!\!\perp C \mid B \cup D \rightarrow P(C|ABD) = P(C|BD) \\
&= \frac{P(ABCD)}{P(BCD)} \\
&= P(A|BCD) = P(A|CD) \qquad \textit{since } A \perp\!\!\!\perp C \mid B \cup D \rightarrow P(A|BCD) = P(A|BD) \\
&= \frac{P(ACD)}{P(CD)} \\
&P(ABCD) > 0 \rightarrow P(CD) > 0, \textit{ we divide } P(CD) \textit{ on both side of the equation, then} \\
&P(A|BD)P(CD) = P(ACD) \\
&\sum_C P(A|BD)P(CD) = \sum_C P(ACD) \\
&P(A|BD)P(D) = P(AD) \\
&P(A|BD) = P(A|D)
\end{aligned}
$$

Now, we show $P(ABC|D) = P(A|D)P(BC|D)$

$$
\begin{aligned}
& P(A, BC|D) = \frac{P(ABCD)}{P(D)} = \frac{P(AC|BD)P(BD)}{P(D)} \\
= \;& \frac{P(A|BD)P(C|BD)P(BD)}{P(D)} \qquad \textit{since } A \perp\!\!\!\perp C \mid B \cup D \rightarrow P(AC|BD) = P(A|BD)P(C|BD) \\
= \;& P(A|BD)P(BC|D) \\
= \;& P(A|D)P(BC|D) \qquad \textit{since } P(A|BD) = P(A|D) \\
\therefore \;& A \perp\!\!\!\perp B \cup C \mid D
\end{aligned}
$$

Therefore, the Proposition 5 holds.

2.2 Undirected Graphs

Let $G = (X, E)$ be an undirected graph. When its vertex set X is associated with random variables and its edge set E is associated with relationships between random variables, then G describes a probability distribution P on these random variables.

2.2.1 Factorization

Let $C = \{A \subseteq X \mid A \text{ is a clique of } G\}$. A probability distribution P over X factors over G if its joint probability density function has the form:

$$f(X) = \frac{\prod_{A \in C} \phi_A(A)}{Z} = \prod_{A \in C} \psi_A(A) \tag{2.1}$$

Here, Z is a normalization constant, s.t. $Z = \sum_X \prod_{A \subseteq X} \phi_A(A)$.

For example, in Figure 2.1,

$$f(X) = \psi_{12}(1,2)\psi_{14}(1,4)\psi_{256}(2,5,6)\psi_{236}(2,3,6)\psi_{367}(3,6,7)\psi_{389}(3,8,9)\psi_{349}(3,4,9)\psi_{4910}(4,9,10)$$

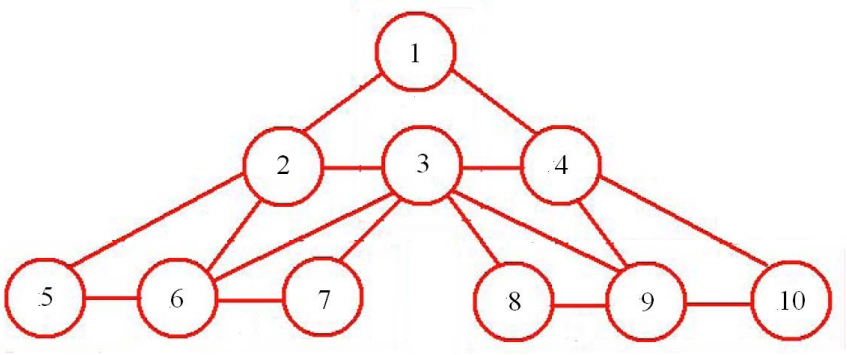


Figure 2.1: Computing $f(X)$ from the undirected graph G

$f(X) = \psi_{12}(1,2)\psi_{14}(1,4)\psi_{256}(2,5,6)\psi_{236}(2,3,6)\psi_{367}(3,6,7)\psi_{389}(3,8,9)\psi_{349}(3,4,9)\psi_{4910}(4,9,10)$

2.2.2 Global Markov Property

Let P be a probability distribution and let G be the conditional independence graph of P. Let A, B, and C be subsets of X. In G, if A is separated from B by C, we write as

$$A \perp_G B \mid C.$$

In P, if A is independent of B given C, $P(A, B|C) = P(A|C)P(B|C)$, we write as

$$A \perp\!\!\!\perp B \mid C.$$

We say that P has the global Markov property if

$$A \perp_G B \mid C \Longrightarrow A \perp\!\!\!\perp B \mid C \tag{2.2}$$

For example, in Figure 2.1, if P satisfies the global Markov property, then

$$\{2, 5, 6, 7\} \perp\!\!\!\perp \{4, 8, 9, 10\} \mid \{1, 3\}$$

$$\{3, 8\} \perp\!\!\!\perp \{10\} \mid \{4, 9\}$$

$$\{1\} \perp\!\!\!\perp \{5, 6, 7, 8, 9, 10\} \mid \{2, 3, 4\}$$

2.2.3 Local Markov Property

Let G be a conditional independence graph of P. P satisfies the local Markov property if

$$\forall x \in X : x \perp\!\!\!\perp rm(x) \mid nb(x) \tag{2.3}$$

where $nb(x)$ denotes the neighbors of x and $rm(x) = X - nb(x)$ in graph G.

The local Markov property states that every variable is conditionally independent of the remaining variables given its neighbors. For example, in Figure 2.1, some conditional independences can be

obtained from the local Markov property:

$$1 \perp\!\!\!\perp \{3,5,6,7,8,9,10\} \mid \{2,4\}$$

$$3 \perp\!\!\!\perp \{1,5,10\} \mid \{2,6,7,4,8,9\}$$

$$9 \perp\!\!\!\perp \{1,2,5,6,7\} \mid \{3,4,8,10\}$$

2.2.4 Pairwise Markov Property

Let G be a conditional independence graph of P. P satisfies the pairwise Markov property if

$$\forall x, x' \in X : x \perp\!\!\!\perp x' \mid rm(x, x') \tag{2.4}$$

where $rm(x, x') = V - \{x, x'\}$ in G.

The pairwise Markov property states that every non-adjacent pair of random variables is conditionally independent given the remaining variables. By the definition of the conditional independence graph, P always satisfies the pairwise Markov property. For example, in Figure 2.1, some conditional independences can be obtained from the pairwise Markov property:

$$1 \perp\!\!\!\perp 5 \mid \{2,3,4,6,7,8,9,10\}$$

$$3 \perp\!\!\!\perp 10 \mid \{1,2,4,5,6,7,8,9\}$$

$$9 \perp\!\!\!\perp 1 \mid \{2,3,4,5,6,7,8,10\}$$

2.2.5 Theorems

Theorem A

Let (F) denote the factorization property. Let (G) denote the global Markov property. Let (L) denote the local Markov property. Let (P) denote the pair wised Markov property. Then,

$$(F) \Longrightarrow (G) \Longrightarrow (L) \Longrightarrow (P) \tag{2.5}$$

Lauritzen [16] has proved (2.5).

If the joint probability density function $f(X) > 0$, then Lauritzen [16] proves that

$$(P) \Longrightarrow (F)$$

Therefore, if $f(x) > 0$, it follows that

$$(F) \Longleftrightarrow (G) \Longleftrightarrow (L) \Longleftrightarrow (P)$$

Moreover, for any semigraphoid, it holds that

$$(G) \Longrightarrow (L) \Longrightarrow (P) \tag{2.6}$$

In the graphoid case

$$(P) \Longrightarrow (F)$$

Theorem B

In the case that P satisfies the graphoid condition,

$$(G) \Longleftrightarrow (L) \Longleftrightarrow (P) \tag{2.7}$$

2.2.6 Joint Probability for a Strongly Decomposable Graph

Let $G = (X, E)$ be a strongly decomposable graph (see Section A.2.11), where $X = \{x_1, \ldots, x_N\}$. G has a set of K cliques. Let these cliques be in running order (see Section A.2.11) of $< C_1, C_2, ..., C_K >$, with separators $S_2, ..., S_K$ where

$$S_k = C_k \cap (\cup_{i=1}^{k-1} C_i), \qquad k = 2, ..., K$$

Then, the joint probability function over all elements in X is:

$$p(X) = p(x_1, ..., x_N) = \frac{\prod_{k=1}^{K} p(x_i, i \in C_k)}{\prod_{m=2}^{K} p(x_i, i \in S_m)} = p(x_i, i \in C_1) \prod_{k=2}^{K} p(x_i, i \in C_k - S_k | S_k) \quad (2.8)$$

For example, G_1 in Figure 2.2 is a strongly decomposable graph, where $C_1 = \{x_1, x_2, x_3, x_4\}$, $C_2 = \{x_3, x_4, x_5\}$, $C_3 = \{x_4, x_6\}$, $S_2 = \{x_3, x_4\}$, $S_3 = \{x_4\}$. Therefore, the joint probability function

$$p(x_1, x_2, x_3, x_4, x_5, x_6) = p(x_1, x_2, x_3, x_4) p(x_5 | x_3, x_4) p(x_6 | x_4).$$

Moreover, G_2 in Figure 2.2 is also a strongly decomposable graph, where $C_1 = \{x_1, x_2, x_3\}$, $C_2 = \{x_2, x_3, x_5\}$, $C_3 = \{x_2, x_4, x_5\}$, $C_4 = \{x_4, x_5, x_6, x_7\}$, $S_2 = \{x_2, x_3\}$, $S_3 = \{x_2, x_5\}$, $S_4 = \{x_4, x_5\}$. Thus, the joint probability function

$$p(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = p(x_1, x_2, x_3) p(x_5 | x_2, x_3) p(x_4 | x_2, x_5) p(x_6, x_7 | x_4, x_5).$$

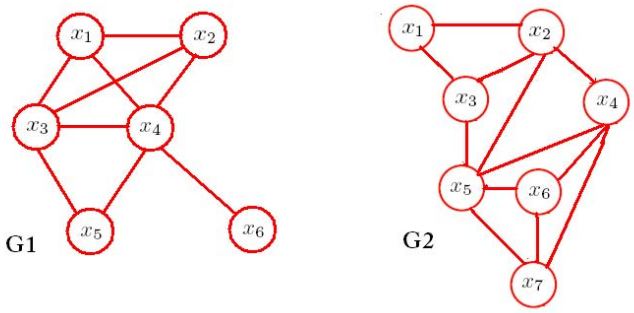


Figure 2.2: Computing joint probabilities for strongly decomposable graphs G_1 and G_2
The joint probability function with respect to G_1 is $p(x_1, x_2, x_3, x_4, x_5, x_6) = p(x_1, x_2, x_3, x_4) p(x_5 | x_3, x_4) p(x_6 | x_4)$ and the joint probability function with respect to G_2 is $p(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = p(x_1, x_2, x_3) p(x_5 | x_2, x_3) p(x_4 | x_2, x_5) p(x_6, x_7 | x_4, x_5)$.

2.3 Directed Acyclic Graph (*DAC*)

2.3.1 Parent

Let $G = (X, E)$ be a directed acyclic graph, $x \in X$. The parents of x is defined as follows:

$$\pi(x) = \{y \in X | (y, x) \in E\}$$

2.3.2 Factorization

A probability distribution P over X factors over G if its joint probability density function has the form:

$$f(X) = \prod_{x \in X} f(x \mid \pi(x)) \tag{2.9}$$

For example, in G_1 of Figure 2.3

$$f(X) = f(x_1)f(x_2)f(x_3)f(x_4|x_1, x_2, x_3).$$

In G_2 of Figure 2.3

$$f(X) = f(x_1)f(x_2)f(x_3|x_1, x_2)f(x_4|x2)f(x_5|x_3)f(x_6)f(x_7|x_4, x_5, x_6).$$

2.3.3 Local Directed Markov Property

A distribution P satisfies the local Markov property with respect to a directed acyclic graph $G = (X, E)$ if

$$x \in X \Rightarrow x \perp\!\!\!\perp nd(x) \mid \pi(x) \tag{2.10}$$

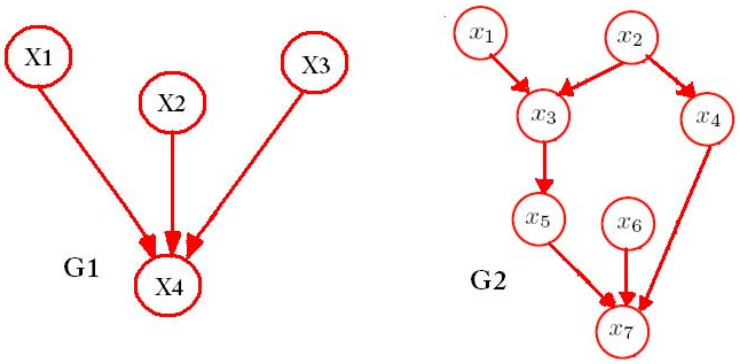


Figure 2.3: Computing $f(X)$ from directed acyclic graphs G_1 and G_2
The factorization over random variables x_i with respect to G_1 is $f(X) = f(x_1)f(x_2)f(x_3)f(x_4|x_1, x_2, x_3)$ and to G_2 is $f(X) = f(x_1)f(x_2)f(x_3|x_1, x_2)f(x_4|x2)f(x_5|x_3)f(x_6)f(x_7|x_4, x_5, x_6)$.

where $nd(x)$ denotes the no descendants of x. The local directed Markov property states that every variable x is conditionally independent of the variables of no descendants of x given all the parents of x. For example, in G_2 of Figure 2.3

$$x_3 \perp\!\!\!\perp \{x_4, x_6\} \mid \{x_1, x_2\}$$

$$x_4 \perp\!\!\!\perp \{x_1, x_3, x_5, x_6\} \mid x_2$$

$$x_7 \perp\!\!\!\perp \{x_1, x_2, x_3\} \mid \{x_4, x_5, x_6\}$$

2.3.4 Rules

Let $G = (X, E)$ be a directed acyclic graph associated with a probabilistic graphical model. Let C be a subset of X. C is associated with a set of observed variables. For any pair of nodes $a, b \in X - C$, a path $a \to \ldots \to b$ is said to be blocked if it satisfies the one of following conditions.

1. *head − to − tail*
    - The arrows on the path meet $a \to \ldots \to c \to \ldots \to b$, where $c \in C$.
    - We call c a *head − to − tail* node.
2. *tail − to − tail*
    - The arrows on the path meet $a \to \ldots \leftarrow c \to \ldots \to b$, where $c \in C$.

- We call c a $tail - to - tail$ node.

3. $head - to - head$

   - The arrows on the path meet $a \rightarrow \ldots \rightarrow x \leftarrow \ldots \rightarrow b$, where $x \notin C$ or $d(x) \notin C$, $d(x)$ is a descendant of x.
   - We call x a $head - to - head$ node.

2.3.5 D-Separation Theory

Let $G = (X, E)$ be a directed acyclic graph associated with a probabilistic graphical model. Let A, B, and C be sets of vertices, s.t. A, B, and C are mutually exclusive. We say that A is d-separated from B by C if every directed path from a node in A to a node in B is blocked by C. We write it as

$$A \perp_G B \mid C \tag{2.11}$$

For example, G_1 in Figure 2.4, $A = \{x_1, x_3, x_5\}$, $B = \{x_4\}$, and $C = \{x_2, x_6, x_7\}$. A, B, and C are mutually exclusive. A is not d-separated from B given C written as $A \not\perp_{G_1} B \mid C$ ($x_1, x_3, x_5 \not\perp_{G_1} x_4 \mid x_2, x_6, x_7$) because a path from node $x_5 \in A$ to node in $x_4 \in B$ is not blocked by node x_7. Although x_7 is a $head - to - head$ node, x_7 is in C. On the other hand, for G_2 in Figure 2.4, consider $A = \{x_1, x_3, x_5\}$, $B = \{x_4, x_6\}$, and $C = \{x_2\}$. A, B, and C are mutually exclusive. A is d-separated from B given C written as $A \perp_{G_2} B \mid C$ ($x_1, x_3, x_5 \perp_{G_2} x_4, x_6 \mid x_2$) because each path from a node in A to a node in B is blocked. First, a path from node $x_5 \in A$ to node in $x_4 \in B$ is blocked by node x_7 where x_7 is a $head - to - head$ node and x_7 is not in C. Second, a path from node $x_3 \in A$ to node in $x_4 \in B$ is blocked by node x_2 where x_2 is a $tail - to - tail$ node and x_2 is in C. Finally, a path from node $x_1 \in A$ to node in $x_2 \in C$ is blocked by node x_3 where x_3 is a $head - to - head$ node and x_3 is not in C.

2.3.6 Global Directed Markov Property

A distribution P satisfies the global Markov property with respect to a directed acyclic graph G if

$$A \perp_G B \mid C \Rightarrow A \perp\!\!\!\perp B \mid C \tag{2.12}$$

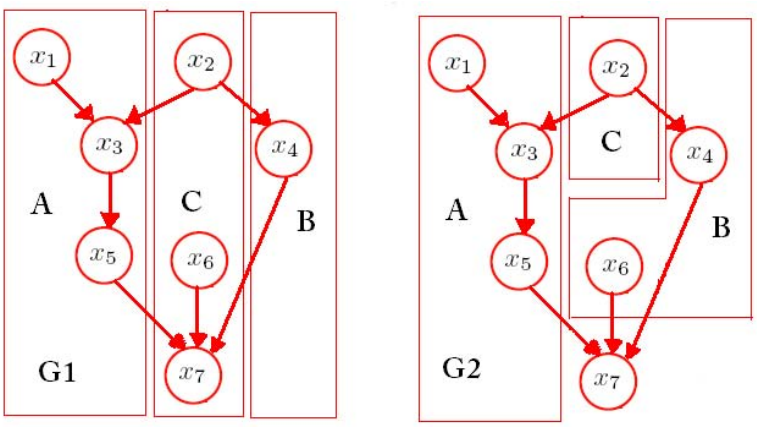


Figure 2.4: In G (G_1 or G_2), A is separated from B by C, written as $A \perp_G B \mid C$.

For example, in G_2 of Figure 2.4, $\{x_1, x_3, x_5\} \perp\!\!\!\perp \{x_4, x_6\} \mid x_2$

2.3.7 Theorem

Let (F) denote the factorization property. Let (G) denote the global directed Markov property. Let (L) be the local directed Markov property. Lauritzen [16] proves that it is always true for a directed acyclic graph G that

$$(F) \Longleftrightarrow (G) \Longleftrightarrow (L) \tag{2.13}$$

2.4 Convert Directed Conditional Independence Graph into (Undirected) Conditional Independence Graph

In a directed acyclic graph, the conditional probability associated with each node i and its parents $\pi(i)$ is $P(i|\pi(i))$. Moreover, we determine the conditional dependences between random variables by the d-separation theorem. For example, in G_1 of Figure 2.5, by the d-separation theorem, we find that

$$x_5 \perp_{G_1} x_6 \mid x_3, x_4.$$

By the global directed Markov property, we have $x_5 \perp\!\!\!\perp x_6 \mid x_3, x_4$. Moreover, it is not true that $x_5 \perp_{G_1} x_6 \mid x_4$ and nor that $x_5 \perp_{G_1} x_6$. These conditional independences can be also determined by

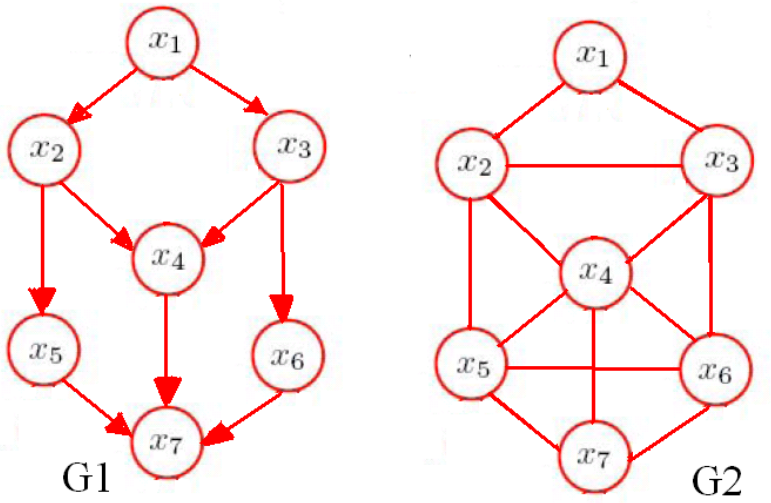


Figure 2.5: Conditional independences from G_1 and its morlized graph G_2

$x_5 \perp\!\!\!\perp x_6 \mid x_3, x_4$ with respect to G_1. However $x_5 \not\perp\!\!\!\perp x_6 \mid x_3, x_4$ with respect to G_2

constructing an undirected graph.

2.4.1 Construct a Moralized Graph

Given a directed graph G, we want to study the independences and conditional independences in the corresponding undirected graph H. We construct H by the following procedures. If a node has more than one parent in G, we add additional undirected links between all pairs of parents for the node. Then, we drop the arrows on the original links of G in H. In this case, H is called the moralized graph of G. From the previous example, G_2 of Figure 2.5 is the moralized graph of G_1. Moreover, G_2' in Figure 2.6 is an moralized graph of a subgraph of G_2 (The subgraph is constructed by considering x_3, x_4, x_5, x_6 and the ancestors in G_1). From G_2', we can easily determine that $x_5 \perp_{G_2'} x_6 \mid x_3, x_4$, $x_5 \not\perp_{G_2'} x_6 \mid x_4$, and $x_5 \not\perp_{G_2'} x_6$.

In addition, if a moralized graph is not a triangulated graph, we add edges to make it a triangulated graph. For example, in Figure 2.7, G_1 is the original directed graph, G_2 is the moralized graph of G_1. G_2 is not a triangulated graph because a cycle $x_2 - x_4 - x_5 - x_3 - x_2$ of four nodes has no a chord. G_3 is a triangulated graph after we add the edge $\{x_2, x_5\}$ in G_2.

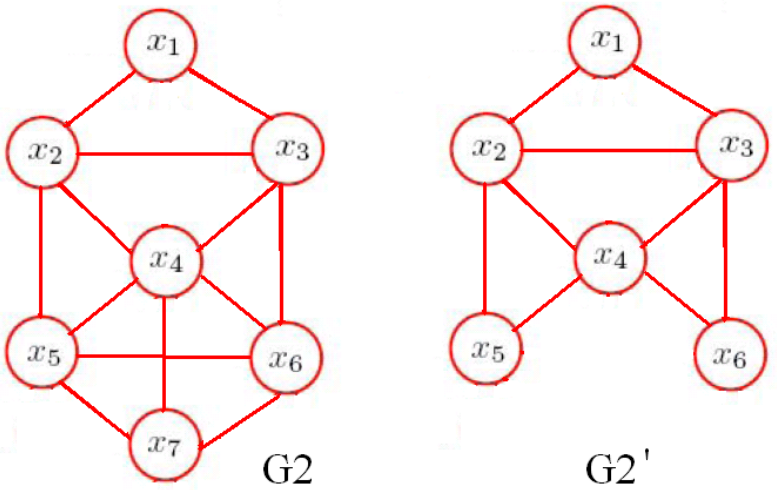


Figure 2.6: Conditional independences from an induced subgraph G_2' of G_2
$x_5 \perp_{G_2'} x_6 \mid x_3, x_4$; $x_5 \not\perp_{G_2'} x_6 \mid x_4$; and $x_5 \not\perp_{G_2'} x_6$

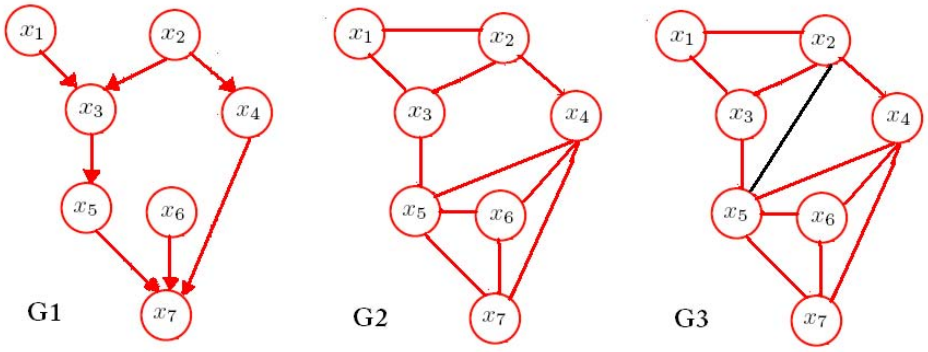


Figure 2.7: By adding the edge $\{x_2, x_5\}$ in G_2, a triangulated Graph G_3 is constructed.

2.4.2 Definition

A directed acyclic graph $G = (V, E)$ is moral if for every $a \rightarrow b \leftarrow c$, $a, b, c \in V$, there is a directed edge between a and c. For example, G_1 in Figure 2.8 is a moral graph.

2.4.3 I-map

Let P be probability distribution. Let $G = (V, E)$ be the conditional independence graph of P. Let A, B, and C be disjoint subsets of V. We say that G is an I-map of P if P has the global Markov property $A \perp_G B \mid C \Longrightarrow A \perp\!\!\!\perp B \mid C$ defined in Section 2.2.2.

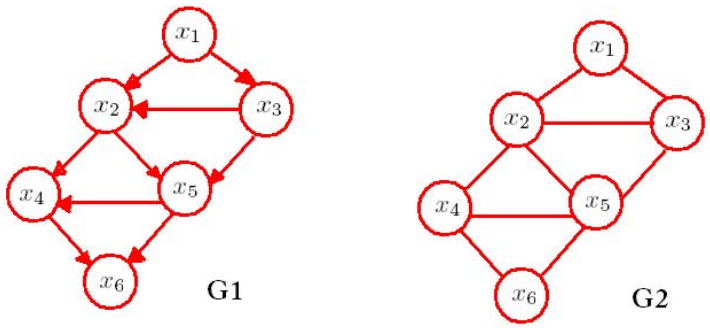


Figure 2.8: G_1 is a moral graph and G_2 is the moralized graph of G_1.

Minimal I-map

We say that a graph G is a minimal I-map of P if it is no longer an I-map if any edge is deleted.

Perfect I-map

We say that a graph G is a perfect map of P if

$$A \perp_G B \mid C \Longleftrightarrow A \perp\!\!\!\perp B \mid C.$$

2.4.4 Theorem

Let P be a probability distribution represented by a directed acyclic graph G. G is moral. If $M(G)$ is the moralized graph of G, then $M(G)$ is a perfect I-map of P.

Proof

By Section 2.4.3, $M(G)$ is a perfect I-map of P means that $A \perp B \mid_{M(G)} C \Longleftrightarrow A \perp\!\!\!\perp B \mid C$. Let $I(G)$ represent a set of independences and conditional independences with respect to G. Let $I(M(G))$ represent a set of independences and conditional independences respect to $M(G)$. Because G is an I-map and $M(G)$ is also an I-map, so, to show the theorem, we need to show $I(G) = I(M(G))$. In order to show $I(G) = I(M(G))$, we need to show (1), $I(M(G)) \subseteq I(G)$, then we show (2), $I(G) \subseteq I(M(G))$. Let H be $M(G)$.

To show (1), by the construction of H from $G = (V, E)$, we have $H = (V, E')$, where $E' = E_1 + E_2$, E_1 is a set of edges in G without directions. E_2 is a set of edges that connects parents of any pair a and b, $a, b \in V$. We have $E \subseteq E'$, where E represents a set of relationships in G. By adding these edges, we will have more relationships. This means that we may lose some independences and conditional independences in H. Therefore, $I(H) \subseteq I(G) \Rightarrow I(M(G)) \subseteq I(G)$.

To show (2), assume that there is an conditional independence $A \perp\!\!\!\perp B \mid C \in I(G)$ but it is not in $I(H)$. Thus, there must exist at least one trial from A to B in H that is active given C. Let consider a trial that is minimal (without shortcuts). By the definition of a moral graph, we know that H and G have the same edges. The trial must exist in G. By the assumption ($A \perp\!\!\!\perp B \mid C \in I(G)$ but it is not in $I(H)$), this means A to B can not be active in G given C. By the d-separation theorem and rule 3 in Section 2.3.4, the trial in G must be: $A \rightarrow \ldots \rightarrow x \rightarrow z \leftarrow y \leftarrow \ldots \leftarrow B$, where $x, y, z \notin C$ are nodes associating with random variables in G. However, G is moral, we must have an edge between X and Y. Figure 2.9 shows this. Therefore, the trial is not minimal. This contradicts our assumption. Therefore, $I(G) \subseteq I(H) \Rightarrow I(G) \subseteq I(M(G))$.

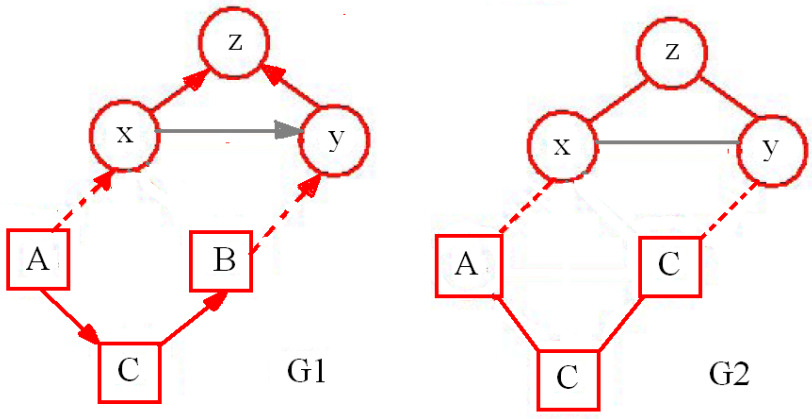


Figure 2.9: The trial in G_1 must be: $A \rightarrow \ldots \rightarrow x \rightarrow z \leftarrow y \leftarrow \ldots \leftarrow B$.
A (B or C) is associating with a set of random variables while x (y or z) is associating with an individual random variable.

2.4.5 Proposition

If $I(M(G)) = I(G)$, then the probability distribution of $M(G)$ over all variables is the same distribution as G over all variables.

For example, in Figure 2.8, G_1 is a directed graph associated with a probabilistic graphical model. The joint probability distribution over all variables can be represented as a product of conditional probability distributions, one for each variable, conditioned on it parents. Moreover, G_1 is a moral graph and G_2 is the moralized graph of G_1. By the proposition, these two graphs have the same joint probability distribution over all variables.

The joint probability function associated with G_1 is:

$$\begin{aligned} & p(x_1, x_2, x_3, x_4, x_5, x_6) \\ = \; & p(x_1)p(x_2|x_1, x_3)p(x_3|x_1)p(x_4|x_2, x_5)p(x_5|x_2, x_3)p(x_6|x_4, x_5) \end{aligned}$$

The joint probability function associated with G_2 is:

$$\begin{aligned} & p(x_1, x_2, x_3, x_4, x_5, x_6) \\ = \; & \frac{p(x_1, x_2, x_3)p(x_2, x_3, x_5)p(x_2, x_4, x_5)p(x_4, x_4, x_6)}{p(x_2, x_3)p(x_2, x_5)p(x_4, x_5)} \\ = \; & p(x_1, x_2, x_3) \cdot \frac{p(x_2, x_3, x_5)}{p(x_2, x_3)} \cdot \frac{p(x_2, x_4, x_5)}{p(x_2, x_5)} \cdot \frac{p(x_4, x_5, x_6)}{p(x_4, x_5)} \\ = \; & p(x_1, x_2, x_3)p(x_5|x_2, x_3)p(x_4|x_2, x_5)p(x_6|x_4, x_5) \\ = \; & p(x_2|x_1, x_3)p(x_1, x_3)p(x_4|x_2, x_5)p(x_5|x_2, x_3)p(x_6|x_4, x_5) \\ = \; & p(x_2|x_1, x_3)p(x_3|x_1)p(x_1)p(x_4|x_2, x_5)p(x_5|x_2, x_3)p(x_6|x_4, x_5) \end{aligned}$$

However, very few directed graphs are moral. If a directed acyclic graph is not a moral graph, in order to convert it into an undirected graph, we add additional undirected edges between all pairs of parents for a node and ignore the arrows to make a moralized graph for the directed graph. Some independences or conditional independences of the directed graph will be lost in the undirected graph because the edges we have added. In the undirected graph, we can not obtain the same joint probability distribution as in the directed graph. First, for simplicity, we assume that the undirected graph is triangulated graph. By introducing some dependences and conditional

independences of the directed acyclic graph into the undirected graph, we can get the same joint probability distribution.

For example, in Figure 2.10, the joint probability function for G_1 is:

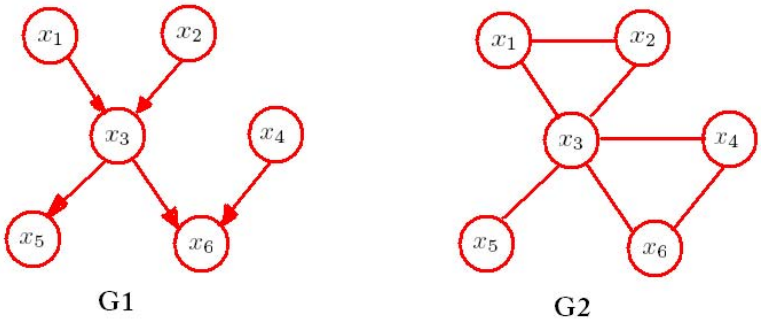


Figure 2.10: $x_1 \perp\!\!\!\perp x_2$ and $x_2 \perp\!\!\!\perp x_4$ are lost in G_2.

By introducing independences $x_1 \perp\!\!\!\perp x_2$ and $x_2 \perp\!\!\!\perp x_4$, we can get the same joint probabilitiy function with respect to G_1 and G_2.

$$p(x_1, x_2, x_3, x_4, x_5, x_6) = p(x_1)p(x_2)p(x_3|x_1, x_2)p(x_4)p(x_5|x_3)p(x_6|x_3, x_4) \tag{2.14}$$

The joint probability function for G_2 is:

$$\begin{aligned} & p(x_1, x_2, x_3, x_4, x_5, x_6) \\ = & \frac{p(x_1, x_2, x_3)p(x_3, x_4, x_6)p(x_3, x_5)}{p(x_3)p(x_3)} \\ = & \frac{p(x_3|x_1, x_2)p(x_1, x_2)p(x_6|x_3, x_4)p(x_3, x_4)p(x_5|x_3)}{p(x_3)} \\ = & \; p(x_3|x_1, x_2)p(x_1, x_2)p(x_6|x_3, x_4)p(x_4, x_3)p(x_5|x_3) \end{aligned} \tag{2.15}$$

Comparing Equation (2.14) with Equation (2.15), the joint probability associated with G_1 is different from that associated with G_2. If we form the joint probability associated with G_2 by introducing independences $x_1 \perp\!\!\!\perp x_2$ and $x_2 \perp\!\!\!\perp x_4$, we have

$$p(x_1, x_2, x_3, x_4, x_5, x_6) = \frac{p(x_3|x_1, x_2)p(x_1)p(x_2)p(x_6|x_3, x_4)p(x_3)p(x_4)p(x_5|x_3)}{p(x_3)}$$

$$= p(x_1)p(x_2)p(x_3|x_1,x_2)p(x_4)p(x_5|x_3)p(x_6|x_3,x_4) \tag{2.16}$$

Comparing Equation (2.14) with Equation (2.16), the joint probability associated with G_1 and the joint probability associated with G_2 are the same.

Figure 2.11 shows another example.

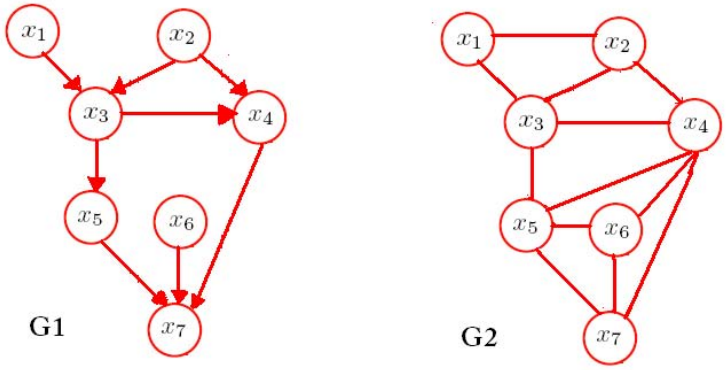


Figure 2.11: $x_1 \perp\!\!\!\perp x_2, x_4 \perp\!\!\!\perp x_5 \mid x_3$, and $x_6 \perp\!\!\!\perp x_4, x_5$ are lost in G_2.
By introducing independences and conditional indepencences $x_1 \perp\!\!\!\perp x_2, x_4 \perp\!\!\!\perp x_5 \mid x_3$, and $x_6 \perp\!\!\!\perp x_4, x_5$, we can get the same joint probabiltiy function with respect to G_1 and G_2.

The joint probability function associated with G_1 is:

$$\begin{aligned} & p(x_1,x_2,x_3,x_4,x_5,x_6,x_7) \\ = \; & p(x_1)p(x_2)p(x_3|x_1,x_2)p(x_4|x_2,x_3)p(x_5|x_3)p(x_6)p(x_7|x_4,x_5,x_6) \end{aligned} \tag{2.17}$$

The joint probability function associated with G_2 is:

$$\begin{aligned} & p(x_1,x_2,x_3,x_4,x_5,x_6,x_7) \\ = \; & \frac{p(x_1,x_2,x_3)p(x_2,x_3,x_4)p(x_3,x_4,x_5)p(x_4,x_5,x_6,x_7)}{p(x_2,x_3)p(x_3,x_4)p(x_4,x_5)} \\ = \; & p(x_3|x_1,x_2)p(x_1,x_2)p(x_4|x_2,x_3)p(x_5|x_3,x_4)\frac{p(x_7|x_4,x_5,x_6)p(x_4,x_5,x_6)}{p(x_4,x_5)} \\ = \; & p(x_3|x_1,x_2)p(x_1,x_2)p(x_4|x_2,x_3)p(x_5|x_3,x_4)p(x_7|x_4,x_5,x_6)p(x_6|x_4,x_5) \end{aligned} \tag{2.18}$$

Comparing Equation (2.17) with Equation (2.18), the joint probability associated with G_1 is different

from that associated with G_2. If we form the joint probability associated with G_2 by introducing independences and conditional independences $x_1 \perp\!\!\!\perp x_2$, $x_4 \perp\!\!\!\perp x_5|x_3$, and $x_6 \perp\!\!\!\perp x_4, x_5$, we have

$$\begin{aligned} & p(x_1, x_2, x_3, x_4, x_5, x_6, x_7) \\ = \; & p(x_3|x_1, x_2)p(x_1)p(x_2)p(x_4|x_2, x_3)p(x_5|x_3)\frac{p(x_7|x_4, x_5, x_6)p(x_4)p(x_5)p(x_6)}{p(x_4)p(x_5)} \\ = \; & p(x_3|x_1, x_2)p(x_1)p(x_2)p(x_4|x_2, x_3)p(x_5|x_3)p(x_7|x_4, x_5, x_6)p(x_6) \\ = \; & p(x_1)p(x_2)p(x_3|x_1, x_2)p(x_4|x_2, x_3)p(x_5|x_3)p(x_6)p(x_7|x_4, x_5, x_6) \end{aligned} \tag{2.19}$$

Comparing Equation (2.18) with Equation (2.19), the joint probability associated with G_1 and the joint probability associated with G_2 are the same.

In the following example, we consider the case that the moralized graph of G is not a triangulated graph. By adding edges, we make the moralized graph into a triangulated graph. Still, by introducing some dependences and conditional independences of the directed acyclic graph into the undirected graph, we get the same joint probability distribution in the corresponding undirected graph.

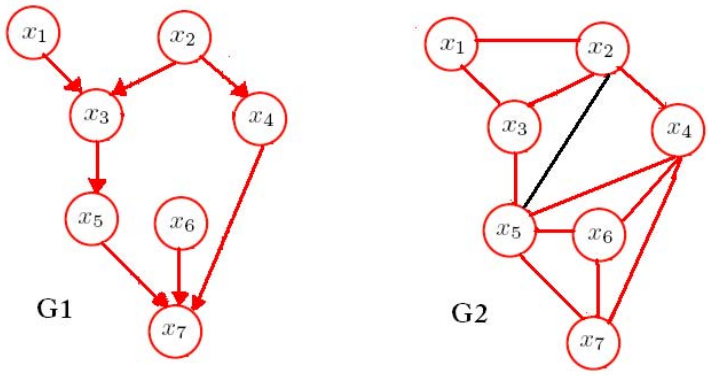


Figure 2.12: $x_1 \perp\!\!\!\perp x_2$, $x_4 \perp\!\!\!\perp x_5 \mid x_2$, $x_5 \perp\!\!\!\perp x_2 \mid x_3$, and $x_6 \perp\!\!\!\perp x_4, x_5$ are lost in G_2.
By introducing independences and conditional indepencences $x_1 \perp\!\!\!\perp x_2$, $x_4 \perp\!\!\!\perp x_5 \mid x_2$, $x_5 \perp\!\!\!\perp x_2 \mid x_3$, and $x_6 \perp\!\!\!\perp x4, x_5$, we can get the same joint probabiltiy function with respect to G_1 and G_2.

The joint probability function for G_1 is:

$$p(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$$

$$= p(x_1)p(x_2)p(x_3|x_1,x_2)p(x_4|x_2)p(x_5|x_3)p(x_6)p(x_7|x_4,x_5,x_6) \tag{2.20}$$

The joint probability function for G_2 is:

$$\begin{aligned} & p(x_1,x_2,x_3,x_4,x_5,x_6,x_7) \\ = & \frac{p(x_1,x_2,x_3)p(x_2,x_3,x_5)p(x_2,x_4,x_5)p(x_4,x_5,x_6,x_7)}{p(x_2,x_3)p(x_2,x_5)p(x_4,x_5)} \\ = & p(x_3|x_1,x_2)p(x_1,x_2)p(x_5|x_2,x_3)p(x_4|x_2,x_5)\frac{p(x_7|x_4,x_5,x_6)p(x_4,x_5,x_6)}{p(x_4,x_5)} \\ = & p(x_3|x_1,x_2)p(x_1,x_2)p(x_5|x_2,x_3)p(x_4|x_2,x_5)p(x_7|x_4,x_5,x_6)p(x_6|x_4,x_5) \end{aligned} \tag{2.21}$$

Comparing Equation (2.20) with Equation (2.21), the joint probability associated with G_1 is different from that associated with G_2. If we introduce the following independences and conditional independences $x_1 \perp\!\!\!\perp x_2$, $x_4 \perp\!\!\!\perp x_5 \mid x_2$, $x_5 \perp\!\!\!\perp x_2 \mid x_3$, and $x_6 \perp\!\!\!\perp x_4, x_5$, we obtain

$$\begin{aligned} & p(x_1,x_2,x_3,x_4,x_5,x_6,x_7) \\ = & p(x_3|x_1,x_2)p(x_1)p(x_2)p(x_5|x_3)p(x_4|x_2)\frac{p(x_7|x_4,x_5,x_6)p(x_4)p(x_5)p(x_6)}{p(x_4)p(x_5)} \\ = & p(x_3|x_1,x_2)p(x_1)p(x_2)p(x_5|x_3)p(x_4|x_2)p(x_7|x_4,x_5,x_6)p(x_6) \\ = & p(x_1)p(x_2)p(x_3|x_1,x_2)p(x_4|x_2)p(x_5|x_3)p(x_6)p(x_7|x_4,x_5,x_6) \end{aligned}$$

Comparing Equation (2.20) with Equation (2.22), the joint probability associated with G_1 and the joint probability associated with G_2 are the same.

Figure 2.13 shows a more complicated example. By adding two edges $\{x_5, x_6\}$ and $\{x_6, x_7\}$. we obtain the moralized graph for G_1. However, the moralized graph is not triangulated graph. By adding edges $\{x_2, x_3\}$, $\{x_3, x_4\}$, $\{x_2, x_6\}$, $\{x_4, x_6\}$, $\{x_2, x_4\}$, we make G_2 a triangulated graph.

The joint probability function for G_1 is:

$$\begin{aligned} & p(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9) \\ = & p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_1)p(x_5|x_2)p(x_6|x_3)p(x_7|x_4)p(x_8|x_6,x_7)p(x_9|x_5,x_6) \end{aligned} \tag{2.22}$$

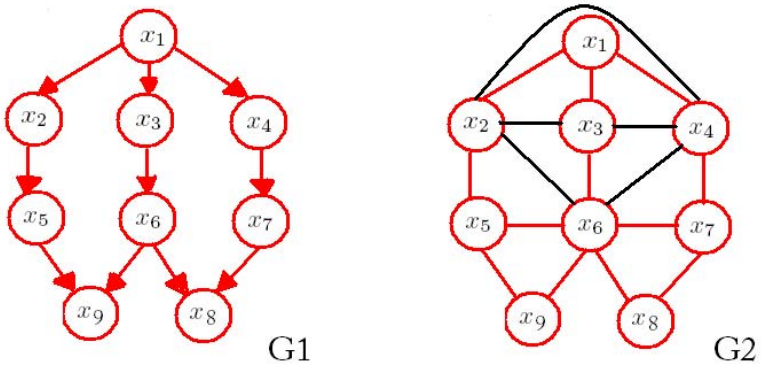


Figure 2.13: $x_2 \perp\!\!\!\perp x_3, x_4 \mid x_1, x_3 \perp\!\!\!\perp x_4 \mid x_1, x_6 \perp\!\!\!\perp x_4 \mid x_3, x_7 \perp\!\!\!\perp x_6 \mid x_4$, and $x_5 \perp\!\!\!\perp x_6 \mid x_2$ are lost in G_2. By introducing independences and conditional indepencences $x_2 \perp\!\!\!\perp x_3, x_4 \mid x_1, x_3 \perp\!\!\!\perp x_4 \mid x_1, x_6 \perp\!\!\!\perp x_4 \mid x_3, x_7 \perp\!\!\!\perp x_6 \mid x_4$, and $x_5 \perp\!\!\!\perp x_6 \mid x_2$, we can get the same joint probabiltiy function with respect to G_1 and G_2.

The joint probability function for G_2 is:

$$\begin{aligned} & p(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9) \\ = \; & \frac{p(x_1, x_2, x_3, x_4)p(x_3, x_4, x_6)p(x_4, x_6, x_7)p(x_6, x_7, x_8)p(x_5, x_6, x_9)p(x_2, x_5, x_6)}{p(x_3, x_4)p(x_4, x_6)p(x_6, x_7)p(x_6)p(x_5, x_6)} \\ = \; & p(x_2, x_3, x_4|x_1)p(x_1)p(x_8|x_6, x_7)p(x_9|x_5, x_6)p(x_6|x_3, x_4)p(x_7|x_4, x_6)p(x_5|x_2, x_6) \end{aligned} \tag{2.23}$$

Comparing Equation (2.22) with Equation (2.23), the joint probability associated with G_1 is different from that associated with G_2. If we introduce the following conditional independences $x_2 \perp\!\!\!\perp x_3, x_4 \mid x_1, x_3 \perp\!\!\!\perp x_4 \mid x_1, x_6 \perp\!\!\!\perp x_4 \mid x_3, x_7 \perp\!\!\!\perp x_6 \mid x_4$, and $x_5 \perp\!\!\!\perp x_6 \mid x_2$, we obtain the joint probability of G_2 as

$$\begin{aligned} & p(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9) \\ = \; & p(x_2|x_1)p(x_3|x_1)p(x_4|x_1)p(x_1)p(x_8|x_6, x_7)p(x_9|x_5, x_6)p(x_6|x_3)p(x_7|x_4)p(x_5|x_2) \\ = \; & p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_1)p(x_5|x_2)p(x_6|x_3)p(x_7|x_4)p(x_8|x_6, x_7)p(x_9|x_5, x_6) \end{aligned} \tag{2.24}$$

2.5 Some Probabilistic Graphical Models

2.5.1 A Naive Bayes (NB)

In Figure 2.14, G_1 is a directed acyclic graph associated with a naive Bayes model where G_2 is the corresponding moralized graphs. The joint probability over all variables associated with G_1 is:

$$p(X, Y) = \prod_{i=1}^{5} p(y_i|x_i)p(x_i)$$

The joint probability over all variables associated with G_2 is:

$$\begin{aligned} & p(X, Y) \\ = & \frac{p(x_1, y_1)p(x_2, y_2)p(x_3, y_3)p(x_4, y_4)p(x_5, y_5)}{1 \cdot 1 \cdot 1 \cdot 1 \cdot 1} \\ = & \prod_{i=1}^{5} p(x_i, y_i) \\ = & \prod_{i=1}^{5} p(y_i|x_i)p(x_i) \end{aligned}$$

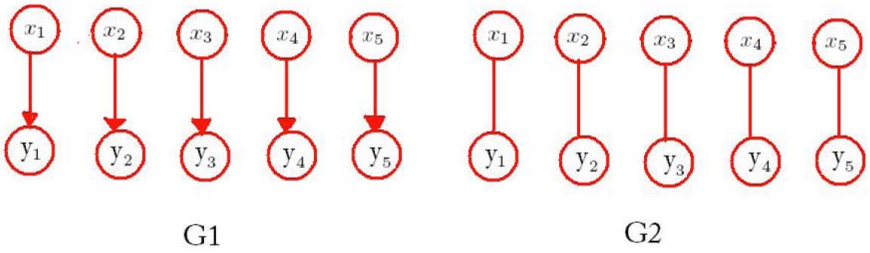


Figure 2.14: G_1 is associated with a Naive Bayes model and G_2 is the moralized graph of G_1.

2.5.2 A Hidden Markov Model (HMM)

In Figure 2.15, G_1 is a directed graph associated with a hidden Markov model while G_2 is the moralized graph of G_1. From the theorem in Section 2.4, G_1 is a moral graph. Thus, $I(G_1) = I(G_2)$. From the proposition in Section 2.4.5, the joint probability over all variables in G_1 equals to the joint probability over all variables in G_2.

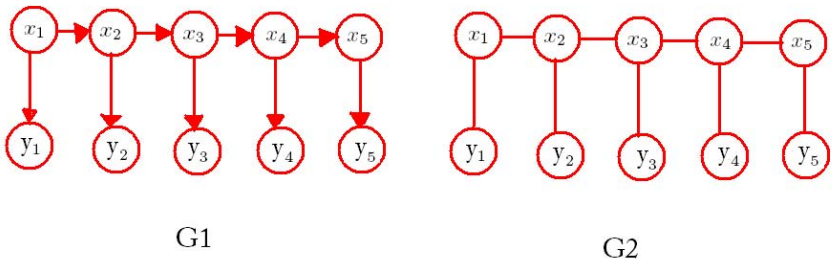


Figure 2.15: G_1 is associated with a hidden Markov model and G_2 the moralized graph of G_1

.

The joint probability distribution over all variables for G_1 in Figure 2.15 is:

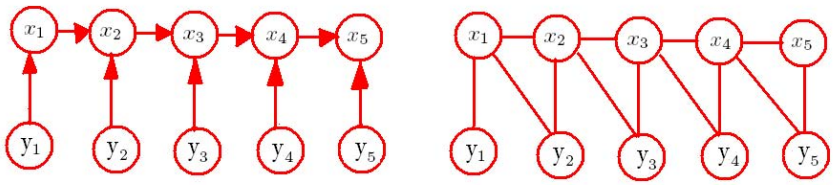
$$
\begin{aligned}
& p(X,Y) \\
= \; & p(x_1)p(y_1|x_1)p(x_2|x_1)p(y_2|x_2)p(x_3|x_2)p(y_3|x_3)p(x_4|x_3)p(y_4|x_4)p(x_5|x_4)p(y_5|x_5) \\
= \; & p(x_1)\prod_{i=2}^{5} p(x_i|x_{i-1})\prod_{j=1}^{5} p(y_j|x_j) \\
= \; & \prod_{i=1}^{5} p(x_i|x_{i-1})p(y_i|x_i) \qquad \textit{where } p(x_1|x_0) = p(x_1)
\end{aligned}
$$

The joint probability distribution over all variables for G_2 in Figure 2.15 is:

$$
\begin{aligned}
& p(X,Y) \\
= \; & \frac{p(x_1,x_2)p(x_2,x_3)p(x_3,x_4)p(x_4,x_5)p(x_1,y_1)p(x_2,y_2)p(x_3,y_3)p(x_4,y_4)p(x_5,y_5)}{p(x_1)p(x_2)p(x_2)p(x_3)p(x_3)p(x_4)p(x_4)p(x_5)} \\
= \; & \frac{p(x_1,x_2)p(x_2,x_3)p(x_3,x_4)p(x_4,x_5)}{p(x_2)p(x_3)p(x_4)} \cdot \frac{p(x_1,y_1)p(x_2,y_2)p(x_3,y_3)p(x_4,y_4)p(x_5,y_5)}{p(x_1)p(x_2)p(x_3)p(x_4)p(x_5)} \\
= \; & \frac{p(x_1)p(x_1,x_2)p(x_2,x_3)p(x_3,x_4)p(x_4,x_5)}{p(x_1)p(x_2)p(x_3)p(x_4)} \cdot \prod_{j=1}^{5} p(y_j|x_j) \\
= \; & \prod_{i=1}^{5} p(x_i|x_{i-1})p(y_i|x_i) \qquad \textit{where } p(x_1|x_0) = p(x_1)
\end{aligned}
$$

2.5.3 A Maximum Entropy Markov Model (MEMM)

In Figure 2.16, the graph $G_1 = (V_1, E_1)$ is associated with a maximum entropy Markov model while G_2 is the moralized graph of G_1. G_1 is not a moral graph because for each $a \rightarrow b \leftarrow c$, there is not an edge between a and c, for $a, b, c \in V_1$. Thus, $I(G_2) \subseteq I(G_1)$. The joint probability over all variables in G_1 will not equal to the joint probability over all variables in G_2. However, if introduce some independences associated with the joint probability of MEMM, we obtain the same probability distribution.



G1 G2

Figure 2.16: G_1 is associated with a maximum entropy model and G_2 is the moralized graph of G_1

.

The joint probability distribution over all variables for G_1 in Figure 2.16 is:

$$
\begin{aligned}
& p(X, Y) \\
= \; & p(y_1)p(y_2)p(y_3)p(y_4)p(y_5)p(x_1|y_1)p(x_2|x_1, y_2)p(x_3|x_2, y_3)p(x_4|x_3, y_4)p(x_5|x_4, y_5) \\
= \; & p(x_1|y_1) \prod_{i=1}^{5} p(y_i) \prod_{j=2}^{5} p(x_j|x_{j-1}y_i) \\
= \; & \prod_{i=1}^{5} p(x_i|x_{i-1}, y_i)p(y_i) \qquad \textit{where } p(x_1|x_0, y_1) = p(x_1|y_1)
\end{aligned}
$$

The joint probability distribution over all variables for G_2 in Figure 2.16 is:

$$
\begin{aligned}
p'(X, Y) &= \frac{p(x_1, y_1)p(x_1, x_2, y_2)p(x_2, x_3, y_3)p(x_3, x_4, y_4)p(x_4, x_5, y_5)}{p(x_1)p(x_2)p(x_3)p(x_4)} \\
&= \frac{p(x_1, y_1)p(x_1, x_2, y_2)p(x_2, x_3, y_3)p(x_3, x_4, y_4)p(x_4, x_5, y_5)p(y_1)p(y_2)p(y_3)p(y_4)p(y_5)}{p(x_1)p(x_2)p(x_3)p(x_4)p(y_1)p(y_2)p(y_3)p(y_4)p(y_5)}
\end{aligned}
$$

$$
\begin{aligned}
p''(X,Y) &= \frac{p(x_1,y_1)p(x_1,x_2,y_2)p(x_2,x_3,y_3)p(x_3,x_4,y_4)p(x_4,x_5,y_5)p(y_1)p(y_2)p(y_3)p(y_4)p(y_5)}{p(y_1)p(x_1,y_2)p(x_2,y_3)p(x_3,y_4)p(x_4,y_5)} \\
&= p(x_1|y_1)\prod_{i=1}^{5} p(y_i)\prod_{j=2}^{5} p(x_j|x_{j-1}y_i) \\
&= \prod_{i=1}^{5} p(x_i|x_{i-1},y_i)p(y_i) \qquad \textit{where } p(x_1|x_0,y_1) = p(x_1|y_1)
\end{aligned}
$$

Clearly, $p'(X,Y)$ and $p''(X,Y)$ are not equal. However, if we use the independences of G_1 $x_1 \perp\!\!\!\perp y_2$, $x_2 \perp\!\!\!\perp y_3$, $x_3 \perp\!\!\!\perp y_4$, and $x_4 \perp\!\!\!\perp y_5$, then $p'(X,Y)$ will equal to $p''(X,Y)$. We have $p'(X,Y) = p''(X,Y) = p(X,Y)$

2.5.4 A Conditional Random Field (CRF)

In this case, the probabilistic graphical model for *CRF* associates the graph G_2 in Figure 2.15. Therefore, the directed graph can be either G_1 in the same figure, or a graph in which each edge has a reversed direction. However, it can not be a graph as the same as G_1 in Figure 2.16. Thus, the joint probability distribution over all variables is:

$$
\begin{aligned}
& p(X,Y) \\
= \; & p(x_1)p(y_1|x_1)p(x_2|x_1)p(y_2|x_2)p(x_3|x_2)p(y_3|x_3)p(x_4|x_3)p(y_4|x_4)p(x_5|x_4)p(y_5|x_5) \\
= \; & \prod_{i=1}^{5} p(x_i|x_{i-1})p(y_i|x_i) \qquad \textit{where } p(x_1|x_0) = p(x_1)
\end{aligned}
$$

2.5.5 The Data Sequence Dependence Model (DSDM)

In Figure 2.17, G_2 is an undirected graph associated with a probabilistic graphical model that it will be developed in Chapter 3. G_1 is one of the directed graphs corresponding with G_1. In this case, G_1 is a moral graph. Therefore $I(G_1) = I(G_2)$. By the proposition in Section 2.4.5, the joint probabilities over all variables corresponding with these two graphs will be the same.

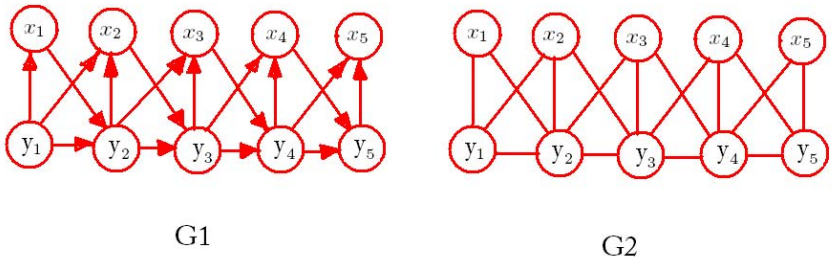


Figure 2.17: G_2 is associated with the data sequence independence model and G_1 is the corresponding directed acyclic graph of G_2.

The joint probability distribution over all variables in G_1 of Figure 2.17 is:

$$
\begin{aligned}
& p(X,Y) \\
= \; & p(x_1|y_1)p(x_2|y_1,y_2)p(x_3|y_2,y_3)p(x_4|y_3,y_4)p(x_5|y_4,y_5) \\
& p(y_1)p(y_2|y_1,x_1)p(y_3|y_2,x_2)p(y_4|y_3,x_3)p(x_5|y_4,x_4) \\
= \; & \prod_{i=1}^{5} p(x_i|y_i,y_{i-1}) \cdot \prod_{j=1}^{5} p(y_j|y_{j-1},x_{j-1}) \\
& \textit{Where: } p(x_1|y_1,y_0) = p(x_1|y_1), \;\; p(y_1|y_0,x_0) = p(y_1) \\
= \; & \prod_{i=1}^{5} p(x_i|y_i,y_{i-1})p(y_i|y_{i-1},x_{i-1})
\end{aligned}
$$

The joint probability distribution over all variables in G_2 of Figure 2.17 is:

$$
\begin{aligned}
& p(X,Y) \\
= \; & \frac{p(x_1,y_1,y_2)p(x_2,y_1,y_2)p(x_2,y_2,y_3)p(x_3,y_2,y_3)}{p(y_1,y_2)p(x_2,y_2)p(y_2,y_3)p(x_3,y_3)} \cdot \\
& \frac{p(x_3,y_3,y_4)p(x_4,y_3,y_4)p(x_4,y_4,y_5)p(x_5,y_4,y_5)}{p(y_3,y_4)p(x_4,y_4)p(y_4,y_5)} \\
= \; & \frac{p(x_1,y_1,y_2)p(x_2,y_2,y_3)p(x_3,y_3,y_4)p(x_4,y_4,y_5)}{p(x_2,y_2)p(x_3,y_3)p(x_4,y_4)} \cdot \\
& \frac{p(x_2,y_1,y_2)p(x_3,y_2,y_3)p(x_4,y_3,y_4)p(x_5,y_4,y_5)}{p(y_1,y_2)p(y_2,y_3)p(y_3,y_4)p(y_4,y_5)} \\
= \; & \frac{p(y_1)p(x_1,y_1,y_2)p(x_2,y_2,y_3)p(x_3,y_3,y_4)p(x_4,y_4,y_5)}{p(x_1,y_1)p(x_2,y_2)p(x_3,y_3)p(x_4,y_4)} \cdot \\
& \frac{p(x_1,y_1)p(x_2,y_1,y_2)p(x_3,y_2,y_3)p(x_4,y_3,y_4)p(x_5,y_4,y_5)}{p(y_1)p(y_1,y_2)p(y_2,y_3)p(y_3,y_4)p(y_4,y_5)}
\end{aligned}
$$

$$
\begin{aligned}
&= \prod_{j=1}^{5} p(y_j|y_{j-1}, x_{j-1}) \cdot \prod_{i=1}^{5} p(x_i|y_i, y_{i-1}) \\
&\quad \textit{Where: } p(x_1|y_1, y_0) = p(x_1|y_1), \;\; p(y_1|y_0, x_0) = p(y_1) \\
&= \prod_{i=1}^{5} p(x_i|y_i, y_{i-1}) p(y_i|y_{i-1}, x_{i-1})
\end{aligned}
$$

Moreover, the joint probability distribution $p(X, Y)$ can be rewritten as:

$$
\begin{aligned}
& p(X, Y) \\
= \; & \frac{1}{p(y_1, y_2)p(y_2, y_3)p(y_3, y_4)p(y_4, y_5)} \\
& \frac{p(x_1, y_1, y_2)p(x_2, y_1, y_2)p(x_2, y_2, y_3)p(x_3, y_2, y_3)}{p(x_2, y_2)p(x_3, y_3)p(x_4, y_4)} \cdot \\
& p(x_3, y_3, y_4)p(x_4, y_3, y_4)p(x_4, y_4, y_5)p(x_5, y_4, y_5) \\
= \; & \frac{1}{p(y_1, y_2)p(y_2, y_3)p(y_3, y_4)p(y_4, y_5)} \\
& \frac{p(x_1, y_1)p(x_1, y_1, y_2)p(x_2, y_1, y_2)p(x_2, y_2, y_3)p(x_3, y_2, y_3)}{p(x_1, y_1)p(x_2, y_2)p(x_3, y_3)p(x_4, y_4)} \cdot \\
& p(x_3, y_3, y_4)p(x_4, y_3, y_4)p(x_4, y_4, y_5)p(x_5, y_4, y_5) \\
= \; & \frac{1}{p(y_1, y_2)p(y_2, y_3)p(y_3, y_4)p(y_4, y_5)} \frac{p(x_1, y_1, y_2)p(x_2, y_2, y_3)p(x_3, y_3, y_4)p(x_4, y_4, y_5)}{p(x_1, y_1)p(x_2, y_2)p(x_3, y_3)p(x_4, y_4)} \\
& p(x_1, y_1)p(x_2, y_1, y_2)p(x_3, y_2, y_3)p(x_4, y_3, y_4)p(x_5, y_4, y_5) \\
= \; & \frac{1}{p(y_1, y_2)p(y_2, y_3)p(y_3, y_4)p(y_4, y_5)} \prod_{i=1}^{4} p(y_{i+1}|x_i, y_i) \prod_{j=1}^{5} p(y_{j-1}, x_j, y_j) \\
& \textit{where: } p(y_0, x_1, y_1) = p(x_1, y_1) \\
= \; & \frac{1}{p(y_1)p(y_1, y_2)p(y_2, y_3)p(y_3, y_4)p(y_4, y_5)} \prod_{i=1}^{4} p(y_{i+1}|x_i, y_i) \prod_{j=1}^{5} p(y_{j-1}|x_j, y_j)p(y_j|x_j)p(x_j) \\
= \; & \frac{1}{p(y_1)p(y_1, y_2)p(y_2, y_3)p(y_3, y_4)p(y_4, y_5)} \prod_{i=1}^{5} p(y_{i+1}|x_i, y_i)p(y_{i-1}|x_i, y_i)p(y_i|x_i)p(x_i) \\
& \textit{where: } p(y_6|x_5, y_5) = 1
\end{aligned}
$$

Chapter 3

Developing a New Probabilistic Graphical Model

In this chapter, first, we discuss two types of economic gain functions. Then, we develop a new probabilistic graphical model called the data sequence dependence model (*DSDM*). We discuss the properties of *DSDM*. Moreover, we describe two other probabilistic graphical models. They are the context independent model (*CIM*) and the class sequence dependence model (*CSDM*). We discuss the time and memory complexities of these models.

3.1 Definition of Economic Gain Function

Let C be a set of classes. Let t be a true class identification from C. Let a be an assigned class from C. The economic gain e is a function, $e : C \times C \rightarrow R$. It must have the property that:

$$e(t,a) = \begin{cases} > 0, & \textit{when } t = a \\ \leq 0, & \textit{otherwise} \end{cases}$$

Thus, when $a = t$, $e(t,a)$ is positive which represents a profit consequence. When $a \neq t$, $e(t,a)$ is zero or negative which represents a zero profit or loss consequence. Therefore, e is a mechanism to evaluate a decision rule. Moreover, if $e(t,a) = 1$ when $a = t$ and 0 otherwise, the gain matrix is an identity gain matrix. For example, Table 3.1 is an identity gain matrix, where $\{\alpha, \beta, \gamma, \delta\}$ is a set

of classes, t is a true class category, and a is an assigned classes. An entry in the diagonal position represents $a = t$ where an entry in the off diagonal position represents $a \neq t$.

Table 3.1: An identity gain matrix

$t\backslash a$	α	β	γ	δ
α	1	0	0	0
β	0	1	0	0
γ	0	0	1	0
δ	0	0	0	1

3.1.1 Expected Economic Gain

Let $p(t, a)$ be the probability that the assigned class a equals to true class t. The expected economic gain is defined as:

$$E[e(t,a)] = \sum_{t \in C} \sum_{a \in C} e(t,a)p(t,a)$$

For example, Table 3.2 shows the probabilities of $p(t, a)$ and Table 3.3 shows the corresponding economic consequences $e(t, a)$, where $t, a \in \{\alpha, \beta, \gamma, \delta\}$. Based on these two tables, the expected economic gain $E[e(t, a)]$ equals to \$15.

$$\begin{aligned}
E[e] &= p(\alpha,\alpha)e(\alpha,\alpha) + p(\alpha,\beta)e(\alpha,\beta) + p(\alpha,\gamma)e(\alpha,\gamma) + p(\alpha,\delta)e(\alpha,\delta) \\
&\quad p(\beta,\alpha)e(\beta,\alpha) + p(\beta,\beta)e(\beta,\beta) + p(\beta,\gamma)e(\beta,\gamma) + p(\beta,\delta)e(\beta,\delta) \\
&\quad p(\gamma,\alpha)e(\gamma,\alpha) + p(\gamma,\beta)e(\gamma,\beta) + p(\gamma,\gamma)e(\gamma,\gamma) + p(\gamma,\delta)e(\gamma,\delta) \\
&\quad p(\delta,\alpha)e(\delta,\alpha) + p(\delta,\beta)e(\delta,\beta) + p(\delta,\gamma)e(\delta,\gamma) + p(\delta,\delta)e(\delta,\delta) \\
&= 0.5 * 20 + 0.2 * (-10) + 0.2 * 0 + 0.1 * (-5) + \\
&\quad 0.1 * (-1) + 0.7 * 10 + 0.1 * (-20) + 0.1 * (-2) + \\
&\quad 0.05 * 0 + 0.1 * (-2) + 0.8 * 8 + 0.05 * (-30) + \\
&\quad 0.1 * (-4) + 0.1 * (-5) + 0.2 * (-10) + 0.6 * 2
\end{aligned}$$

$$= \quad 7.5 + 4.7 + 4.7 - 2.1 = \$15$$

Table 3.2: A probability matrix $p(t, a)$

$t \backslash a$	α	β	γ	δ
α	$p(\alpha, \alpha) = 0.5$	$p(\alpha, \beta) = 0.2$	$p(\alpha, \gamma) = 0.2$	$p(\alpha, \delta) = 0.1$
β	$p(\beta, \alpha) = 0.1$	$p(\beta, \beta) = 0.7$	$p(\beta, \gamma) = 0.1$	$p(\beta, \delta) = 0.1$
γ	$p(\gamma, \alpha) = 0.05$	$p(\gamma, \beta) = 0.1$	$p(\gamma, \gamma) = 0.8$	$p(\gamma, \delta) = 0.05$
δ	$p(\delta, \alpha) = 0.1$	$p(\delta, \beta) = 0.1$	$p(\delta, \gamma) = 0.2$	$p(\delta, \delta) = 0.6$

Table 3.3: A gain matrix $e(t, a)$

$t \backslash a$	α	β	γ	δ
α	\$20	−\$10	0	−\$5
β	−\$1	\$10	−\$20	−\$2
γ	0	−\$2	\$8	−\$30
δ	−\$4	−\$5	−\$10	\$2

3.2 Expected Economic Gains for Probabilistic Graphical Models

Let C be a set of classes. Let $s =< s_1, \ldots, s_N >$ be a sequence of N symbols. Let $c =< c_1, ..., c_N >$ be a sequence of assigned classes for s, each $c_n \in C$. Let $c^T =< c_1^T, ..., c_N^T >$ be a sequence of true class classes for s, each $c_n^T \in C$. Let e be the economic gain function $e : C^N \times C^N \rightarrow \mathcal{R}$. Two types of economic gains are used by probabilistic graphical models.

3.2.1 Type One

$$e(c_1^T, \ldots, c_N^T, c_1, \ldots, c_N) = \sum_{i=1}^{N} e(c_i^T, c_i) \tag{3.1}$$

where

$$e(c_i^T, c_i) = \begin{cases} 1, & \textit{when } c_i^T = c_i \\ 0, & \textit{otherwise} \end{cases}$$

The economic gain of obtaining a sequence of assigned class categories $c_1, \ldots, c_N$ when we have a sequence of true class categories $c_1^T, \ldots, c_N^T$ equals to the sum of all the economic gains of each assigned class category. Moreover, it equals to the number of cases where $c_n = c_n^T$, $n = 1, \ldots, N$. For example, suppose we have a sequence $s_1, s_2, \ldots, s_{10}$ with a true category sequence $c_1^T, c_2^T, \ldots c_{10}^T$ and assigned category sequence $c_1, \ldots, c_N$. If $c_2 = c_2^T$, $c_5 = c_5^T$, $c_7 = c_7^T$, $c_9 = c_9^T$, and $c_{10} = c_{10}^T$, then the economic gain equals five, $e(c_1, \ldots c_{10}, c_1^T, \ldots, c_{10}^T) = 5$.

3.2.2 Type Two

$$e(c_1^T, \ldots, c_N^T, c_1, \ldots, c_N) = \begin{cases} 1, & \textit{when } c_1^T, \ldots, c_N^T = c_1, \ldots, c_N \\ 0, & \textit{otherwise} \end{cases} \tag{3.2}$$

The economic gain when the sequence of assigned class categories is $c_1, \ldots, c_N$ and the sequence of true class categories $c_1^T, \ldots, c_N^T$ equals to 1, for all class assignments, if each class assignment equals to its true class category. Otherwise, it equals to 0. For the previous example, even if we have five cases, $c_2 = c_2^T$, $c_5 = c_5^T$, $c_7 = c_7^T$, $c_9 = c_9^T$, and $c_{10} = c_{10}^T$, the economic gain still equals to zero, $e(c_1, \ldots c_{10}, c_1^T, \ldots, c_{10}^T) = 0$.

3.2.3 Expected Economic Gain of $c_1, \ldots, c_N$

The expected economic gain of assigning $s_1, \ldots, s_N$ to $c_1, \ldots, c_N$, where the sequence of the true class categories is $c_1^T, \ldots, c_N^T$ is defined in Equation (3.4).

$$E[e](c_1, \ldots, c_N, c_1^T, \ldots, c_N^T) = \sum_{c_1^T, \ldots, c_N^T} e(c_1^T, ..., c_N^T, c_1, ..., c_N) p(c_1^T, ..., c_N^T | s_1, ..., s_N) \tag{3.3}$$

For simplicity, we let $E[e](c_1, \ldots, c_N) = E[e](c_1, \ldots, c_N, c_1^T, \ldots, c_N^T)$.

$$E[e](c_1, \ldots, c_N) = \sum_{c_1^T, \ldots, c_N^T} e(c_1^T, ..., c_N^T, c_1, ..., c_N) p(c_1^T, ..., c_N^T | s_1, ..., s_N) \tag{3.4}$$

3.3 Expected Economic Gain of the First Type

From Equation (3.4) and Equation (3.1), we have:

$$\begin{aligned} E[e](c_1,\ldots,c_N) &= \sum_{c_1^T,\ldots,c_N^T}\sum_{n=1}^{N} e(c_n^T,c_n)p(c_1^T,...,c_N^T|s_1,...,s_N) \\ &= \sum_{n=1}^{N}\sum_{c_1^T,\ldots,c_N^T} e(c_n^T,c_n)p(c_1^T,...,c_N^T|s_1,...,s_N) \\ &= \sum_{n=1}^{N}\sum_{c_n^T} e(c_n^T,c_n) \sum_{c_1^T,\ldots,c_{n-1}^T,c_{n+1}^T,\ldots,c_N^T} p(c_1^T,...,c_N^T|s_1,...,s_N) \\ &= \sum_{n=1}^{N}\sum_{c_n^T} e(c_n^T,c_n)p(c_n^T|s_1,\ldots,s_N) \end{aligned}$$

Corresponding to each $e(c_n^T, c_n)$, $c_n^T, c_n \in C$, there is a gain matrix. Each entry of the matrix is either 0 or 1. When the gain matrix is diagonal, then $\sum_{c_n^T} e(c_n^T, c_n)p(c_n^T|s_1,\ldots,s_N) = e(c_n, c_n)\ p(c_n|s_1,\ldots,s_N)$. Therefore,

$$\begin{aligned} E[e](c_1,\ldots,c_N) &= \sum_{n=1}^{N} e(c_n,c_n)p(c_n|s_1,\ldots,s_N) \\ &= \sum_{n=1}^{N} p(c_n|s_1,\ldots,s_N) \end{aligned} \tag{3.5}$$

By Equation (3.5), the expected economic gain of a sequence of class assignments $c_1,\ldots,c_N$ for a sequence of symbols $s_1,\ldots,s_N$ when the true class categories for that symbol sequence are $c_1^T,\ldots,c_N^T$ is the summation over n of the conditional probabilities of a class assignment c_n over the given symbol sequence.

$$\begin{aligned} max(E[e](c_1,\ldots,c_N)) &= \max_{c_n\in C}\sum_{n=1}^{N} p(c_n|s_1,\ldots,s_N) \\ &= \sum_{n=1}^{N}\max_{c_n\in C} p(c_n|s_1,\ldots,s_N) \end{aligned} \tag{3.6}$$

Therefore, to find the assigned class sequence $c_1,...,c_N$ for the input sequence $s_1,..,s_N$ that maximizes the expected gain, we need to find an assigned class $c_n \in C$ for $s_1,\ldots,s_N$ that maximizes the

conditional probability function $p(c_n|s_1, ..., s_N)$, where $n = 1, \ldots, N$. This leads to our probability graphical models.

3.3.1 Discussions

The economic gain function in Equation (3.2) is employed by existed probabilistic graphical models, such as HMMs [1], MEMMs [2], or CRFs [3], The gain function specifies a gain of one if all the class assignments are correct and zero if one or more of the class assignments are wrong. No partial credit is given for some correct assignments. As a consequence, a gain of zero can be produced just because the assigned sequence has one incorrect assignment. We use the gain function in Equation (3.1). It has a partial gain for each correct assignment. We choose assignment $c_1, \ldots, c_N$ to maximize the expected gain under equation (3.6).

To find the assigned class sequence $c_1, \ldots, c_N$ for the input sequence $s_1, \ldots, s_N$ that maximizes the expected gain, we need to find an assigned class $c_n \in C$ that maximizes the joint probability function $p(c_n, s_1, \ldots, s_N)$, where the conditional independence graph is shown in Figure 3.1. This leads to a new probabilistic graphical model (see Section 3.4) then other two probabilistic graphical models (see Section 3.5) which are specializations of the one shown in Figure 3.1. We call them the data sequence dependence model (*DSDM*), the context independence model (*CIM*), and the class sequence dependence model (*CSDM*).

Each of the models can be represented by equations and a graph called the conditional indepen-

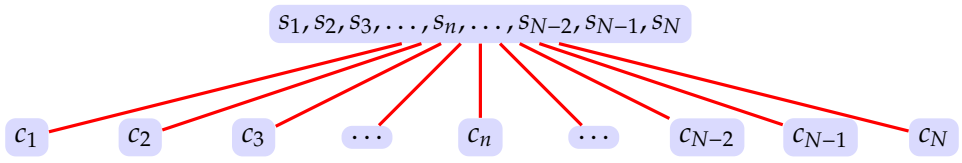


Figure 3.1: The expected gain leads to our graphical model.

dence graph. In the conditional independence graph, the nodes are the random variables. No edge between nodes x_i and x_j means that variable x_i is conditionally independent of variable x_j given all the remaining variables.

3.4 Data Sequence Dependence Model (DSDM)

Let $s =< s_1, \ldots, s_N >$ be a sequence of N symbols. Let C be a set of M classes, $C = \{C_1, \ldots, C_M\}$. Let $c =< c_1, ..., c_N >$ be a sequence of classes. Let e be the economic gain function $e : C^N \times C^N \rightarrow \mathcal{R}$. The data sequence dependence model is built under the following conditional independence assumptions:

- c_i is independent of c_j, $j \neq i$, given s_{i-1}, s_i, s_{i+1}

3.4.1 Conditional Independence Graph of *DSDM*

The conditional independence graph of the data sequence dependence model is shown in Figure 3.2

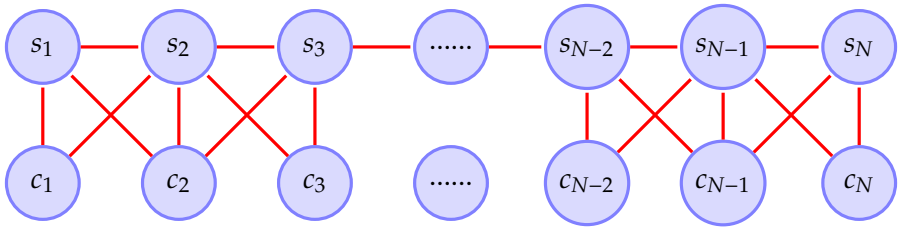


Figure 3.2: The conditional independence graph defining the data sequence dependence model.

3.4.2 Computing $p(c_1, \ldots, c_N, s_1, \ldots, s_N)$

Based on Chapter 2, the joint probability distribution $p(c_1, \ldots, c_N, s_1, \ldots, s_N)$ can be directly obtained from G. It is represented by Equation (3.7).

$$
\begin{aligned}
& p(c_1, ..., c_N, s_1, \ldots, s_N) \\
= & \frac{p(s_1, s_2, c_1)p(s_1, s_2, c_2)p(s_2, s_3, c_2)\ldots p(s_{N-1}, s_N, c_{N-1})p(s_{N-1}, s_N, c_N)}{p(s_1, s_2)p(s_2, s_3)\ldots p(s_{N-1}, s_N)p(s_2, c_2)\ldots p(s_N, c_N)} \\
= & \frac{\prod_{n=1}^{N-1} p(s_n, s_{n+1}, c_n)p(s_n, s_{n+1}, c_{n+1})}{\prod_{m=1}^{N-1} p(s_m, s_{m+1}) \prod_{m=2}^{N-1} p(s_m, c_m)} \\
= & \frac{1}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times \frac{\prod_{n=1}^{N-1} p(s_n, s_{n+1}, c_n)}{\prod_{m=2}^{N-1} p(s_m, c_m)} \times \prod_{n=1}^{N-1} p(s_n, s_{n+1}, c_{n+1})
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{p(s_1, s_2, c_1)}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times \frac{\prod_{n=2}^{N-1} p(s_n, s_{n+1}, c_n)}{\prod_{m=2}^{N-1} p(s_m, c_m)} \times \prod_{n=1}^{N-1} p(s_n, s_{n+1}, c_{n+1}) \\
&= \frac{p(s_1, s_2, c_1)}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times \prod_{n=2}^{N-1} p(s_{n+1}|s_n, c_n) \times \prod_{m=2}^{N} p(s_{m-1}, s_m, c_m) \\
&= \frac{p(s_1, s_2, c_1) p(s_{N-1}, s_N, c_N)}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times \prod_{n=2}^{N-1} p(s_{n+1}|s_n, c_n) \times \prod_{m=2}^{N-1} p(s_{m-1}, s_m, c_m) \\
&= \frac{p(s_1, s_2, c_1) p(s_{N-1}, s_N, c_N)}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times \prod_{n=2}^{N-1} p(s_{n+1}|s_n, c_n) p(s_{n-1}, s_n, c_n) \\
&= \frac{1}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times \prod_{n=1}^{N} p(s_{n+1}|s_n, c_n) p(s_{n-1}, s_n, c_n) \\
&\quad \textit{where} \quad p(s_0, s_1, c_1) = p(s_1, c_1) = 1, \quad p(s_{N+1}|s_N, c_N) = 1
\end{aligned}
$$

We define $\mathcal{M}_{s_1 \ldots s_N} = \frac{1}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})}$. Therefore

$$
\begin{aligned}
p(c_1, ..., c_N, s_1, \ldots, s_N) &= \underset{s_1 \ldots s_N}{\mathcal{M}} \prod_{n=1}^{N} p(s_{n+1}|s_n, c_n) p(s_{n-1}|s_n, c_n) p(s_n|c_n) p(c_n) \\
&\propto \prod_{n=1}^{N} p(s_{n+1}|s_n, c_n) p(s_{n-1}|s_n, c_n) p(s_n|c_n) p(c_n)
\end{aligned} \tag{3.7}
$$

Moreover, we can get $p(c_1, \ldots, c_N|s_1, \ldots, s_N)$ by applying the Bayes' theorem from Equation (3.7).

$$
\begin{aligned}
p(c_1, \ldots, c_N|s_1, \ldots, s_N) &= \frac{\prod_{n=1}^{N} p(s_{n+1}|s_n, c_n) p(s_{n-1}|s_n, c_n) p(s_n|c_n) p(c_n)}{\sum_{c \in C} \prod_{n=1}^{N} p(s_{n+1}|s_n, c_n) p(s_{n-1}|s_n, c_n) p(s_n|c_n) p(c_n)} \\
&\propto \prod_{n=1}^{N} p(s_{n+1}|s_n, c_n) p(s_{n-1}|s_n, c_n) p(s_n|c_n) p(c_n)
\end{aligned} \tag{3.8}
$$

3.4.3 Computing $max(E[e](c_1, \ldots, c_N))$

$$
\begin{aligned}
& p(c_n|s_1, \ldots, s_N) = p(c_n|s_{n-1}, s_n, s_{n+1}) \\
= \; & \frac{p(c_n, s_{n-1}, s_n, s_{n+1})}{p(s_{n-1}, s_n, s_{n+1})} \\
= \; & \frac{p(s_{n-1}|c_n, s_n, s_{n+1}) p(s_{n+1}|c_n, s_n) p(s_n|c_n) p(c_n)}{p(s_{n-1}, s_n, s_{n+1})}
\end{aligned}
$$

$$= \frac{p(s_{n-1}|c_n, s_n)p(s_{n+1}|c_n, s_n)p(s_n|c_n)p(c_n)}{p(s_{n-1}, s_n, s_{n+1})} \qquad since \;\; s_{n-1} \perp\!\!\!\perp s_{n+1} \mid c_n, s_n$$
$$= \frac{1}{p(s_{n-1}, s_n, s_{n+1})} p(s_{n-1}|c_n, s_n)p(s_{n+1}|c_n, s_n)p(s_n|c_n)p(c_n)$$

Therefore,

$$\begin{aligned}
& \max_{c_1,\ldots,c_N} (E[e](c_1, \ldots, c_N) \\
= & \max_{c_1\ldots,c_N} \sum_{n=1}^{N} p(c_n|s_1, \ldots, s_N) \\
= & \sum_{n=1}^{N} \max_{c_n \in C} p(c_n|s_1, \ldots, s_N) \\
= & \sum_{n=1}^{N} \max_{c_n \in C} p(c_n|s_{n-1}, s_n, s_{n+1}) \\
= & \sum_{n=1}^{N} \frac{1}{p(s_{n-1}, s_n, s_{n+1})} \max_{c_n \in C} p(s_{n-1}|c_n, s_n)p(s_{n+1}|c_n, s_n)p(s_n|c_n)p(c_n) \\
\propto & \sum_{n=1}^{N} \max_{c_n \in C} p(s_{n-1}|c_n, s_n)p(s_{n+1}|c_n, s_n)p(s_n|c_n)p(c_n) \qquad (3.9)
\end{aligned}$$

$$\underset{c_1,\ldots,c_N}{argmax}\, p(c_1, \ldots, c_N|s_1, \ldots, s_N) = \prod_{n=1}^{N} \underset{c_n}{argmax}\, p(s_{n-1}|c_n, s_n)p(s_{n+1}|c_n, s_n)p(s_n|c_n)p(c_n)$$

3.4.4 Probability Table

The probability table shows the amount of memory spaces that stores the probabilities of $p(s_{n-1}|p(c_n, s_n)$, $p(s_{n+1}|p(c_n, s_n)$, $p(s_n|c_n)$, and $p(c_n)$ derived from the training set. Let $S = \{S_1, \ldots, S_K\}$ be a set of possible symbols. Let $C = \{C_1, \ldots, C_M\}$ be a set of class categories. Let $|S| = K$ and $|C| = M$. The probability table is shown in Table 3.4.

Table 3.4: The probability table

Memory space	$p(s_{i-1}\|c_i, s_i)$	$p(s_{i+1}\|c_i, s_i)$	$p(s_i\|c_i)$	$p(c_i)$
$O(K^2M)$	$K \times K \times M$	$K \times K \times M$	$K \times M$	M

The probabilities for an unseen events, such as (s_{i-1}, s_i, c_i), (s_{i+1}, s_i, c_i) and (s_i, c_i) are not stored. They are computed based on Equation (6.5) discussed in Section 6.3.1.

3.4.5 Complexities

In order to compute the on-line complexity, we only consider the time and memory complexities of assigning a sequence of class categories for an input sequence. We ignore the complexities for estimating the probabilities of $p(s_{n-1}|p(c_n, s_n)$, $p(s_{n+1}|p(c_n, s_n)$, $p(s_n|c_n)$, and $p(c_n)$ from the training set. From Equation (3.9), let

$$g(s_{n-1}, s_n, s_{n+1}, c_n) = p(s_{n-1}|c_n, s_n)p(s_{n+1}|c_n, s_n)p(s_n|c_n)p(c_n).$$

Time Complexity

For each symbol s_n, we need to assign a c_n, such that, $g(s_{n-1}, s_n, s_{n+1}, c_n) \geq g(s_{n-1}, s_n, s_{n+1}, c'_n), c'_n \in C$. To compute $g(s_{n-1}, s_n, s_{n+1}, c_n)$, we need four multiplications. To obtain the maximum probability value, we require $M - 1$ comparisons. In the case of a sequence of N symbols, we need

$$T_c = 4 * N * (M - 1) = O(N * M) \tag{3.10}$$

Memory Complexity

Because the global maximum probability is determined by each local maximal probability, for a path of N symbols, we need to store the information of the current node. That is, we need only store M probability values in order to find the maximal probability value. Based on Section 3.4.4, for obtaining a probability table, we need to have $O(N^2M)$ memory space. Therefore,

$$M_c = M = O(M) + O(N^2M) = O(N^2M) \tag{3.11}$$

3.4.6 Properties

Property 1

The Markov blanket [1] for node c_n is s_{n-1}, s_n, s_{n+1}. Therefore, the Markov blanket property of the conditional independence graph tells us that class c_n is conditionally independent of $s_1, \ldots, s_{n-2}, s_{n+2}$,

[1] A Markov Blanket for a node a is a set of nodes X, such that $\{a, x\} \in E, x \in X$. E is a set of edges.

$\ldots s_N$ given s_{n-1}, s_n, s_{n+1}. Therefore,

$$P(c_n|s_1, \ldots, s_N) = p(c_n|s_{n-1}, s_n, s_{n+1})$$

Property 2

Notice that in our conditional independence graph, all paths between nodes s_{n-1} and s_{n+1} must go through one of the nodes in s_n and c_n. This means that s_{n-1} is conditionally independent of s_{n+1} given s_n and c_n. Hence $p(s_{n-1}, s_{n+1}|s_n, c_n) = p(s_{n-1}|s_n, c_n)p(s_{n+1}|s_n, c_n)$. Therefore,

$$p(c_n|s_{n-1}, s_n, s_{n+1}) = \frac{p(s_{n-1}|s_n, c_n)p(s_{n+1}|s_n, c_n)p(s_n|c_n)p(c_n)}{p(s_{n-1}, s_n, s_{n+1})}, \quad n = 2, \ldots, N-1$$

Property 3

Property 1 and 2 imply that

$$p(c_n|s_1, \ldots, s_N) = \frac{p(s_{n-1}|s_n, c_n)p(s_{n+1}|s_n, c_n)p(s_n|c_n)p(c_n)}{p(s_{n-1}, s_n, s_{n+1})}, \quad n = 2, \ldots, N-1$$

Property 3*a*

From the property 3,

$$p(c_1, \ldots, c_N|s_1, \ldots, s_N) = \prod_{n=1}^{N} p(c_n|s_1, \ldots, s_N)$$

$$= \prod_{n=1}^{N} \frac{p(s_{n-1}|s_n, c_n)p(s_{n+1}|s_n, c_n)p(s_n|c_n)p(c_n)}{p(s_{n-1}, s_n, s_{n+1})}$$

where:

$$n = 1: \ p(s_{n-1}|s_n, c_n) = p(s_{n-1}, s_n, s_{n+1}) = 1$$

$$n = N: \ p(s_{n+1}|s_n, c_n) = p(s_{n-1}, s_n, s_{n+1}) = 1$$

Property 4

A node s_i together with its left neighbor s_{i-1} and the node c_i forms a clique. A node s_i together with its right neighbor s_{i+1} and the node c_i forms of a clique. Moreover, nodes s_i and c_i form a separator

and nodes s_i and s_{i+1} form a separator. See appendix A for descriptions.

Property 5

For a sequence of N symbols, our model has a set of $2N-2$ cliques and a set of $2N-3$ separators.

Property 6

Our model has an unique tree $G_1 = (V_1, E_1)$, such that each $v \in V_1$ is a clique. Each $e \in E_1$ is a separator. This tree is called the junction tree and is illustrated in Figure 3.3, where nodes $a = \{s_{N-2}, s_{N-1}, c_{N-2}\}$, $b = \{s_{N-2}, s_{N-1}, c_{N-1}\}$, $c = \{s_{N-1}, s_N, c_{N-1}\}$, $d = \{s_{N-1}, s_N, c_N\}$, $e = \{s_1, s_2, c_1\}$, $f = \{s_1, s_2, c_2\}$, $g = \{s_2, s_3, c_2\}$, $h = \{s_2, s_3, c_3\}$.

Property 7

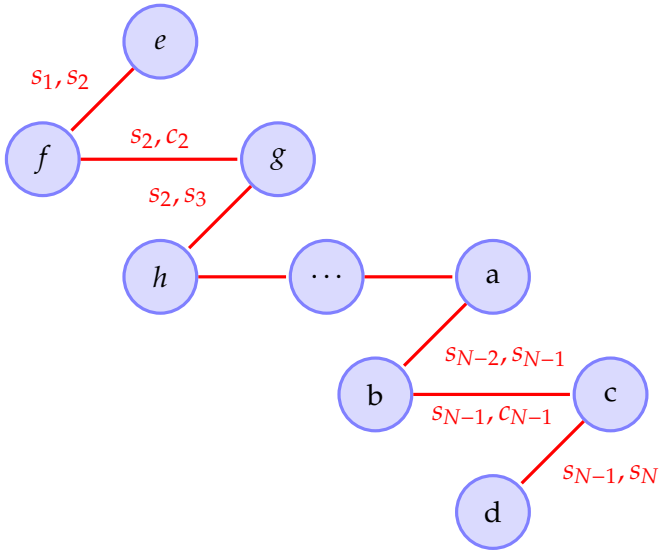


Figure 3.3: The junction tree shows the cliques in running order and the separators between them. The product of the probabilities for the cliques divided by the product of the probabilities for the separators is the joint probability $p(c_1, \ldots, c_N, s_1, \ldots, s_N)$. $a = \{s_{N-2}, s_{N-1}, c_{N-2}\}$, $b = \{s_{N-2}, s_{N-1}, c_{N-1}\}$, $c = \{s_{N-1}, s_N, c_{N-1}\}$, $d = \{s_{N-1}, s_N, c_N\}$, $e = \{s_1, s_2, c_1\}$, $f = \{s_1, s_2, c_2\}$, $g = \{s_2, s_3, c_2\}$, $h = \{s_2, s_3, c_3\}$.

Since the conditional independence graph is triangulated, the product of the probabilities for the cliques divided by the product of the probabilities for the separators is the joint probability $p(c_1, \ldots, c_N, s_1, \ldots, s_N)$

$$p(c_1, \ldots, c_N, s_1, \ldots, s_N) = \frac{\prod_{n=1}^{N} p(s_{n-1}|s_n, c_n) p(s_{n+1}|s_n, c_n) p(s_n|c_n) p(c_n)}{\prod_{m=1}^{N} p(s_m, s_{m+1})}$$
$$\textit{where:} \quad m = N : \; p(s_m, s_{m+1}) = 1$$

Property 8

By property 3*a* and property 7, the probability of $s_1, \ldots, s_N$ can be obtained by:

$$\begin{aligned} p(s_1, \ldots, s_N) \quad &= \quad \frac{p(c_1, \ldots, c_N, s_1, \ldots, s_N)}{p(c_1, \ldots, c_N | s_1, \ldots, s_N)} = \prod_{n=1}^{N} \frac{p(s_n, s_{n+1})}{p(s_{n-1}, s_n, s_{n+1})} \\ \textit{where:} \quad & n = 1 : \; p(s_{n-1}, s_n, s_{n+1}) = 1 \\ & n = N : \; p(s_n, s_{n+1}) = p(s_{n-1}, s_n, s_{n+1}) = 1 \end{aligned}$$

Property 9

By property 7 and property 8, to find a class sequence $< c_1^*, \ldots, c_N^* >$ given a sequence of symbols $< s_1, \ldots, s_N >$, we only need to find c_n^* for s_n individually. Note, the denominator of the equation $p(c_1, \ldots, c_N |\, s_1, \ldots, s_N)$ in the property 3*a* is a constant. Therefore, it does not effect a decision for assigning c_i to s_i, s.t.

$$< c_1^*, c_2^*, \ldots, c_N^* > \quad = \quad \prod_{n=1}^{N} \underset{c_n \in C}{argmax}\, p(s_{n-1}|s_n, c_n) p(s_{n+1}|s_n, c_n) p(s_n|c_n) p(c_n)$$

3.4.7 Comparisons

Different Conditional Independence Graphs

Commonly used graphical models are HMMs, MEMMs, and CRFs. Figure 3.4 shows an *HMM* [1], a *MEMM* [2], a *CRF* [3] [17], and *DSDM*. While the *HMM* and the *MEMM* are directed graphical models, the *CRF* and *DSDM* are undirected graphical models. While others have a link from c_{i-1} to c_i, *DSDM* links c_i and s_{i+1}, s_i, and s_{i-1}. We believe that c_i can be better predicated from s_{i-1}, s_i, and s_{i+1} rather than c_{i-1} when symbols contain several types of information. For example, in the case of NP chunking, POS tag information carried on a symbol is much more useful than the class information assigned on the previous symbol.

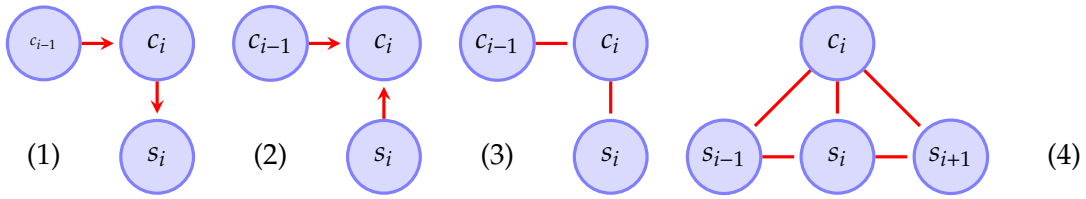


Figure 3.4: (1): an HMM model, (2): a MEMM model, (3): a CRF model, and (4): the data sequence dependence model *DSDM*

Different Assumptions

In a conditional independence graph, s_i and c_i, $i = 1, ..., N$, are nodes. We have $2N$ nodes in total. If no assumption is made, there is a link between every pair of nodes. The degree of each node should be $2N - 1$. Some assumptions are made for the graphical models shown in Figure 3.4. Compared with these graphic models, we find that for each c_i, the degree of our model is three while others are two, which indicates that our model has less independence assumptions. The *HMM* model is built under two conditional independence assumptions. First, given its previous class, the current class is independent of other classes. Moreover, given its current class, the symbol is independent of other classes and symbols. The *MEMM* model is built under one conditional independence assumption. Given its previous class and the current symbol, the current class is independent of other classes and symbols. The *CRF* model is built under the same two conditional assumptions as the *HMM* model. *DSDM* makes one conditional independence assumption. Given the current, the preceding, and the succeeding symbol, the current class is independent of other classes and symbols.

Different Mathematical Expressions

The mathematical expression for the joint probability of the class sequence and the symbol sequence can be directly built from its conditional independence graph. By Chapter 2, for a model having a strongly decomposable graph, the joint probability of the class sequence and the symbol sequence can be derived as a product of the probabilities of cliques divided by a product of probabilities of separators. Then the conditional probability can be obtained by applying Bayes' theorem. In Figure 3.4, the *HMM* and the *MEMM* are directed models while the *CRF* and *DSDM* are undirected

models. The mathematical expressions of these models are stated as following.

- *HMM*:

$$p(c_1, ...c_N, s_1, ..., s_N) = \prod_{i=1}^{N} p(s_i|c_i)p(c_i|c_{i-1})$$

- *MEMM*:

$$p(c_1, ..., c_N|s_1, ..., s_N) = \frac{\prod_{i=1}^{N} p(c_i|c_{i-1}s_i)}{\sum_{c_1,...,c_N \in C} \prod_{n=1}^{N} p(c_n|c_{n-1}s_n)}$$

- *CRF*:

$$p(c_1, ..., c_N|s_1, ..., s_N) = \frac{\prod_{i=1}^{N} p(s_i|c_i)p(c_i|c_{i-1})}{\sum_{c_1,...,c_N \in C} \prod_{n=1}^{N} p(s_n|c_n)p(c_n|c_{n-1})}$$

- *DSDM*:

$$p(c_1, ..., c_N|s_1, ..., s_N) = \frac{\prod_{i=1}^{N} p(s_{i-1}|s_i, c_i)p(s_{i+1}|s_i, c_i)p(s_i|c_i)p(c_i)}{\sum_{c_1,...,c_N \in C} \prod_{n=1}^{N} p(s_{n-1}|s_n, c_n)p(s_{n+1}|s_n, c_n)p(s_n|c_n)p(c_n)}$$

Comparing with the numerators of $p(c_1, ...c_N|s_1, ..., s_N)$, the *MEMM* has one term, the *CRF* has two terms, and *DSDM* has four terms. Moreover, different from the *MEMM* and the *CRF*, none of the terms in the *DSDM* is associated with the a neighboring pair of classes.

Different Time and Memory Complexities

From Equation (3.10) and Equation (3.11), for *DSDM*, we have a time complexity of $O(NM)$ and a memory complexity of $O(N^2M)$. On the other hand, other commonly used probabilistic graphical models, such as *HMM*s, *MEMM*s, and *CRF*s, employ a dynamic programming algorithm to obtain a sequence of optimal classes for a sequence of symbols (will discuss in Chapter 4). By dynamic programming, an optimal class for the current symbol is obtained based on an optimal class of the previous symbol. Therefore, the optimal class for the last symbol is determined after the last symbol has been reached. The optimal class sequence needs to be determined by tracing back from the last optimal class to the first optimal class. For each symbol, information for M classes needs to be stored. Hence, for a sequence of N symbols, we need to have $T_c = O(M^2N)$ time complexity and $M_c = O(MN)$ memory complexity.

The ratio of time complexity of *DSDM* to other commonly used probabilistic graphical models is

$$\frac{NM}{M^2N} = \frac{1}{M}$$

The ratio of memory complexity of *DSDM* to other commonly used probabilistic graphical models is

$$\frac{MN^2}{MN} = N$$

Therefore, if we need to recognize a sequence of N symbols with M categories, *DSDM* only takes $\frac{1}{M}$ time and N memory space of commonly used probabilistic graphical models.

3.5 Two Other Probabilistic Graphical Models

Let $s =< s_1, \ldots, s_N >$ be a sequence of N symbols. Let C be a set of M classes, $C = \{C_1, \ldots, C_M\}$. Let $c =< c_1, ..., c_N >$ be a sequence of classes. Let e be the economic gain function $e : C^N \times C^N \rightarrow \mathcal{R}$.

3.5.1 Context Independence Model (CIM)

The context independence model (CIM) is built under the following conditional independence assumptions:

- c_i is independent of s_j, $i \neq j$, given s_i
- c_i is independent of c_j

The conditional independence graph of the context independence model is shown in Figure 3.5

Computing $p(c_1, \ldots, c_N, s_1, \ldots, s_N)$

The joint probability distribution $p(c_1, \ldots, c_N, s_1, \ldots, s_N)$ can be directly obtained from G. It is represented by Equation (3.12).

$$p(c_1, \ldots, c_N, s_1, \ldots, s_N)$$

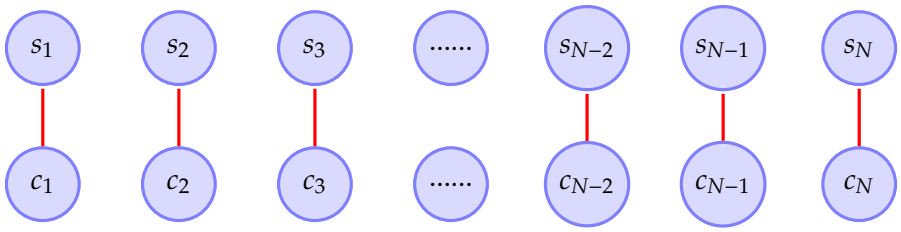


Figure 3.5: The conditional independence graph defining the context independence model.

$$
\begin{aligned}
&= \frac{p(s_1, c_1)p(s_2, c_2)\ldots p(s_N, c_N)}{1 \cdot 1 \cdot \ldots \cdot 1} \\
&= \prod_{n=1}^{N} p(s_n, c_n) = \prod_{n=1}^{N} p(s_n|c_n)p(c_n)
\end{aligned} \tag{3.12}
$$

Computing $max(E[e](c_1, \ldots, c_N))$

$$
p(c_n|s_1, \ldots, s_N) = p(c_n|s_n) = \frac{p(c_n, s_n)}{p(s_n)}
$$

Therefore

$$
\begin{aligned}
\max_{c_1,\ldots,c_N} (E[e](c_1, \ldots, c_N) &= \max_{c_1,\ldots,c_N} \sum_{n=1}^{N} p(c_n|s_1, \ldots, s_N) \\
&= \sum_{n=1}^{N} \max_{c_n \in C} p(c_n|s_1, \ldots, s_N) \\
&= \sum_{n=1}^{N} \max_{c_n \in C} p(c_n|s_n) \\
&= \sum_{n=1}^{N} \frac{1}{p(s_n)} \max_{c_n \in C} p(c_n, s_n) \\
&= \sum_{n=1}^{N} \frac{1}{p(s_n)} \max_{c_n \in C} p(s_n|c_n)p(c_n) \\
&\propto \sum_{n=1}^{N} \max_{c_n \in C} p(s_n|c_n)p(c_n)
\end{aligned} \tag{3.13}
$$

$$
\operatorname*{argmax}_{c_1,\ldots,c_N} p(c_1, \ldots, c_N, s_1, \ldots, s_N) = \prod_{n=1}^{N} \operatorname*{argmax}_{c_n} p(s_n|c_n)p(c_n)
$$

3.5.2 Class Sequence Dependence Model (CSDM)

The class sequence dependence model is built under the following conditional independence assumptions:

- s_i is independent of c_j, $j \neq i$, given c_i.
- c_i is independent of c_j given c_{i-1}.

The conditional independence graph of the class sequence dependence model is shown in Figure 3.6

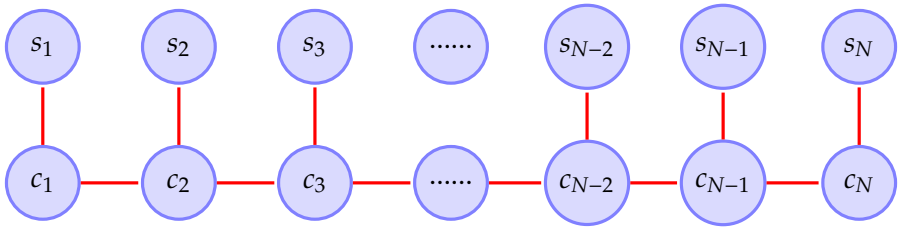


Figure 3.6: The conditional independence graph defining the class sequence dependence model.

Computing $p(c_1, \ldots, c_N, s_1, \ldots, s_N)$

The joint distribution of $p(c_1, ..., c_N, s_1, ..., s_N)$ from Figure 3.6 is:

$$\begin{aligned} & p(c_1, ..., c_N, s_1, ..., s_N) \\ = & \frac{p(s_1, c_1)p(c_1, c_2)p(s_2, c_2)p(c_2, c_3)\ldots p(c_{N-1}, c_N)p(s_N, c_N)}{p(c_1)p(c_2)p(c_2)\ldots p(c_{N-1}p(c_{N-1})p(c_N)} \\ = & \prod_{n=1}^{N} p(s_n|c_n)p(c_n|c_{n+1}) \qquad\qquad \textit{where} \qquad p(c_N|c_{N+1}) = p(c_N) \end{aligned} \tag{3.14}$$

Computing $max(E[e](c_1, \ldots, c_N))$

From Figure 3.6, we have

$$p(c_1, \ldots, c_N) = \frac{p(c_1, c_2)p(c_2, c_3)\ldots p(c_{N-1}, c_N)}{p(c_2)p(c_3)\ldots p(c_{N-1})}$$

$$
\begin{aligned}
&= p(c_1)\frac{p(c_1,c_2)p(c_2,c_3)\ldots p(c_{N-1},c_N)}{p(c_1)p(c_2)p(c_3)\ldots p(c_{N-1})p(c_N)}p(c_N) \\
&= p(c_1)\cdot\prod_{i=1}^{n-1}p(c_{i+1}|c_i)\cdot\frac{1}{p(c_n)}\cdot\prod_{j=n+1}^{N}p(c_{j-1}|c_j)\cdot p(c_N)
\end{aligned}
$$

$$
\begin{aligned}
& p(c_n|s_1,\ldots,s_N) = \frac{p(c_n,s_1,\ldots,s_N)}{p(s_1,\ldots,s_N)} \\
= \;& \frac{1}{p(s_1,\ldots,s_N)}\sum_{c_1,\ldots,c_{n-1},c_{n+1},\ldots,c_N} p(c_1,\ldots,c_N,s_1,\ldots,s_N) \\
= \;& \frac{1}{p(s_1,\ldots,s_N)}\sum_{c_1,\ldots,c_{n-1},c_{n+1},\ldots,c_N} p(s_1,\ldots,s_N|c_1,\ldots,c_N)p(c_1,\ldots,c_N) \\
= \;& \frac{1}{p(s_1,\ldots,s_N)}\sum_{c_1,\ldots,c_{n-1},c_{n+1},\ldots,c_N}\prod_{n=1}^{N}p(s_n|c_n)p(c_1,\ldots,c_N) \\
= \;& \frac{1}{p(s_1,\ldots,s_N)}\sum_{\substack{c_1,\ldots,c_{n-1}\\ c_{n+1},\ldots,c_N}}\prod_{n=1}^{N}p(s_n|c_n)\cdot p(c_1)\cdot\prod_{i=1}^{n-1}p(c_{i+1}|c_i)\cdot\frac{1}{p(c_n)}\cdot\prod_{j=n+1}^{N}p(c_{j-1}|c_j)\cdot p(c_N) \\
= \;& \frac{1}{p(s_1,\ldots,s_N)}\left[\sum_{c_1,\ldots,c_{n-1}} p(c_1)\prod_{i=1}^{n-1}p(s_i|c_i)\cdot p(c_{i+1}|c_i)\right] \\
& \cdot\frac{p(s_n|c_n)}{p(c_n)}\cdot \\
& \left[\sum_{c_{n+1},\ldots,c_N} p(c_N)\prod_{j=n+1}^{N}p(s_j|c_j)p(c_{j-1}|c_j)\right] \\
= \;& \frac{1}{p(s_1,\ldots,s_N)}\left[\sum_{c_1}p(s_1|c_1)p(c_2|c_1)\cdot p(c_1)\sum_{c_2}p(s_2|c_2)p(c_3|c_2)\ldots\sum_{c_{n-1}}p(s_{n-1}|c_{n-1})p(c_n|c_{n-1})\right] \\
& \cdot\frac{p(s_n|c_n)}{p(c_n)}\cdot \\
& \left[\sum_{c_{n+1}}p(s_{n+1}|c_{n+1})p(c_{n+2}|c_{n+1})\ldots\sum_{c_N}p(s_N|c_{N-1})p(c_N|c_{N-1})\cdot p(c_N)\right] \\
= \;& \frac{1}{p(s_1,\ldots,s_N)}f(c_n)\cdot\frac{p(s_n|c_n)}{p(c_n)}\cdot g(c_n) \qquad (3.15)
\end{aligned}
$$

Note: where f and g are functions, s.t. $f, g : R \rightarrow R$.

$$f(c_n) = p(c_1) \sum_{c_1} p(s_1|c_1)p(c_2|c_1) \sum_{c_2} p(s_2|c_2)p(c_3|c_2) \ldots \sum_{c_{n-1}} p(s_{n-1}|c_{n-1})p(c_n|c_{n-1})$$

and

$$\begin{aligned} g(c_n) &= p(c_N) \sum_{c_{n+1}} p(s_{n+1}|c_{n+1})p(c_{n+2}|c_{n+1}) \sum_{c_{n+2}} p(s_{n+2}|c_{n+2})p(c_{n+3}|c_{n+2}) \ldots \\ &\quad \sum_{c_N} p(s_N|c_{N-1})p(c_N|c_{N-1}) \end{aligned}$$

Therefore,

$$\begin{aligned} \max_{c_1,\ldots,c_N} (E[e](c_1, \ldots, c_N)) &= \max_{c_1,\ldots,c_N} \sum_{n=1}^{N} p(c_n|s_1, \ldots, s_N) \\ &= \sum_{n=1}^{N} \max_{c_n \in C} p(c_n|s_1, \ldots, s_N) \\ &= \sum_{n=1}^{N} \frac{1}{p(s_1, \ldots, s_N)} \max_{c_n \in C} f(c_n) \cdot \frac{p(s_n|c_n)}{p(c_n)} \cdot g(c_n) \\ &\propto \sum_{n=1}^{N} \max_{c_n \in C} (f(c_n) \cdot \frac{p(s_n|c_n)}{p(c_n)} \cdot g(c_n)) \end{aligned} \tag{3.16}$$

$$\operatorname*{argmax}_{c_1,\ldots,c_N} p(c_1, \ldots, c_N, s_1, \ldots, s_N) = \prod_{n=1}^{N} \operatorname*{argmax}_{c_n} (f(c_n) \cdot \frac{p(s_n|c_n)}{p(c_n)} \cdot g(c_n))$$

3.5.3 Complexities

Time Complexity

For the context independence model, from Equation (3.13), let

$$f(s_n, c_n) = p(s_n, c_n) = p(s_n|c_n)p(c_n).$$

For each symbol s_n, we need to assign a c_n, such that,

$$f(s_n, c_n) \geq f(s_n, c'_n), c'_n \in C.$$

To compute $f(s_n, c_n)$, there are two multiplications. To obtain the maximum probability value in the case of M classes, we require $M-1$ comparisons. In the case of a sequence of N symbols, we need

$$T_c = 2 * N * (M-1) = O(N * M)$$

For the class sequence dependence model, from Equation (3.16), let

$$h(s_n, c_n) \quad = \quad f(c_n) \cdot \frac{p(s_n|c_n)}{p(c_n)} \cdot g(c_n).$$

For each symbol s_n, we need to assign a c_n, such that,

$$h(s_n, c_n) \geq h(s_n, c'_n), \quad c'_n \in C.$$

To compute $h(s_n, c_n)$, we need to have $2(n-1) + 2(N-n+1) + 3 = 2N + 3$ multiplications. To obtain the maximum probability value, we require $M-1$ comparisons. In the case of a sequence of N symbols, we need

$$T_c = (2N+3) * N * (M-1) = O(N^2 M)$$

Memory Complexity

For the context independence model and class sequene dependence model, because the global maximum probability is determined by each local maximal probability, for a path of N symbols, we need to store the information of the current node. That is, we need only store M probability values in order to find the maximal probability value. Moreover, for obtaining a probability table,

we need to have $O(NM)$ memory space. Therefore,

$$M_c = M = O(M) + O(NM) = O(NM)$$

Chapter 4

Finding an Class Sequence

In this chapter, we describe commonly used methods for finding an optimal sequence for probabilistic graphical models. We discuss two types of the hidden Markov models built based on the expected economic gain of the second type. We discuss two dynamic programming algorithms. We give examples to show the procedures of these algorithms.

4.1 Finding an Optimal Assigned Class Categories

In this section, we discuss an algorithm for finding a sequence of optimal assigned class categories $c_1, \ldots, c_N$, s.t. $c_n \in C = \{C_1, \ldots, C_M\}$ for a sequence of symbols $s_1, \ldots, s_N$ based on maximizing $p(c_n|s_1, \ldots, s_N)$, $n = 1, 2, \ldots, N-1, N$. This algorithm is called the Baum-Welch algorithm [18]. From Equation (3.16), we have

$$\max_{c_1,\ldots,c_N} (E[e](c_1, \ldots, c_N)) \propto \quad \sum_{k=1}^{N} max_{c_k \in C} f(c_k) \cdot \frac{p(s_k|c_k)}{p(c_k)} \cdot g(c_k)$$

where

- $f(c_1) = p(c_1)$
- $f(c_k) = \sum_{c_{k-1}} p(s_{k-1}|c_{k-1}) p(c_k|c_{k-1}) f(c_{k-1}), \qquad k = 2, \ldots, N$
  - $f(c_2) = \sum_{c_1} p(s_1|c_1) p(c_2|c_1) f(c_1)$

- $g(c_N) = p(c_N)$
- $g(c_k) = \sum_{c_{k+1}} p(s_{k+1}|c_{k+1})p(c_k|c_{k+1})g(c_{k+1}), \qquad k = N-1, \ldots, 1$
  - $g(c_{N-1}) = \sum_{c_N} p(s_N|c_N)p(c_{N-1}|c_N)g(c_N)$

4.1.1 The Forward-Backward Algorithm (the Baum-Welch Algorithm)

Let $S =< s_1, s_2, ...s_N >$, $C = \{C_1, \ldots, C_M\}$

- Let $\alpha(i, C_m)$ be the probability that we assign C_m to i^{th} symbol s_i starting from the first symbol s_1.
  - $\alpha(1, C_m) = p(C_m) * p(s_1|C_m)$
  - $\alpha(i+1, C_m) = \sum_{C_n \in C} \alpha(i, C_n)p(C_m|C_n)p(s_{i+1}|C_m)$
- Let $\beta(i, C_m)$ be the probability that we assign C_m to i^{th} symbol s_i starting from the last symbol s_N.
  - $\beta(N, C_m) = 1$
  - $\beta(i, C_m) = \sum_{C_n \in C} \beta(i+1, C_n)p(C_n|C_m)p(s_{i+1}|C_n)$
- Let $\gamma(i, C_m)$ be the probability that we assign C_m to s_i
  - $\gamma(i, C_m) = \frac{\alpha(i,C_m)}{p(S)} \sum_{C_n \in C} p(C_n|C_m)p(s_{i+1}|C_n)\beta(i+1, C_n)$

The computation of $\alpha(i, C_m)$ is presented in Figure 4.1. For example, $\alpha(2, C_1)$ is computed as:

$$\begin{aligned}
\alpha(2, C_1) &= \alpha(1, C_1)p(C_1|C_1)p(s_2|C_1) + \ldots + \alpha(1, C_m)p(C_1|C_m)p(s_2|C_1) + \ldots + \\
&\quad \alpha(1, C_M)p(C_1|C_M)p(s_2|C_M) \\
&= p(C_1)p(s_1|C_1)p(C_1|C_1)p(s_2|C_1) + \ldots + p(C_m)p(s_1|C_m)p(C_1|C_m)p(s_2|C_1) + \ldots + \\
&\quad p(C_M)p(s_1|C_M)p(C_1|C_M)p(s_2|C_1)
\end{aligned}$$

The computation of $\beta(i, C_m)$ is presented in Figure 4.2. For example, $\beta(N-1, C_1)$ is computed as:

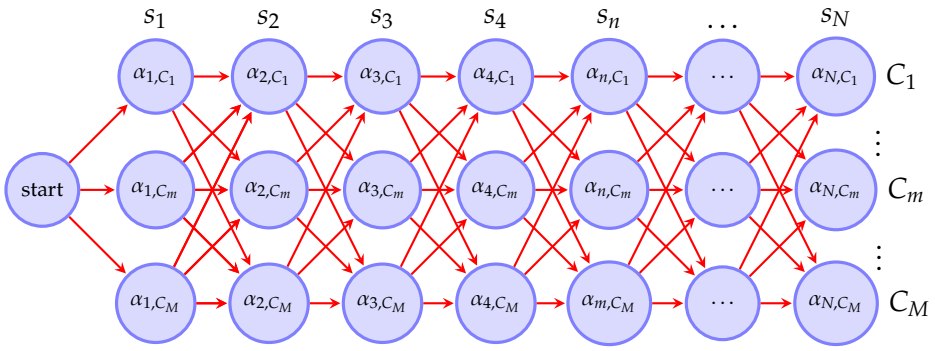


Figure 4.1: Computing α_{i,C_m}.

$$\begin{aligned}
\beta(N-1,C_1) &= \beta(N,C_1)p(C_1|C_1)p(s_N|C_1) + \ldots + \beta(N,C_m)p(C_m|C_1)p(s_N|C_m) + \ldots + \\
&\quad \beta(N,C_M)p(C_M|C_1)p(s_N|C_M) \\
&= 1 \cdot p(C_1|C_1)p(s_N|C_1) + \ldots + 1 \cdot p(C_m|C_1)p(s_N|C_m) + \ldots + \\
&\quad 1 \cdot p(C_M|C_1)p(s_N|C_M) \\
&= p(C_1|C_1)p(s_N|C_1) + \ldots + \\
&\quad p(C_m|C_1)p(s_N|C_m) + \ldots + p(C_M|C_1)p(s_N|C_M)
\end{aligned}$$

The computation of $\gamma(i,C_m)$ is presented in Figure 4.3. For example, $\gamma(2,C_1)$ can be computed as:

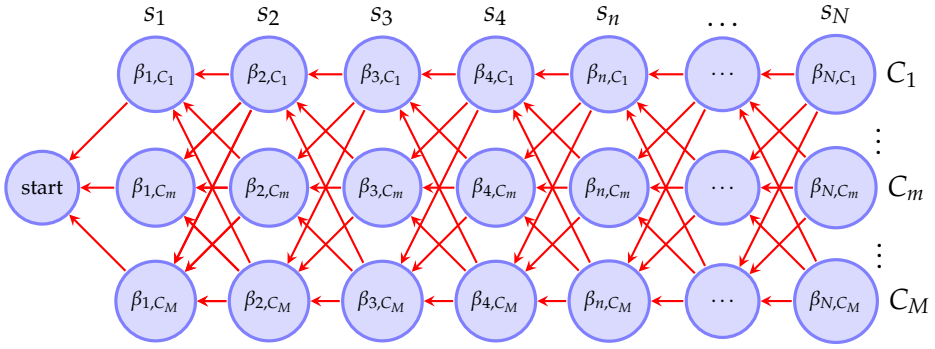


Figure 4.2: Computing β_{i,C_m}.

$$\begin{aligned}
\gamma(2,C_1) &= \frac{\alpha(2,C_1)}{\alpha(N,C_1) + \ldots + \alpha(N,C_m) + \ldots + \alpha(N,C_M)} \cdot \\
&\quad (p(C_1|C_1)p(s_3|C_1)\beta(3,C_1) + \ldots + p(C_m|C_1)p(s_3|C_m)\beta(3,C_m) + \ldots +
\end{aligned}$$

$$p(C_M|C_1)p(s_3|C_M)\beta(3, C_M))$$

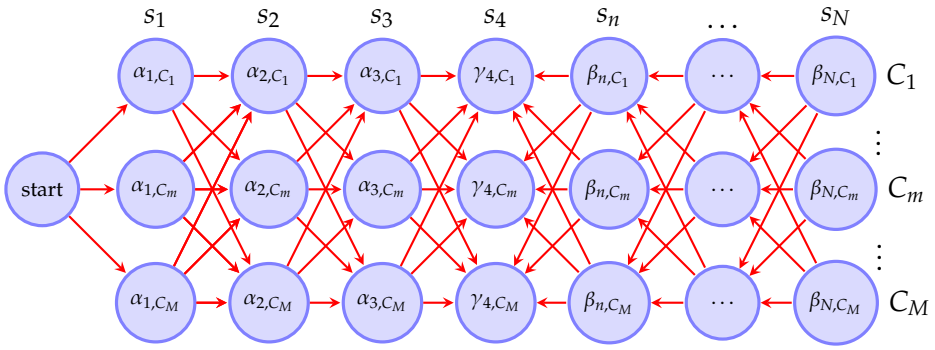


Figure 4.3: Computing γ_{i,C_m}.

4.1.2 An Example

Let C be a set of categories, s.t. $C = \{C_1, C_2, C_3\}$. Our data only has five letters: $\{A, B, C, D, E\}$. The probabilities of each letter occurring in categories are shown in Table 4.1. For example, among all letters under C_1, A occurs with a probability of 30%, B with 20%, C with 40%, D with 10%, and E with 0%. Moreover, the transition probabilities from C_i to C_j are shown in Table 4.2. We want to find the most probable path for the input sequence $ACDBBCE$ where the probability of starting to each category is 0.33. Applying the Baum-Welch algorithm, we found that the largest probability of $p(ACDBBCE, c_A c_C c_D c_B c_B c_C c_E)$. The details are shown in Table 4.3, Table 4.4, and Table 4.5. The most probable path is determined by tracing back from the node in the last column and row in Figure 4.8. It is $C_2C_1C_3C_3C_3C_1C_3$. The optimal path is shown in Figure 4.9.

4.2 Expected Economic Gain of the Second Type

From Section 3.2 in Chapter 3, we have two types of economic gain functions. The expected economic gain of a sequence of assigned class categories $c_1, \ldots, c_N$ given a sequence of input

Table 4.1: Probability of an observed letter in a category

$P(Letter\|$ $category)$	C_1	C_2	C_3
$P(A\|C_i)$	0.3	0.4	0.2
$P(B\|C_i)$	0.2	0.1	0.25
$P(C\|C_i)$	0.4	0.1	0.15
$O(D\|C_i)$	0.1	0.3	0.2
$P(E\|C_i)$	0.0	0.1	0.2

Table 4.2: Transition probabilities from C_i to C_j

	C_1	C_2	C_3
C_1	0.3	0.3	0.4
C_2	0.3	0.4	0.3
C_3	0.25	0.25	0.5

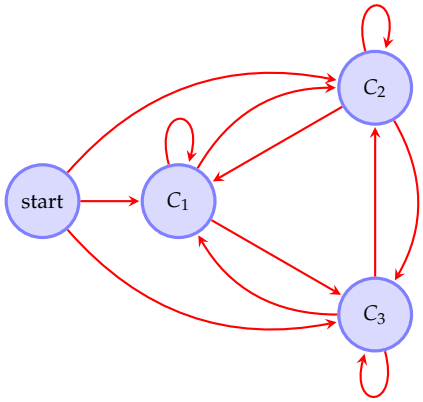


Figure 4.4: The state diagram of the example

Table 4.3: Computation of α

α_{i,C_m}	A	C	D	B	B	E	C
C_1	0.099000	0.0343200	0.001747	0.000688	0.000126	0.000048	0
C_2	0.132000	0.009900	0.005339	0.000400	0.000067	0.000013	0.000003
C_3	0.066000	0.0168300	0.005223	0.001218	0.000251	0.000029	0.000008

symbols $s_1, \ldots, s_N$ and a sequence of true class categories $c_1^T, \ldots, c_N^T$ is:

$$E[e](c_1, \ldots, c_N) = \sum_{c_1^T, \ldots, c_N^T} e(c_1^T, \ldots, c_N^T, c_1, \ldots, c_N) p(c_1^T, \ldots, c_N^T | s_1, \ldots, s_N)$$

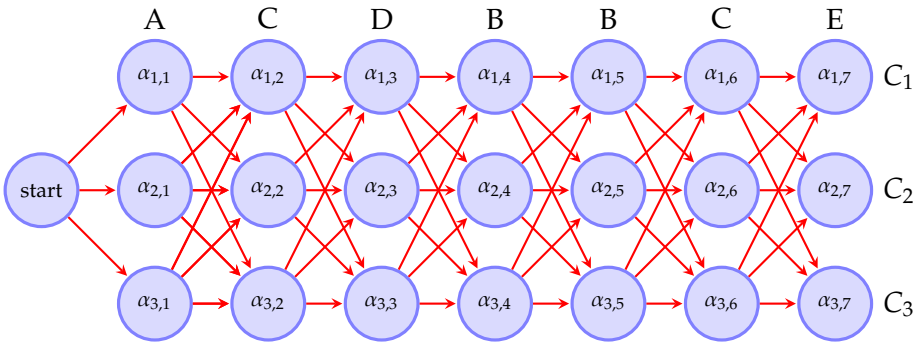


Figure 4.5: Computing $\alpha_{i,j}$, where $1 \leq i \leq 3, 1 \leq j \leq 7$.

Table 4.4: Computation of β

β_{i,C_m}	A	C	D	B	B	E	C
C_1	0.000026	0.000125	0.000606	0.003151	0.017175	0.080000	1.000000
C_2	0.000028	0.000144	0.000640	0.003332	0.018075	0.090000	1.000000
C_3	0.000038	0.0000180	0.000986	0.005073	0.025250	0.130000	1.000000

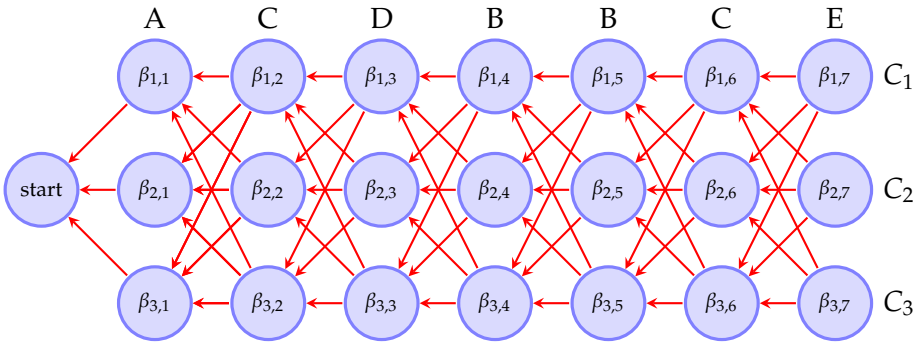


Figure 4.6: Computing $\beta_{i,j}$, where $1 \leq i \leq 3, 1 \leq j \leq 7$.

Table 4.5: Computation of γ

γ_{i,C_m}	A	C	D	B	B	E	C
C_1	0.290833	0.516867	0.135554	0.274798	0.247080	0.517576	0
C_2	0.371525	0.148603	0.379245	0.141976	0.124512	0.124167	0.262475
C_3	0.190573	0.265156	0.428121	0.530064	0.489175	0.358257	0.737525

In the case of $c_1^T, \ldots, c_N^T = c_1, \ldots, c_N$ and $e(c_1, \ldots, c_N, c_1, \ldots, c_N) = 1$, then:

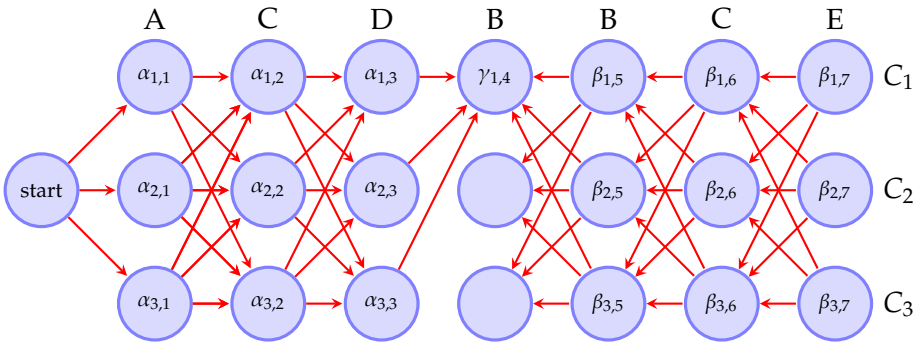


Figure 4.7: Computing $\gamma_{i,j}$, where $1 \leq i \leq 3$, $1 \leq j \leq 7$.

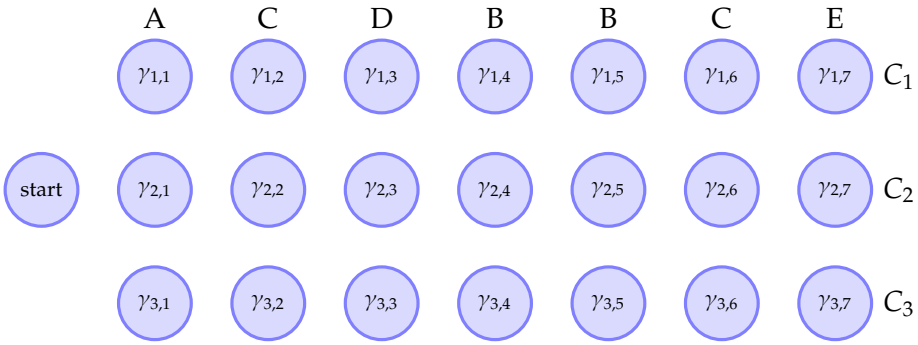


Figure 4.8: Computing $\gamma_{i,j}$, where $1 \leq i \leq 3$, $1 \leq j \leq 7$.

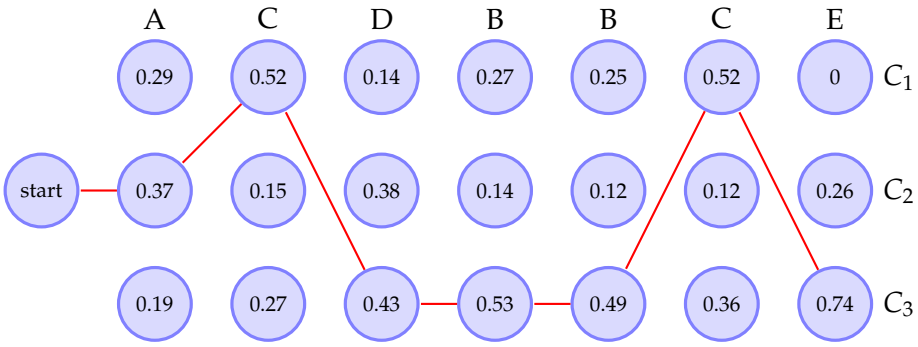


Figure 4.9: The optimal path for the input sequence *ACDBBCE* obtained by the Baum-Welch algorithm

$$E[e](c_1, \ldots, c_N) = p(c_1, \ldots, c_N | s_1, \ldots, s_N) \tag{4.1}$$

By equation (4.1), the expected economic gain for a sequence of class assignments is the probability of the class assignment sequence for a given symbol sequence. Maximizing the expected gain is associated with maximizing $p(c_1, \ldots, c_N | s_1, \ldots, s_N)$.

$$\begin{aligned} \max_{c_1,\ldots,c_N} (E[e](c_1, \ldots, c_N)) &= \max_{c_1,\ldots,c_N} (p(c_1, \ldots, c_N | s_1, \ldots, s_N)) \\ &= \frac{1}{p(s_1, \ldots, s_N)} \max_{c_1,\ldots,c_N} (p(c_1, \ldots, c_N, s_1, \ldots, s_N)) \\ &\propto \max_{c_1,\ldots,c_N} p(c_1, \ldots, c_N, s_1, \ldots, s_N) \end{aligned}$$

Therefore, to find the assigned class sequence $c_1, ..., c_N$ for the input sequence $s_1, .., s_N$ that maximizes the expected gain, we need to find a sequence of assigned classes $c_1, \ldots, c_n \in C$ for $s_1, \ldots, s_N$ that maximizes the joint probability function $p(c_1, \ldots, c_n, s_1, ..., s_N)$. This leads to the following two different probability graphical models.

4.2.1 Hidden Markov Model I

Let $G_1 = (V, E_1)$ be a directed graph, where $V = \{c_1, \ldots, c_N, s_1, \ldots, s_N\}$ and $E_1 \subseteq V \times V$, s.t. $E_1 = \{(c_1, s_1), (c_1, c_2), \ldots, (c_{N-1}, c_N), (c_N, s_N)\}$. G is represented in Figure 4.10.

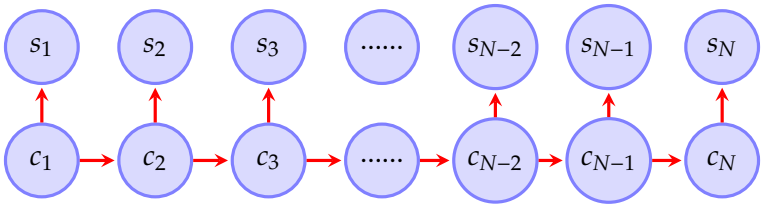


Figure 4.10: Directed graph G associates with a hidden Markov model I.

Let $G = (V, E)$ be an moralized graph of G_1 in Figure 4.10. It is an undirected graph, where $V = \{c_1, \ldots, c_N, s_1, \ldots, s_N\}$ and $E \subseteq V \times V$, s.t. $E = \{ \{c_1, s_1\}, \{c_1, c_2\}, \ldots, \{c_{N-1}, c_N\}, \{c_N, s_N\} \}$. G is represented in Figure 4.11. By Chapter 2, G_1 is a moral graph, that is, no edge needs to be added to its moralized graph G. In this case, the joint probabilities associated with G_1 in Figure 4.10 and G in Figure 4.11 are the same.

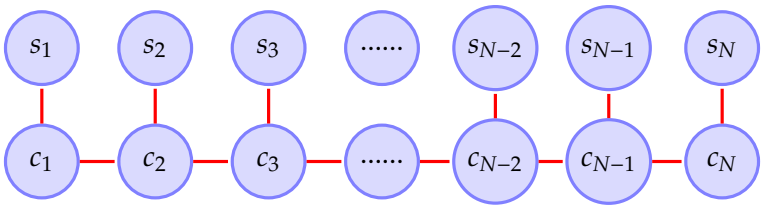


Figure 4.11: Undirected graph G associates with a hidden Markov model I.

Computing $p(c_1, \ldots, c_N, s_1, \ldots, s_N)$

$$\begin{aligned} & p(c_1, \ldots, c_N, s_1, \ldots, s_N) \\ = & \frac{p(s_1, c_1)p(c_1, c_2)p(s_2, c_2)p(c_2, c_3)\ldots p(c_{N-1}, c_N)p(s_N, c_N)}{p(c_1)p(c_2)p(c_2)\ldots p(c_{N-1})p(c_{N-1})p(c_N)} \\ = & \prod_{n=1}^{N} p(s_n|c_n)p(c_n|c_{n+1}) \qquad where \;\; p(c_N|c_{N+1}) = p(c_N) \end{aligned} \tag{4.2}$$

Therefore,

$$\begin{aligned} \max_{c_1,\ldots,c_N}(E[e](c_1, \ldots, c_N)]) &= \max_{c_1,\ldots,c_N}(p(c_1, \ldots, c_N|s_1, \ldots, s_N)) \\ &= \frac{1}{p(s_1, \ldots, s_N)} \max_{c_1,\ldots,c_N} p(c_1, \ldots, c_N, s_1, \ldots, s_N) \\ &= \frac{1}{p(s_1, \ldots, s_N)} \max_{c_1,\ldots,c_N} \prod_{n=1}^{N} p(s_n|c_n)p(c_n|c_{n+1}) \\ &= \frac{1}{p(s_1, \ldots, s_N)} \max_{c_1,\ldots,c_N} \sum_{n=1}^{N} log(p(s_n|c_n)p(c_n|c_{n+1})) \\ &\propto \max_{c_1,\ldots,c_N} \sum_{n=1}^{N} log(p(s_n|c_n)p(c_n|c_{n+1})) \end{aligned} \tag{4.3}$$

4.2.2 Hidden Markov Model II

Let $G_1 = (V, E_1)$ be a directed graph, where $V = \{c_1, \ldots, c_N, s_1, \ldots, s_N\}$ and $E_1 \subseteq V \times V$, s.t. $E_1 = \{(s_1, c_1), (c_1, c_2), (s_2, c_2), (c_2, c_3) \ldots, (c_{N-1}, c_N), (s_N, c_N)\}$. G is represented in Figure 4.12.

Let $G = (V, E)$ be a moralized graph of directed graph G_1, $V = \{c_1, \ldots, c_N, s_1, \ldots, s_N\}$ and $E \subseteq V \times V$, s.t. $E = \{\{s_1, c_1\}, \{c_1, c_2\}, \{s_2, c_2\}, \{c_2, c_3\} \ldots, \{c_{N-1}, c_N\}, \{s_N, c_N\}\}$. G is represented in Figure 4.13.

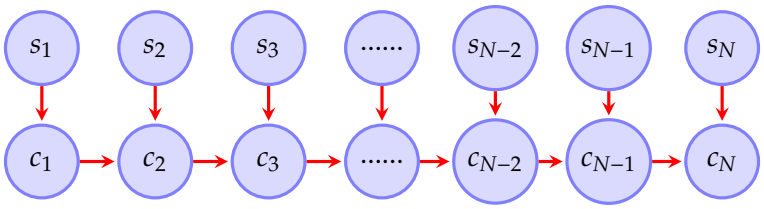


Figure 4.12: Directed graph G_1 associates with a hidden Markov model II.

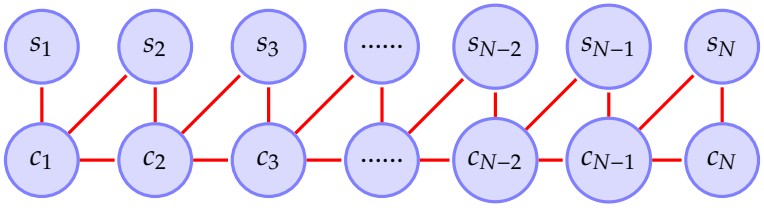


Figure 4.13: Undirected graph G associates with a hidden Markov model II.

Computing $p(c_1, \ldots, c_N, s_1, \ldots, s_N)$

Based on Figure 4.13, we have:

$$\begin{aligned} p(c_1, \ldots, c_N, s_1, \ldots, s_N) &= \frac{p(s_1, c_1)p(s_2, c_1, c_2)p(s_3, c_2, c_3)\ldots p(s_N, c_{N-1}, c_N)}{p(c_1)p(c_2)p(c_3)\ldots p(c_{N-1})} \\ &= \prod_{i=1}^{N} p(s_i, c_i|c_{i-1}) \qquad \textit{where} \;\; p(s_1, c_1|c_0) = p(s_1, c_1) \\ &= \prod_{i=1}^{N} \frac{p(c_i|s_i, c_{i-1})p(s_i, c_{i-1})}{p(c_{i-1})} \end{aligned}$$

Based on Chapter 2, if we introduce the dependence $s_i \perp\!\!\!\perp c_{i-1}$ in G_1, we have

$$\begin{aligned} p(c_1, \ldots, c_N, s_1, \ldots, s_N) &= \frac{p(c_i|s_i, c_{i-1})p(s_i)p(c_{i-1})}{p(c_{i-1})} \\ &= \prod_{i=1}^{N} p(c_i|s_i, c_{i-1})p(s_i) \\ \textit{Define} \quad p(s_i) &= \mathcal{M}_{s_i} \\ p(c_1, \ldots, c_N, s_1, \ldots, s_N) &= \prod_{i=1}^{N} \mathcal{M}_{s_i} p(c_i|s_i, c_{i-1}) \end{aligned} \tag{4.4}$$

Therefore,

$$\begin{aligned}
\max_{c_1,\ldots,c_N}(E[e](c_1,\ldots,c_N)]) &= \max_{c_1,\ldots,c_N}(p(c_1,\ldots,c_N|s_1,\ldots,s_N)) \\
&= \frac{1}{p(s_1,\ldots,s_N)} \max_{c_1,\ldots,c_N}(p(c_1,\ldots,c_N,s_1,\ldots,s_N)) \\
&= \frac{1}{p(s_1,\ldots,s_N)} \max_{c_1,\ldots,c_N} \prod_{i=1}^{N} \mathcal{M}_{s_i} p(c_i|c_{i-1},s_i) \\
&= \frac{1}{p(s_1,\ldots,s_N)} \max_{c_1,\ldots,c_N} \sum_{i=1}^{N} log(\mathcal{M}_{s_i} p(c_i|c_{i-1},s_i)) \\
&\propto \max_{c_1,\ldots,c_N} \sum_{i=1}^{N} log(p(c_i|c_{i-1},s_i)) \qquad (4.5)
\end{aligned}$$

4.2.3 Finding an Optimal Assigned Class Categories

In this section, we discuss an algorithm for finding a sequence of optimal assigned class categories $c_1,\ldots,c_N$ for an input symbol sequence $s = s_1,\ldots,s_N$, s.t. $c_n \in C = \{C_1,\ldots,C_M\}$ for a sequence of symbols $s_1,\ldots,s_N$ based on maximizing $p(c_1, \ldots, c_n, s_1, \ldots, s_N)$. This algorithm is called Viterbi algorithm.

4.2.4 The Viterbi Algorithm

The algorithm is applied for Equation (4.3).

- Let $w(i, j)$ be the weight of a node in the position of (i, j).
- For $m = 1$ to M
- $\quad w(1, m) = log\, p(s_1|C_m) + log\, p_1(C_m)$
- For $n = 1$ to N
- $\quad$ For $m = 2$ to M
- $\quad\quad w(n, m) = log\,(p(s_n|C_m)) + max_{m'=1,\ldots,M}\{w(n-1, m') + log\, p_{n-1,n}(C_m|C_{m'})\}$

The Trellis of Viterbi's Algorithm

$G = (V, E)$ in Figure 4.14 represents the trellis of Viterbi algorithm. A label $L(i, j)$ inside of each node

$v_{i,j} \in V$ represents $log\, p(s_i|C_j)$, where $1 \leq i \leq N$, $1 \leq j \leq M$. Moreover, a label $M(v_{i',j'}, v_{j,i})$ on a edge $e(v_{i',j'}, v_{j,i})$ represents $log\, p_{i,j}(C_{i'}|C_{j'})$, where $1 \leq i', j' \leq M$. For example, $M(v_{1,2}, v_{2,1}) = log\, p_{1,2}(C_1|C_2)$.

Forming Category Paths

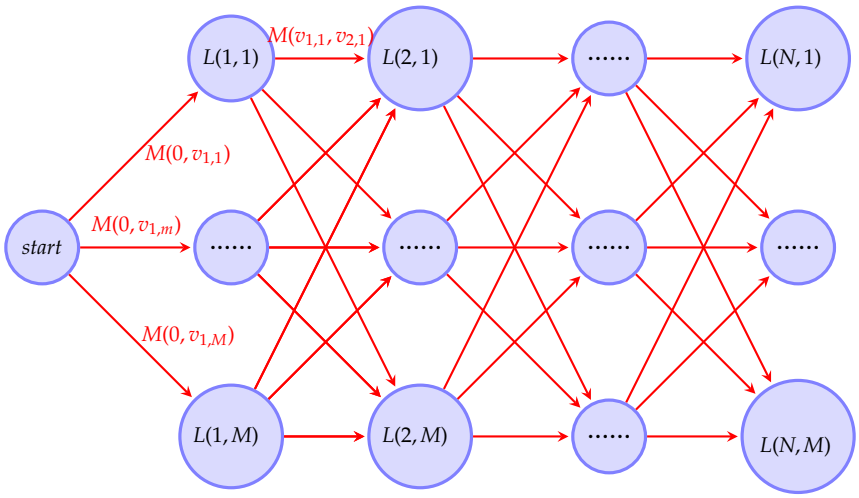


Figure 4.14: The trellis of the Viterbi algorithm

A sequence of optimal class categories $c_1, \ldots, c_N$ for the input sequence $s_1, \ldots, s_N$ is obtained by computing $p(c_1, ...c_N, s_1, ..., s_N) = \sum_{n=1}^{N} log\, p(c_n|c_{n+1}) + log\, p(s_n|c_n)$

$G = (V, E)$ in Figure 4.15 shows that when the algorithm is executing, for each node $v_{i,j} \in V$, there is a weight $w(i, j)$ determined by $w(i, j) = log\,(p(s_i|C_j)) + max_{m'=1,...,M}\,\{w(i-1, m') + log\, p_{i-1,i}\,(C_j|C_{m'})\}$ For example, $w(2, 1) = log\, p(s_2|C_1) + max\{w(1, 1) + logp_{1,2}(C_1|C_1), w(1, 2) + logp_{1,2}(C_1|C_2), \ldots, w(1, M) + log_{1,2}(C_1|C_M)\}$. Moreover, each node $v_{i,j}$ remembers its parent (represented by a solid line). For example, the parent of $v_{2,M}$ is $v_{1,1}$. By the algorithm, a set of M paths can be obtained. For example, one of the paths in Figure 4.15 is $start \rightarrow v_{1,1} \rightarrow v_{2,1} \rightarrow \ldots \rightarrow v_{N,M}$.

Determing the Optimal Path

$G = (V, E)$ in Figure 4.16 shows the optimal path determined by selecting a node that has the smallest value of min $\{w_{N,m}|m = 1, \ldots, M\}$. From this node, we trace back to its parent recursively until we have reached the start node. The optimal path in Figure 4.16 is $start \rightarrow v_{1,1} \rightarrow \ldots \rightarrow v_{N,1}$.

4.2.5 An Example

We use an example in Section 4.1.2 to show how the Viterbi algorithm works. Given $C = \{C_1, C_2, C_3\}$, the probabilities of letters $\{A, B, C, D, E\}$ occurring in categories are in Table 4.1, and the transition

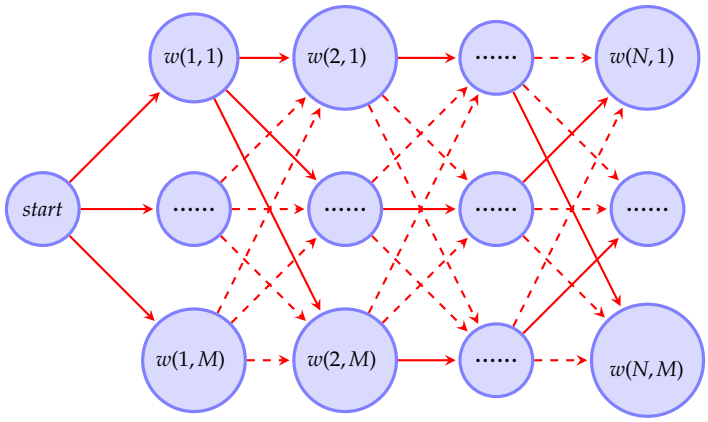


Figure 4.15: Category paths (solid lines) by the Viterbi algorithm
A sequence of optimal class categories $c_1, \ldots, c_N$ for the input sequence $s_1, \ldots, s_N$ is obtained by computeing $p(c_1, ...c_N, s_1, ..., s_N) = \sum_{n=1}^{N} log\, p(c_n|c_{n+1}) + log\, p(s_n|c_n)$

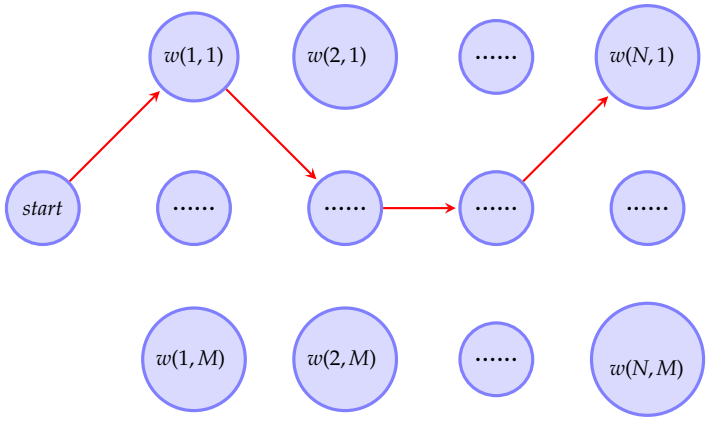


Figure 4.16: An optimal class category path (solid lines) by the Viterbi algorithm
A sequence of optimal class categories $c_1, \ldots, c_N$ for the input sequence $s_1, \ldots, s_N$ is obtained by computing $p(c_1, ...c_N, s_1, ..., s_N) = \sum_{n=1}^{N} log\, p(c_n|c_{n+1}) + log\, p(s_n|c_n)$

probabilities from C_i to C_j are in Table 4.2. We need to find an optimal path for the input sequence *ACDBBCE* by Vitrbi algorithm.

Computation of $w(i, j)$:

- $w(1, :)$
  - $w(1, 1) = log(p(A|C_1)) + log(p_{s,1}(C_1)) = -3.3364$
  - $w(1, 2) = log(p(A|C_2)) + log(p_{s,1}(C_2)) = -2.9214$
  - $w(1, 3) = log(p(A|C_3)) + log(p_{s,1}(C_3)) = -3.9214$

- $w(2,:)$
  - $w(2,1) = log(p(C|C_1)) + max\{w(1,1) + log(p_{1,2}(C_1|C_1)), w(1,2) + log(p_{1,2}(C_1|C_2)), w(1,3) + log(p_{1,2}(C_1|C_3))\} = log(p(C|C_1)) + [w(1,2) + log(p_{1,2}(C_1|C_2))] = -5.9803$
  - $w(2,2) = log(p(C|C_2)) + max\{w(1,1) + log(p_{1,2}(C_2|C_1)), w(1,2) + log(p_{1,2}(C_2|C_2)), w(1,3) + log(p_{1,2}(C_2|C_3))\} = log(p(C|C_2)) + [w(1,2) + log(p_{1,2}(C_2|C_2))] = -7.5653$
  - $w(2,3) = log(p(C|C_3)) + max\{w(1,1) + log(p_{1,2}(C_3|C_1)), w(1,2) + log(p_{1,2}(C_3|C_2)), w(1,3) + log(p_{1,2}(C_3|C_3))\} = log(p(C|C_3)) + [w(1,1) + log(p_{1,2}(C_3|C_1))]$ =-7.3953
- $w(3,:)$
  - $w(3,1) = log(p(D|C_1)) + max\{w(2,1) + log(p_{2,3}(C_1|C_1)), w(2,2) + log(p_{2,3}(C_1|C_2)), w(2,3) + log(p_{2,3}(C_1|C_3))\} = log(p(D|C_1)) + [w(2,1) + log(p_{2,3}(C_1|C_1))] =$ -11.0392
  - $w(3,2) = log(p(D|C_2)) + max\{w(2,1) + log(p_{2,3}(C_2|C_1)), w(2,2) + log(p_{2,3}(C_2|C_2)), w(2,3) + log(p_{2,3}(C_2|C_3))\} = log(p(D|C_2)) + [w(2,1) + log(p_{2,3}(C_2|C_1))] =$ -9.4542
  - $w(3,3) = log(p(D|C_3)) + max\{w(2,1) + log(p_{2,3}(C_3|C_1)), w(2,2) + log(p_{2,3}(C_3|C_2)), w(2,3) + log(p_{2,3}(C_3|C_3))\} = log(p(D|C_3)) + [w(2,1) + log(p_{2,3}(C_3|C_1))] =$ -9.6242
- $w(4,:)$
  - $w(4,1) = log(p(B|C_1)) + max\{w(3,1) + log(p_{3,4}(C_1|C_1)), w(3,2) + log(p_{3,4}(C_1|C_2)), w(3,3) + log(p_{3,4}(C_1|C_3))\} = log(p(B|C_1)) + [w(3,2) + log(p_{3,4}(C_1|C_2))] =$ -13.5131
  - $w(4,2) = log(p(B|C_2)) + max\{w(3,1) + log(p_{3,4}(C_2|C_1)), w(3,2) + log(p_{3,4}(C_2|C_2)), w(3,3) + log(p_{3,4}(C_2|C_3))\} = log(p(B|C_2)) + [w(3,2) + log(p_{3,4}(C_2|C_2))] =$ -14.0981
  - $w(4,3) = log(p(B|C_3)) + max\{w(3,1) + log(p_{3,4}(C_3|C_1)), w(3,2) + log(p_{3,4}(C_3|C_2)), w(3,3) + log(p_{3,4}(C_3|C_3))\} = log(p(B|C_3)) + [w(3,3) + log(p_{3,4}(C_3|C_3))] =$ -12.6242
- $w(5,:)$
  - $w(5,1) = log(p(B|C_1)) + max\{w(4,1) + log(p_{4,5}(C_1|C_1)), w(4,2) + log(p_{4,5}(C_1|C_2)), w(4,3) + log(p_{4,5}(C_1|C_3))\} = log(p(B|C_1)) + [w(4,3) + log(p_{4,5}(C_1|C_3))] =$ - 16.9461
  - $w(5,2) = log(p(B|C_2)) + max\{w(4,1) + log(p_{4,5}(C_2|C_1)), w(4,2) + log(p_{4,5}(C_2|C_2)), w(4,3) + log(p_{4,5}(C_2|C_3))\} = log(p(B|C_2)) + [w(4,3) + log(p_{4,5}(C_2|C_3))] =$ -17.9461

- $w(5,3) = log(p(B|C_3)) + max\{w(4,1) + log(p_{4,5}(C_3|C_1)), w(4,2) + log(p_{4,5}(C_3|C_2)), w(4,3) + log(p_{4,5}(C_3|C_3))\} = log(p(B|C_3)) + [w(4,3) + log(p_{4,5}(C_3|C_3))]$ = -16.3621

- $w(6,:)$

  - $w(6,1) = log(p(C|C_1)) + max\{w(5,1) + log(p_{5,6}(C_1|C_1)), w(5,2) + log(p_{5,6}(C_1|C_2)), w(5,3) + log(p_{5,6}(C_1|C_3))\} = log(p(C|C_1)) + [w(5,3) + log(p_{5,6}(C_1|C_3))]$ = -19.6840
  - $w(6,2) = log(p(C|C_2)) + max\{w(5,1) + log(p_{5,6}(C_2|C_1)), w(5,2) + log(p_{5,6}(C_2|C_2)), w(5,3) + log(p_{5,6}(C_2|C_3))\} = log(p(C|C_2)) + [w(5,3) + log(p_{5,6}(C_2|C_3))]$ = -21.6840
  - $w(6,3) = log(p(C|C_3)) + max\{w(5,1) + log(p_{5,6}(C_3|C_1)), w(5,2) + log(p_{5,6}(C_3|C_2)), w(5,3) + log(p_{5,6}(C_3|C_3))\} = log(p(C|C_3)) + [w(5,3) + log(p_{5,6}(C_3|C_3))]$ = -20.0991

- $w(7,:)$

  - $w(7,1) = log(p(E|C_1)) + max\{w(6,1) + log(p_{6,7}(C_1|C_1)), w(6,2) + log(p_{6,7}(C_1|C_2)), w(6,3) + log(p_{6,7}(C_1|C_3))\} = log(p(E|C_1)) + [w(6,1) + log(p_{6,7}(C_1|C_1))]$ = - ∞
  - $w(7,2) = log(p(E|C_2)) + max\{w(6,1) + log(p_{6,7}(C_2|C_1)), w(6,2) + log(p_{6,7}(C_2|C_2)), w(6,3) + log(p_{6,7}(C_2|C_3))\} = log(p(E|C_2)) + [w(6,1) + log(p_{6,7}(C_2|C_1))]$ = - 24.7429
  - $w(7,3) = log(p(E|C_3)) + max\{w(6,1) + log(p_{6,7}(C_3|C_1)), w(6,2) + log(p_{6,7}(C_3|C_2)), w(6,3) + log(p_{6,7}(C_3|C_3))\} = log(p(E|C_3)) + [w(6,1) + log(p_{6,7}(C_3|C_1))]$ = -23.3279

The Optimal Path

$G = (V, E)$ in Figure 4.17 represents three paths formed by the algorithm. Compared with the value of $w(i, j)$ in the last column, we found that -23.3279 is the maximum. Therefore, we select the node that has the value. Based on this node, we trace back its parent recursively to the start node, we find the optimal category path for $ACDBBCE$ is $C_2C_1C_3C_3C_3C_1C_3$. In this case, it happens to be the same as we have obtained by the Baum-welch algorithm in Section 4.1.2.

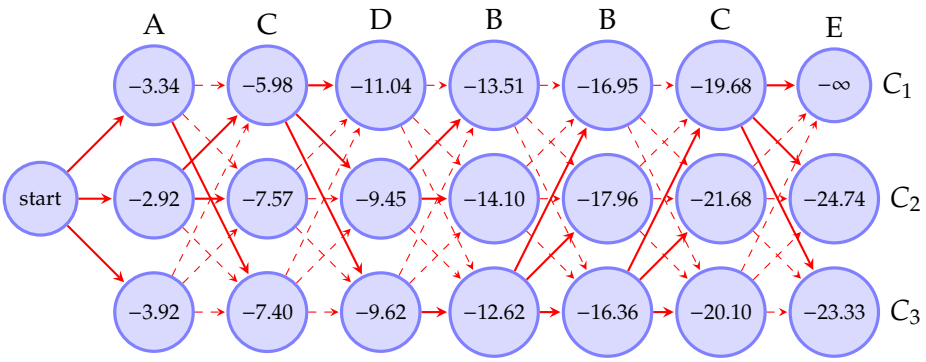


Figure 4.17: The trellis of the Viterbi algorithm for the input sequence *ACDBBCE*.

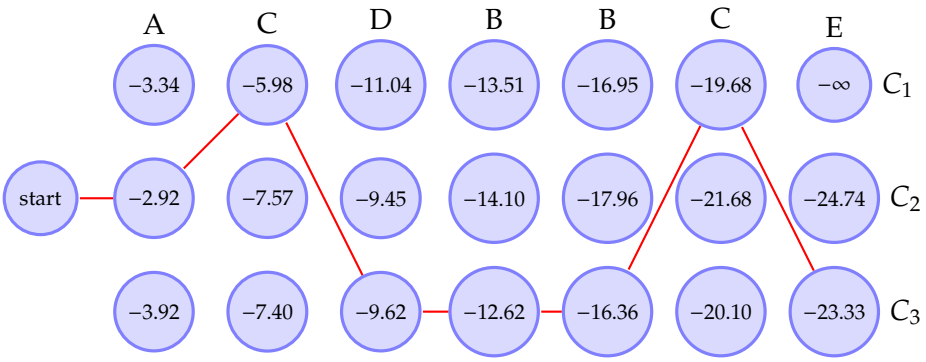


Figure 4.18: The optimal class category path for the input sequence *ACDBBCE* obtained by the Viterbi algorithm.

Chapter 5

NLP Tasks Related Semantics

In this chapter, we discuss three NLP tasks related semantics. They are word sense disambiguation, *NP* chunking, and semantic role labeling. In our research, these tasks are considered as the classification problems. Therefore, we discuss existed classification methods for these tasks. Moreover, we give examples to demonstrate the procedures of some methods in details.

5.1 Word Sense Disambiguation

In the English language, many words have more than one possible meaning. For instance, the word *bank* can be a place where one keeps money or the edge of a river; the word *plant* can be an industrial building or a living organism. In linguistic terms, these words are called polysemous words. The different meanings of a polysemous word are recognized as senses. And the process of determining which sense has been used in a particular context is known as word sense disambiguation (WSD). Word sense disambiguation is an important task in natural language understanding, information retrieval and machine translation.

The WSD task is to determine the contexts or features of a polysemous word, and then, to find the sense of the word. It can be seen as a classification problem or a clustering problem. In our research, WSD is considered as a classification problem.

5.1.1 A Polysemous Word Representation

Two types of contexts are defined in WSD: global contexts and local contexts. While the global contexts can be several paragraphs in a document where the target word is embedded, the local contexts can be N words to one (or several) sentences around the target word. The local contexts may be directly used for WSD algorithms [19], [20], [21]. However, most WSD algorithms need to find features from contexts [22], [23], [24]. These features are determined by terms from linguistic theory and hypotheses, such as morphology, POS, collocations, subcategories, semantic word associations, and syntactic relations [25]. The statistical techniques in Table 5.1 are used by most of the WSD algorithms. In the table, w represents a word, d represents a document, C represents a collection, O represents observations, and $E[x]$ represents the expectation of x. Term frequency is represented by $tf_{i,j}$, which is the total number of times of word w_i appears in the document d_j. Document frequency is represented by df_i, which is the total number of documents in which the word w_j is embedded. Collection frequency is represented by Cf_i, which is the total number of times of word w_i appears in collection C.

The Chi-square test is represented by $\mathcal{X}^2$. It is statistically used for making a goodness of fitting hypothesis test. It equals the square of the observed of data D minus the expectation of data D then divided by the expectation of the data D. By the $\mathcal{X}^2$ test, the *null hypothesis* can be tested. The *null hypothesis* is rejected if the $\mathcal{X}^2$ value is larger than the level of significance. The entropy of a random variable X is represented by $H(X)$. It is the negative of the sum of each probability value $x \in X$ times the logarithm of the probability value of x. The mutual information for random variables X and Y is represented by $I(X, Y)$. It is the entropy of X minus the entropy of X given by Y or the entropy Y minus the entropy of Y given by X. The information gain is represented by $IG(X, a)$. It is the entropy of X minus the entropy of X given when $X = a$. The Kullback-Leibler divergence of the probability distribution p with respect to probability distribution q is represented by $D(p||q)$, which is the sum of $p(x)$, times the logarithm of $p(x)$ divided by the logarithm of $q(x)$.

Table 5.1: Techniques of feature extraction

Name	Notation	Expression
Term frequency	$tf_{i,j}$	$\#\{w_i \mid w_i \in d_j\}$
Document frequency	df_i	$\#\{d \mid w_i \in d\}$
Collection frequency	cf_i	$\#\{w_i \in C\}$
Chi-square test	$\mathcal{X}^2$	$\frac{(O(D)-E[D])^2}{E[D]}$
Entropy	$H(X)$	$H(X) = -\Sigma_{x \in X} p(x) log p(x)$
Mutual information	$I(X;Y)$	$H(X) - H(X\|Y) = H(Y) - H(Y\|X)$
Information gain	$IG(X,a)$	$H(X) - H(X\|a)$
Kullback-Leibler divergence	$D(q\|\|p)$	$\Sigma_{x \in X} p(x) log \frac{p(x)}{q(x)}$

5.1.2 Classifying Senses of a Polysemous Word

We are given a set of training samples. Each of the samples can be a sequence of symbols or an ontology containing a target word. The target word has a set of senses (classes). From the training set, a classifier is built so that it can be used to recognize the polysemous word in a new incoming sentence.

5.1.3 Naive Bayes Method

The use of the Naive Bayes classifier was proposed by Gale [21] in 1992. In this algorithm, a polysemous word is represented by the words around it in a large context window C. Let S be a set of senses for the polysemous word. The polysemous word is classified to be the sense s', if and only if

$$p(s'|C) > p(s|C) \qquad s \in S \tag{5.1}$$

In this algorithm, two assumptions are made. First, the ordering of words within the context is ignored. Moreover, every word in the context is independent of every other word. Thus, the Naive Bayse model is a bag of words model. Let context

$$C =< v_1, v_2, \ldots, v_{N-1}, v_N >$$

then,

$$p(s'|C) = \frac{p(C|s')p(s')}{\sum_{s \in S} p(C|s)p(s)}$$

To maxmize $p(s'|C)$ is to maxmize $p(C|s')p(s')$.

$$\begin{aligned} p(C|s')p(s') &= p(v_1, \ldots, v_N|s')p(s') \\ &= p(s') \prod_{i=1}^{N} p(v_i|s') \end{aligned} \tag{5.2}$$

To maximize $p(s'|C)$ is to maximize $log\, p(C|s')p(s')$. We take log for the both sides of Equation (5.2).

$$log\, p(C|s')p(s') = log\, p(s') + \sum_{i=1}^{N} log\, p(v_i|s') \tag{5.3}$$

Although the Naive Bayes assumption is incorrect in the context of text processing, it is useful in many NLP applications. When a new word needs to be disambiguated, for each sense of that ambiguous word, based on Equation (5.3), a score for the context is calculated by summing up the scores of every word and disambiguation is performed by summing up scores of all the word contexts and adding it to the sense's prior score and choosing the sense with highest score. There are many papers that use the Naive Bayes approach including [26], [24], and [25].

5.1.4 Decision List Method

The decision list approach was proposed by Yarowsky [27]. The decision list algorithm has the advantage of relying on only the strongest features with certain collocations rather than a bag of words. It is one of the popular methods in WSD. The algorithm was developed based on one of the linguistic hypotheses that the nearby words provide strong and consistent clues to the sense of a target word. The decision list algorithm selects the strongest collocational features from the context of a ambiguous word. Let F be a set features. Let S be a set of senses for the polysemous

word. For each $f_i \in F$ and each $s \in S$, the algorithm computes the probability $p(s|f_i)$:

$$p(s|f_i) = \frac{p(f_i|s)p(s)}{\sum_{s' \in S} p(f_i|s')p(s')}$$

Then, the algorithm defines the weight of the feature f_i on sense s as the logarithm of the probability of sense s given by the feature f_i divided by the sum of the probabilities for each sense s', where $s' \neq s$ given by the feature f_i. This indicates the likelihood of a particular sense given a particular feature value. The mathematical expression is represented as Equation (5.4).

$$w(s, f_i) = log \frac{p(s|f_i)}{\sum_{s' \in S, s' \neq s} p(s'|f_i)} \tag{5.4}$$

Then the algorithm orders these features in ascending order by their weights to make a decision list. When testing a new unknown polysemous word, the decision list is checked and the feature with the highest weight that matches the test data are selected as the winner.

5.1.5 Decision Tree Method

The decision tree was proposed by Quinlan [28]. It is useful when a class is represented by several informative features. Moreover, it is intuitive and easy to interpret.

In the decision tree representation, each internal node tests a feature based on a threshold, each branch corresponds to the threshold of the test, and each leaf node assigns a classification. A general decision tree training process includes two procedures: tree growing and tree pruning. In the tree growing procedure, two criteria are required: a splitting criterion is for finding the feature and the threshold that we will split on and a stopping criterion determines when to stop splitting. The stopping criterion is trivial when all elements at a node have the same class so that there is no need to further distinguish them. The splitting criteria in this case is maximum information gain.

$$\begin{aligned} & G(t, f, A) = H_{c \in C}(t) - H_{c \in C}(t|f, A) \\ = \; & H_{c \in C}(t) - (p_l \cdot H_{c \in C}(t_l) + p_r \cdot H_{c \in C}(t_r)) \end{aligned}$$

$$= \quad -\sum_{c \in C} p(c)\ log_2\ p(c) - (-p_l \cdot \sum_{c_l \in C} p(c_l)\ log_2\ p(c_l) - p_r \cdot \sum_{c_r \in C} p(c_r)\ log_2\ p(c_r))$$

Where: f is a feature; A is a threshold of f; C is a set of possible senses; t is the set of training instances that come into the root node of a decision tree. Based on A, we split t into t_l and t_r; p_l is the probability of t_l given by t; and p_r is the probability of t_r given by t. Let N be the total training instances, then, $p_t = \frac{|t|}{N}$, $p_l = \frac{|t_l|}{|t|}$, $p_r = \frac{|t_r|}{|t|}$. We go through all possible partitions of the classes present at the node For each class partition go through all possible distinctions. For each class partition and each way of distinction, evaluate the result. Select the best partition and the best way of distinction.

An example

Computing the information gain for the first node n_0, $G(\{\{c_1\}, \{c_2, c_3\}\}, w_1, 2)$, where $C = \{c_1, c_2, c_3\}$

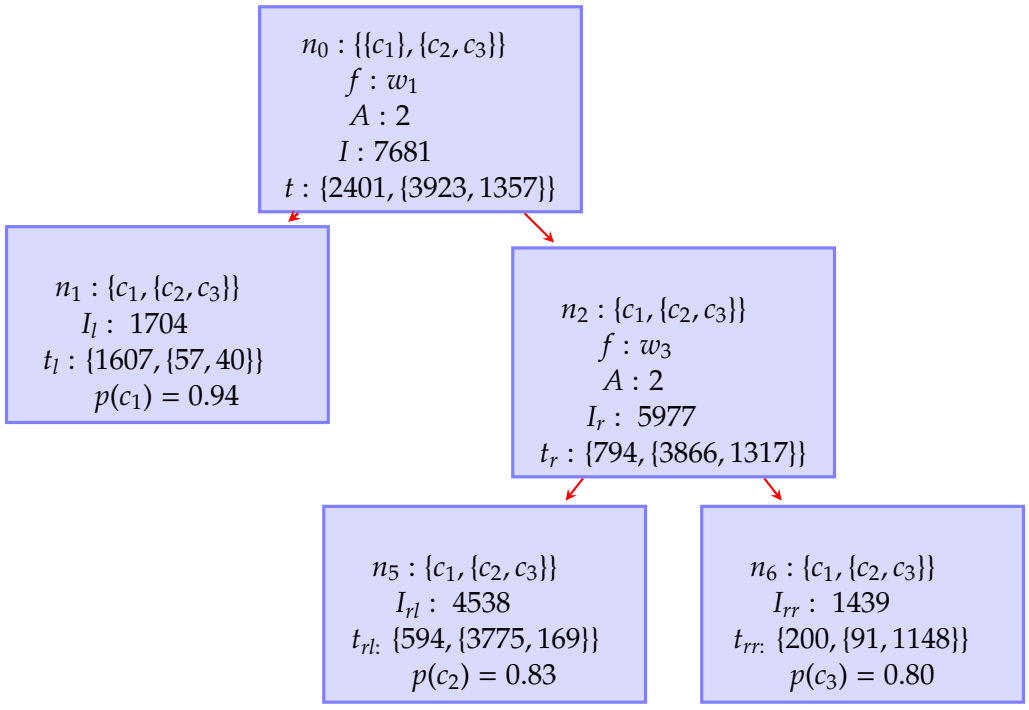


Figure 5.1: A decision tree

This tree determines whether a polysemous word is the sense c_1, the sense c_2, or the sense c_3. At the node n_1, we can assign a word to the sense c_1 with a probability 94%. At the node n_5, we can assigned a word to the sense c_2 with a probability 83%. At the node n_6, we can assigned a word to the sense c_3 with a probability 80%.

- At node n_0, among 7681 instances, there are 2401 instances that belongs to $\{c_1\}$ and 5280

instances that belongs to $\{c_2, c_3\}$. Therefore,

- $H(t) = -\sum_{c \in \{\{c_1\},\{c_2,c_3\}\}} p(c) log_2\ p(c) = -\frac{2401}{7681} log_2\ \frac{2401}{7681} - \frac{5280}{7681} log\ \frac{5280}{7681} = 0.52 + 0.37 = 0.89$
- At node n_1, among 1704 instances, there are 1607 instances that belongs to c_1. There are $1704 - 1607 = 97$ instances that belongs to $\{c_2, c_3\}$
- Therefore, $H(t_l) = -\sum_{c_l \in \{c_1,\{c_2,c_3\}\}} p(c_l) log_2\ p(c_l) = -\frac{1607}{1704} log_2\ \frac{1607}{1704} - \frac{97}{1704} log_2\ \frac{97}{1704} = 0.32$
- At node n_2, among $7681 - 1704 = 5977$ instances, there are 794 instances for $\{c_1\}$. Therefore, there are 5183 instances that belong to $\{c_2, c_3\}$.
- Therefore, $H(t_r) = -\sum_{c_r \in \{c_1,\{c_2,c_3\}\}} p(c_r) log_2\ p(c_r) = -\frac{794}{5977} log_2\ \frac{794}{5977} - \frac{5183}{5977} log_2\ \frac{5183}{5977} = 0.57$
- Hence, $G(\{\{c_1\}, \{c_2, c_3\}\}, w_1, 2) = H(t) - (p_l \cdot H(t_l) + p_r \cdot H(t_r) = 0.89 - (\frac{1704}{7681} \cdot 0.32 + \frac{5977}{7681} \cdot 0.57) = 0.32$

The information gain $G(\{c_1, \{c_2, c_3\}\}, w_3, 2)$ for the second node n_2 is:

- From the previous step, we have:
  $H(t_r) = -\sum_{c_r \in \{c_1,\{c_2,c_3\}\}} p(c_r) log_2\ p(c_r) = 0.57.$
- At node n_5, among 4538 instances, we have 594 instances that belong to c_1. Therefore, we have $4538 - 594 = 3944$ instances that belong to $\{c_2, c_3\}$.
- $H(t_{rl}) = -\sum_{c_{rl} \in \{c_1,\{c_2,c_3\}\}} p(c_{rl}) log_2\ p(c_{rl})$
  $= -\frac{594}{4538} log_2\ \frac{594}{4538} - \frac{3944}{4538} log_2\ \frac{3944}{4538} = 0.56$
- At node n_6, among $5977 - 4538 = 1439$ instances, we have 200 instances that belong to c_1. Therefore, we have $1439 - 200 = 1239$ instances that belong to $\{c_2, c_3\}$
- $H(t_{rr}) = -\sum_{c_{rr} \in \{c_1,\{c_2,c_3\}\}} p(c_{rr}) log_2\ p(c_{rr})$
  $= -\frac{200}{1439} log_2\ \frac{200}{1439} - \frac{1239}{1439} log_2\ \frac{1239}{1439} = 0.58$
- $G(\{c_1, \{c_2, c_3\}\}, w_3, 2) = 0.57 - (\frac{4538}{5977} * 0.56 + \frac{1439}{5977} * 0.58) = 0.0052$

Leaves of the tree correspond to an assignment. For this example, at node n_1, we can assign a polysemous word to the sense c_1 with a probability of 94% of being correct. At node n_5, we can

assign a polysemous word to the sense c_2 with a probability of 83% of being correct. At node n_6, we can assign a polysemous word to the sense c_3 with a probability of 80% of being correct. Once a tree is fully grown, it needs to be pruned to avoid over-fitting and to optimize the performance on new data. When testing new data, starting from the root of a decision tree where feature f will be corresponded to threshold A, the node determines whether or not feature f of a data input is greater than or equal to A. Depending on this answer, it branches to the appropriate child node, and repeats this process until a leaf node is reached. Because decision trees split the training set into smaller and smaller, this makes correct generalization harder and incorrect generalization easier when there are long branches.

5.1.6 Vector Space Model (VSM)

The vector space model was introduced by Salton et al [29] in the field of information retrieval and has shown very good performance. In this model, a document can be represented as a vector in a space where the axes (or dimensions) correspond to words. The coordinates or term weights of these vector are usually derived from occurrence counts, or a function of a term frequency (how many times the term appear in a document), or document frequency (how many documents contain the term), or both. This model is illustrated by Figure 5.2. Real world examples have a vocabulary of several thousands of words but here for simplicity, a vocabulary of two words (i.e., car and insurance) is assumed. So, the lexical space has only two dimensions. Four vectors are represented by a query q and three documents d_1, d_2, d_3 as follows: the similarity between the q and d_i can be measured by cosine measure or normalized correlation coefficient. If the vectors are normalized, the similarity between these two vectors can be computed by a simple dot product.

In the case of *WSD*, a polysemous is represented by a set of features. Therefore, the corresponding feature vector can be formed. Let $\overrightarrow{x}$ be the average feature vector from training set for sense s. Let y be a new feature vector from a test instance. By Equation (5.5), we can compute the similarity between x and y to determine whether y belongs to c.

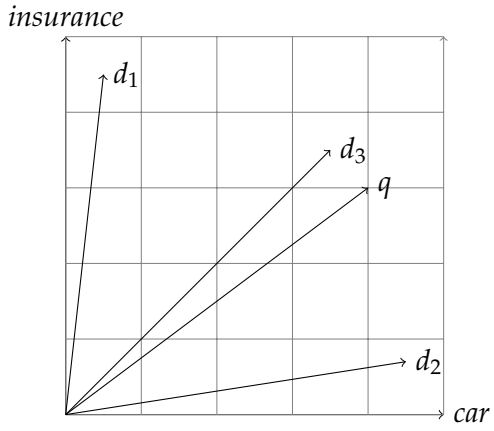


Figure 5.2: A vector space model, d_i represents a document i and q represents a query.

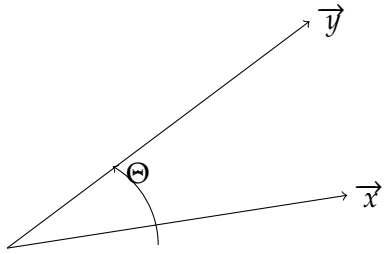


Figure 5.3: Computing θ

$$cos\ \Theta = \frac{\overrightarrow{x}}{||\overrightarrow{x}||} \cdot \frac{\overrightarrow{y}}{||\overrightarrow{y}||} = \frac{\sum_{i=1}^{N} x_i \cdot y_i}{\sqrt{\sum_{j=1}^{N} x_j^2 \cdot \sum_{k=1}^{N} y_k^2}} \tag{5.5}$$

5.2 NP Chunking

Let V be the input vocabulary of the application. Given an input sentence:

$$s =< w_1, \ldots w_N >,$$

where $\forall n, w_n \in V$. Let C be a set of output categories. A category can be an I, or a B, or an O, where I represents a word is inside of a chunk, B represents a word is beginning of a chunk, and O

represents a word is outside of a chunk. The task is to find a sequence output tags, $c =< c_1, \ldots, c_N >$, where $\forall n, c_n \in C$.

5.2.1 Hidden Markov Method

The stochastic hidden Markov models *HMM* are presented in papers [30] [1]. These methods can be stated as follows. Given an input sentence $s =< w_1, \ldots w_N >$, we want to find a sequence output categories, $c =< c_1, \ldots, c_N >$, $\forall n, c_n \in C$, that maximizes probability $p(c|s)$.

$$\begin{aligned} < c_1^*, \ldots, c_N^* > &= \underset{c_1 \ldots c_N}{argmax}\, p(c|s) \\ &= \underset{c_1 \ldots c_N}{argmax}\, \frac{p(s,c)}{\sum_{c \in C} p(s,c)} = \underset{c_1 \ldots c_N}{argmax}\, \frac{p(s|c)p(c)}{\sum_{c \in C} p(s,c)} \\ &= \underset{c_1, \ldots c_N}{argmax} \prod_{j:1,\ldots,N} p(c_j|c_{j-1}, c_{j-2}) \cdot p(w_j|c_j) \end{aligned}$$

Here $p(c_j|c_{j-1}, c_{j-2})$ is called a contextual probability and $p(w_j|c_j)$ is called a lexical probability. The contextual probability is a conditional probability which can be computed by the frequencies of c_i, c_{i-1}, c_{i-2} divided by the frequencies of c_{i-1}, c_{i-2}. The lexical probability is also a conditional probability which can be computed by the frequencies of w_i and a category c_i divided by the frequencies of w_i.

5.2.2 Maximum Entropy Method

Maximum entropy methods *MEM* for finding speech tags and chunks in a sentence are presented in papers [2] and [31]. These methods can be summarized as follows. Let $\mathcal{H}$ be a set of possible word and tag or word and chunk contexts called histories. For example, for a sequence of words $w =< w_1, \ldots, w_N >$ with it POS tags $< c_1, \ldots, c_N >$, the history h_i is $\{w_i, w_{i+1}, w_{i+2}, w_{i-1}, w_{i-2}, c_{i-1}, c_{i-2}\}$. Let C be a set of categories. Let f be a set of feature functions, for each $f_j \in f$, $f_j : \mathcal{H} \times C \rightarrow \{0, 1\}$, s.t, $f_j(h_i, c_i) = 1$ means that condition j is true for history h_i and category c_i, $f_j(h_i, c_i) = 0$ means that

condition j is false for history h_i and category c_i. For example,

$$f_j(h_i, c_i) = \begin{cases} 1 & \textit{if suffix}(w_i) = \textit{ing} \ \& \ c_i = \textit{VBG} \\ 0 & \textit{otherwise} \end{cases}$$

The probability of a category $c \in C$ given by a history $h \in \mathcal{H}$ can be computed as:

$$\begin{aligned} p(c|h) &= \frac{p(c,h)}{\sum_{h \in \mathcal{H}} p(c,h)} = \frac{e^{\sum_{i=1}^{K} \lambda_i \cdot f_i(h,c)}}{\sum_{h \in \mathcal{H}} e^{\sum_{i=1}^{K} \lambda_i \cdot f_i(h,c)}} = \frac{e^{\sum_{i=1}^{K} \lambda_i \cdot f_i(h,c)}}{Z(h)} \\ \textit{where:} \quad & p(c,h) = e^{\sum_{i=1}^{K} \lambda_i \cdot f_i(h,c)} \\ & Z(h) = \sum_{h \in \mathcal{H}} e^{\sum_{i=1}^{K} \lambda_i \cdot f_i(h,c)} \end{aligned} \tag{5.6}$$

Here $f = \{f_1, f_2, \ldots, f_K\}$ is a set of K features and $\{\lambda_1, \lambda_2, \ldots, \lambda_K\}$ is a set of K parameters. These parameters can be determined from the following procedure. Let p be the distribution of $\mathcal{H} \times C$. The entropy of p is defined as;

$$H(p) = -\sum_{(h,c) \in \mathcal{H} \times C} p(h,c) \log_2 p(h,c)$$

Given an sequence of words $w_1, \ldots, w_N$ and categories $c_1, \ldots, c_N$ as training data. Define h_i as the history and c_i as the category. The parameters $\{\lambda_1, \lambda_2, \ldots, \lambda_K\}$ are chosen to maximize the likelihood of p from the training data.

$$L(p) = \prod_{j=1}^{N} p(h_j, c_j) = \prod_{j=1}^{N} e^{\sum_{i=1}^{K} \lambda_i \cdot f_i(h_j, c_j)}$$

We want to maximize the entropy of p subject to the constraint that the expected value of feature f_j equals to the observed feature $\bar{f_j}$, where $1 \leq j \leq K$.

$$\begin{aligned} & E[f_j] = \bar{f}_j \\ \textit{where:} \quad & E[f_j] = \sum_{(h,c) \in \mathcal{H} \times C} p(h,c) f_j(h,c) \approx \sum_{i=1}^{N} \bar{p}(h_i) \cdot p(c_i|h_i) f_j(h_i, c_i) \end{aligned} \tag{5.7}$$

$$\bar{f}_j = \frac{1}{N}\sum_{n=1}^{N} f_j(h_n, c_n)$$

Here $\bar{p}(h_i) = \frac{\#\{m|h_m=h_i, 1\leq m\leq N\}}{N}$ is the prior probability of the history h_i in the training set. Therefore,

$$\begin{aligned}
\frac{1}{N}\sum_{n=1}^{N} f_j(h_n, c_n) &= \sum_{i=1}^{N} \bar{p}(h_i)\cdot p(c_i|h_i)\cdot f_j(h_i, c_i) \\
&= \sum_{i=1}^{N} \bar{p}(h_i)\cdot \frac{e^{\sum_{j=1}^{K}\lambda_j\cdot f_j(h_i,c_i)}}{Z(h)}\cdot f_j(h_i, c_i) \\
&= \sum_{i=1}^{N} \frac{\#\{m|h_m = h_i, 1\leq m\leq N\}}{N}\cdot \frac{e^{\sum_{j=1}^{K}\lambda_j\cdot f_j(h_i,c_i)}}{Z(h)}\cdot f_j(h_i, c_i)
\end{aligned} \tag{5.8}$$

Let $\Gamma_i = \frac{\#\{m|h_m=h_i, 1\leq m\leq N\}}{Z(h)}$. Then Equation (5.8) can be simplified as (5.9):

$$\sum_{n=1}^{N} f_j(h_n, c_n) = \sum_{i=1}^{N} \Gamma_i \cdot e^{\sum_{j=1}^{K}\lambda_j\cdot f_j(h_i,c_i)} \tag{5.9}$$

The model parameters $\{\lambda_1 \ldots, \lambda_K\}$ for the distribution p can be obtained by an iterative method called *Generalized Iterative Scaling* from Equation (5.9) by setting initial values of $\{\lambda_1^{(1)}, \ldots, \lambda_K^{(1)}\}$ to the same constant (say 1). Based on Equation (5.6), we obtain $\{p(c_i|h_i)^{(1)}|i = 1 \ldots N\}$. Combining with Equation (5.8), we update $\{\lambda_1^{(2)}, \ldots, \lambda_K^{(2)}\}$. We repeatedly do it until Equation (5.7) is satisfied. If p has the form (5.6) and satisfies constraints (5.7), then it maximizes the entropy $H(p)$ over the distribution that satisfies the constraints of (5.7) and likelihood $L(p)$ over the distribution of (5.6). For any new input sentence $s =< w_1, \ldots w_N >$, we want to find a sequence output categories, $c =< c_1, \ldots, c_N >, \forall n, c_n \in C$.

$$\begin{aligned}
< c_1^*, \ldots, c_N^* > &= \underset{c_1,\ldots,c_N}{argmax}\, p(c_1, \ldots, c_N|w_1, \ldots, w_N) \\
&= \underset{c_1,\ldots,c_N}{argmax} \prod_{n=1}^{N} p(c_n|h_n) = \underset{c_1,\ldots,c_N}{argmax} \prod_{n=1}^{N} \frac{p(h_n, c_n)}{\sum_{c_n\in C} p(h_n, c_n)}
\end{aligned}$$

5.2.3 Support Vector Machine Method

The support vector machine *SVM* for finding speech tags and chunks in a sentence is presented in papers [32] and [33]. The method can be understood as follows. Let $\{-1, +1\}$ be two classes. Let $\{(x_1, y_1), \ldots, (x_L, y_L)\}$ be a training set, where $x_l \in R^N$, $y_l \in \{-1, +1\}$ is a l^{th} class label; L is the number of the given training samples. A hyperplane is represented as $w \cdot x + b = 0$, where $w \in R^N$ and $b \in R$. We want to find an optimal hyperplane so that it can separate two classes. In order to find it, we need to consider *the margin M* between two classes. In the case that the training data is linearly separable, we can select two parallel hyperplanes in a way that they separate the data and there are no points between them. The margin is the distance between the parallel hyperplanes. Then it tries to maximize the margin. The support vector machine method maximizes the margin. Therefore, the problem can be stated as:

$$\begin{aligned} &\textit{Minimize:} && \frac{1}{2}\|w\|^2 \\ &\textit{Subject to:} && y_k[(w \cdot x_i) + b] \geq 1 \quad k = 1, \ldots K; \;\; y_k \in \{-1, +1\} \end{aligned}$$

*SVM*s are binary classifiers. In this case, we need to extend *SVM*s to multi-class classifiers in order to classify a set of three classes: $\{B, I, O\}$. Basically, two methods can be used to extend a binary classifier to identify K classes. One is one class *vs*. all classes, that is to build K binary classifiers to separate one class from others. The other is pairwise classification, also called *round robin classification*. This needs to have $\frac{K \times (K-1)}{2}$ classifiers. Then, the final decision is achieved by their weighted voting [34].

Pairwise Classification

Pairwise classification reduces a multi-class problem to multiple two-class problems by learning one classifier for each pair of classes, using only training examples for these two classes and ignoring all others [34] and [35].

Let $C = \{C_j | 1 \leq j \leq M\}$ be a set of M classes. Let $T = \{(t_i, c_i) | 1 \leq i \leq N\}$ be the training set, where t_i is an observed sequence and $c_i \in C$. Let r_{ij} be a binary classifier that returns probabilities of pairwise

class C_i and C_j based on T, s.t.

$$\begin{aligned} p(C_i|C_i \vee C_j) &= \frac{\#\{t_n|c_n = C_i\}}{\#\{t_n|c_n = C_i \vee c_n = C_j\}} \\ p(C_j|C_i \vee C_j) &= 1 - p(C_i|C_i \vee C_j) \end{aligned}$$

Therefore, for a set of M classes, we have $\binom{M}{2} = \frac{M!}{(M-2)!2!} = \frac{M \cdot (M-1)}{2}$ such binary classifies. For any new input sequence, we assign the class that receives the maximum number of votes:

$$c' = \underset{i=1,\ldots,M}{argmax} \sum_{j=1}^{M} p(C_i|C_i \vee C_j)$$

5.2.4 Conditional Random Field Method

The conditional random field *CRF* for finding speech tags and chunks in a sentence is presented in papers [3] and [17]. Let X be a random variable over data sequences to be labeled. Let Y be a random variable over corresponding label sequences. All components Y_i of Y are elements in a finite label alphabet $\mathcal{Y}$. We define the conditional random field as follows.

Let $G = (V, E)$ be a graph such that $Y = (Y_v)_{v \in V}$, so that Y is indexed by the vertices of G. Then (X, Y) is a conditional random field conditioned on X. The random variables Y_v obey the Markov property with respect to the graph: $p(Y_v|X, Y_{w,w \neq v}) = p(Y_v|X, Y_{w,w \sim v})$, where $w \sim v$ means that w and v are neighbors in G.

Let $x =< x_1, \ldots, x_N >$ be a data sequence and $y =< y_1, \ldots, y_N >$ be a label sequence. Let f_k and g_k be given and fixed. The conditional probability of $p(y|x)$ is:

$$\begin{aligned} p_\theta(y|x) &\approx e^{\sum_{e \in E,k} \lambda_k f_k(e, y|_e, x) + \sum_{v \in V,k} \mu_k g_k(v, y|_v, x)} \\ &= e^{\sum_{e \in E,k} \lambda_k f_k(e, y|_e, x)} \cdot e^{\sum_{v \in V,k} \mu_k g_k(v, y|_v, x)} \end{aligned} \tag{5.10}$$

Here $y|_s$ is a set of components of y associated with the vertices in subgraph s. The parameters $\theta = \{(\lambda_1, \ldots, \lambda_M), (\mu_1, \ldots, \mu_M)\}$ can be obtained from the training data $\{(x^1, y^1), (x^2, y^2), \ldots, (x^N, y^N)\}$ with the empirical distribution $\bar{p}(x, y)$ by using an iterative scaling algorithm that maximizes the

log-likelihood objective function

$$O(\theta) = \sum_{i=1}^{N} log_2\, p_\theta(y^i|x^i) \propto \sum_{x,y} \tilde{p}(x,y) log_2 p_\theta(y|x)$$

.

In the case that the dependencies of Y conditioned on X forms a chain, illustrated in Figure 5.4, the conditional probability of a label sequence is expressed in matrix form. Suppose that $p_\theta(Y|X)$

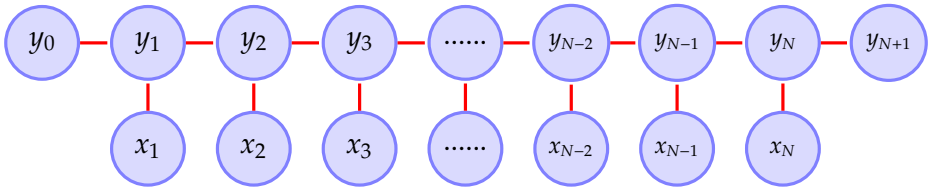


Figure 5.4: The dependencies of Y conditioned on X forms a chain.

is a *CRF* given by Equation (5.10). For each position i in the observation sequence x, we define the $|\mathcal{Y}| \times |\mathcal{Y}|$ matrix random variable $M_i(x) = [M_i(y', y|x)]$ by:

$$M_i(y', y|x) = e^{(\Gamma_i(y', y|x))} = e^{\sum_k \lambda_k f_k(e_i, Y|e_i=(y', y), x) + \sum_k \mu_k g_k(v_i, Y|v_i=y, x)}$$

where e_i is the edge with labels (Y_{i-1}, Y_i) and v_i is the vertex with label Y_i. The normalization (partition function) $Z_\theta(x)$ is the (start, stop) entry of the product of these matrices: $Z_\theta(x) = (M_1(x) \times M_2(x) \times \ldots \times M_{N+1}(x))_{start,stop}$. The conditional probability of a label sequence y is rewritten as:

$$p_\theta(y|x) = \frac{\prod_{i=1}^{N+1} M_i(y_{i-1}, y_i|x)}{Z_\theta(x)}$$

Where: y_0 equals to *start* and y_{N+1} equals to *stop*.

5.3 Semantic Role Labeling

A semantic role is the underlying relationship that a participant has with the main verb in a clause. It is also called thematic relation, thematic role, theta role. Table 5.5 lists a set of semantic roles. A

Figure 5.5: A set of semantic roles

Role name	Description	Example
Agent	deliberately performs the action	Bill ate his soup quietly.
Experiencer	receives sensory or emotional input	The smell of lilies filled Jennifer's nostrils.
Theme	undergoes the action but does not change its state	I like Kim.
Patient	undergoes the action and has its state changed	The falling rocks crushed the car
Instrument	used to carry out the action	Jamie cut the ribbon with a pair of scissors.
Force or Natural Cause	mindlessly performs the action	An avalanche destroyed the ancient temple.
Location	where the action occurs	Johnny and Linda played carelessly in the park.
Goal	what the action is directed towards	The caravan continued on toward the distant oasis.
Recipient	a special kind of goal associated with verbs expressing a change in ownership, possession.	I sent John the letter.
Source	where the action originated	The rocket was launched from Central Command.
Time	the time at which the action occurs	The rocket was launched yesterday.
Beneficiary	the entity for whose benefit the action occurs	I baked Reggie a cake.
Manner	the way in which an action is carried out	With great urgency, Tabatha phoned 911.
Purpose	the reason for which an action is performed	Tabatha phoned 911 right away in order to get some help.
Cause	what caused the action to occur in the first place; not for what, rather because of what	Since Clyde was hungry, he ate the cake.

semantic role can also be labeled with numbers rather than names, s.t.

$$\{ \quad A0, A1, A2, A3, A4, A5, AM\_ADV, AM\_CAU,$$
$$AM\_DIR, AM\_DIS, AM\_EXT, AM\_LOC, AM\_MNR,$$
$$AM\_MOD, AM\_NEG, AM\_PNC, AM\_PRD, AM\_PEC,$$
$$AM\_TMP, R\_A0, R\_A1, R\_A2, R\_A3,$$

R_A4, R_AM_ADV, R_AM_CAU,

R_AM_EXT, R_AM_LOC, R_AM_MNR, R_AM_TMP, V}

Table 5.6 shows the descriptions. Semantic Role labeling is a task that groups sequences words

Figure 5.6: A set of semantic roles labeled with numbers.

Argument of a verb	Description	Related thematic roles of the VerbNet %
$Arg0$	Prototypical Agent	Agent:85, Experiencer:7, Theme:2
$Arg1$	Prototypical patient or theme	Theme:47 , Topic:23, Patient:11
$Arg2$		Recipient:22, Extent:15, Predicate:14
$Arg3$		Asset:33, Theme2:14, Recipient:13
$Arg4$		Location:89, Beneficiary:5
$Arg5$		Location:94, Destination:6
$ArgM-ADV$		general purpose (others)
$ArgM-CAU$		cause (why?)
$ArgM-DIR$		direction (where to?)
$ArgM-DIS$		discourse connectives
$ArgM-EXT$		extent
$ArgM-LOC$		location (where at?)
$ArgM-MNR$		manner (how?)
$ArgM-MOD$		modal verb
$ArgM-NEG$		negation marker
$ArgM-PNC$		purpose
$ArgM-PRD$		predication (refers to or modifies another)
$ArgM-REC$		reciprocal (himself, themselves, each other)
$ArgM-TMP$		temporal (when?)

together in a sentence and classify them by using semantic labels. A sentence can be represented as a a parse tree, a dependency tree, and a sequence symbols, each symbol is a lexicon of a word plus its speech tag.

5.3.1 Linear Interpolation

The linear interpolation algorithm for finding semantic roles filled by constituents in a parse tree is presented in the paper [36].

Let *r* be a semantic role that we want to predict. Let *pt* be a phrase type that indicates a syntactic category of the phrase expressing a semantic role. Let *gov* be a governing category that indicates

the correlation between semantic roles and syntactic realizations, for example, agent and subject. Let *path* be the path from the target word (the predicate) through the parse tree to the constituent in question. Let t be the predicate that invokes the semantic role r. Let *position* be the location that indicates the constituent occurred before or after t. Let *voice* be the voice of t. Let h be a head word of a constituent in question. The probability of a constituent belonging to the semantic role r is defined by the context sum:

$$\begin{aligned} & p(r|constituent) \\ = \; & \lambda_1 p(r|t) + \lambda_2 p(r|pt,t) + \lambda_3 p(r|pt,gov,t) + \lambda_4 p(r|pt,position,voice) \\ & + \lambda_5 p(r|pt,position,voice,t) + \lambda_6 p(r|t) + \lambda_7 p(r|h,t) + \lambda_8 p(r|h,pt,t) \end{aligned}$$

Here, $\{\lambda_1, \ldots, \lambda_8\}$ is a set of parameters, $\lambda_i \geq 0$, and $\lambda_1 + \ldots + \lambda_8 = 1$.
Let f_e be a binary indicator of a given constituent in the parse tree is not in a semantic role. Let *path* be a path from the target word t to the constituents' head word h. The probability that a constituent is a semantic role boundary is defined as:

$$p(fe|constituent) = \lambda_1 p(fe|path) + \lambda_2 p(fe|path,t) + \lambda_3 p(fe|h,t)$$

Here, $\lambda_1, \lambda_2, \lambda_3 \geq 0$ and $\lambda_1 + \ldots + \lambda_3 = 1$.

5.3.2 Dependency Tree

Using a dependency tree for finding semantic roles is presented in paper [37]. The dependency tree of a sentence is generated from the constituency tree (the parser tree) of the sentence. The dependency tree directly encodes the argument structure of lexical units populated at their nodes through dependency relations. For example, Figure 5.7 is a dependency tree of the sentence:

$$\underbrace{The}_{w_0}\ \underbrace{dollar}_{w_1}\ \underbrace{posted}_{w_2}\ \underbrace{gains}_{w_3}\ \underbrace{in}_{w_4}\ \underbrace{quient}_{w_5}\ \underbrace{training}_{w_6}\ \underbrace{as}_{w_7}\ \underbrace{concerns}_{w_8}\ \underbrace{about}_{w_9}\ \underbrace{equities}_{w_{10}}\ \underbrace{abated}_{w_{11}}\ \underbrace{.}_{w_{12}}$$

In the graph, the relations are: $R0$: $p[AM\_LOC]$, $R0.1$: $pobj[O]$, $R0.1.1$: $adj[O]$, $R1$: $obj[A1]$, $R2$: $mod[A0]$, $R2.1$: $det[O]$, $R3$: $obj[AM\_ADV]$, $R3.1$: $in[O]$, $R3.1.1$: $mod[O]$ $R3.1.1.1$: $p[O]$, $R3.1.1.1.1$: $pobj[O]$, $R4$: $punc[O]$.

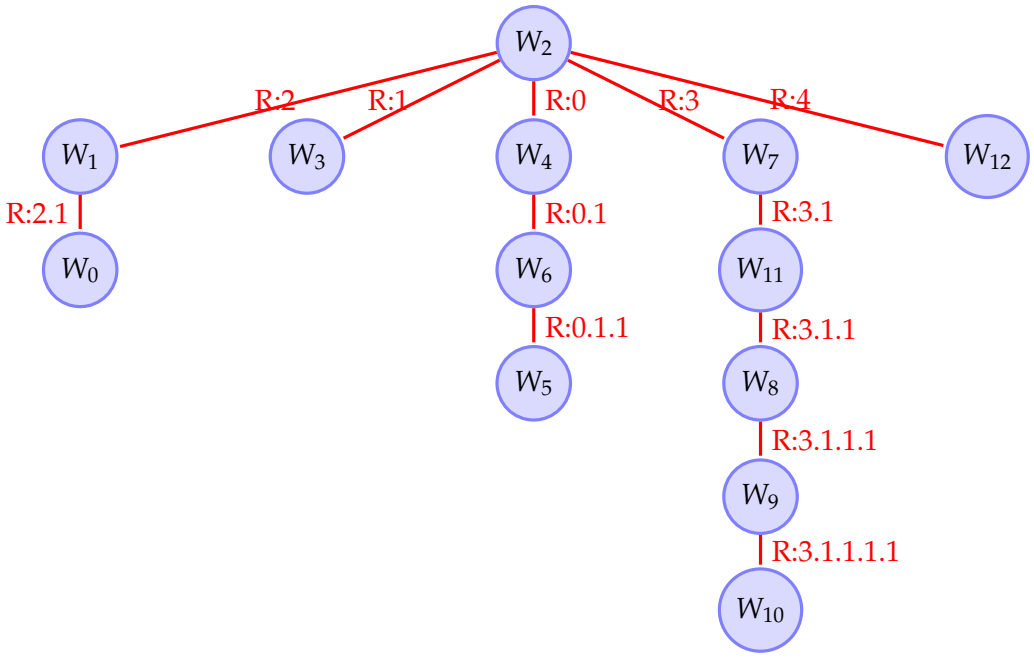


Figure 5.7: The dependency tree associating with the sentence: *The dollar posted gains in quiet training as concerns about equities abated.*

To determine the semantic roles from a set of dependency relations, a set of features are selected. These features can be the type of a dependency relation, the family membership related to the predicate, the position of a headword related to the predicate, the lexicon and POS of the predicate, the lexicon and POS of the dependent word, the path from the node in question to the predicate, the *POS* pattern of the predicate's children, the relation pattern of the predicate's children, the *POS* pattern of the predicate's siblings, and the relation pattern of the predicate's siblings. Then, a set of one-to-all support vector machine classifiers, described in Section 5.2.3 is used to determine the semantic roles.

5.3.3 Tree Conditional Random Field

Using conditional random fields for finding semantic roles from a constituent tree is presented in the paper [38]. Conditional random fields have been discussed in Section 5.2.4. For finding semantic roles, six types of features are considered: basic features, context features, common ancestor of the verb, feature conjunctions, default features, and joint features. Basic features are the head word, *POS* tags of the head word, a phrase's syntactic category, phrase path, position

related to the predicate, surface distance to the predicate, predicate lemma, predicate token, predicate voice, predicate sub-categorization, and syntactic frame. Context features are head word of first *NP* in a preposition phrase, left and right sibling head words and syntactic categories, the first and the last word in a phrase and their *POS* tags, parent syntactic category and head word, and common ancestor of a verb. Feature conjunctions are {*predicate lemma + syntactic category*, *predicate lemma + relative position*, *predicate lemma + first word of the phrase*} Default feature is the prior probabilities over the possible argument labels. Joint features are {*parent head words + child head words*, *parent syntactic categories + child syntactic categories*, *parent relative position + child relative position*, *parent relative position + child relative position + predicate POS tag + predicate lemma*}

5.3.4 Maximum Entropy Modeling

Maximum entropy modeling for finding semantic roles is presented in paper [39]. Maximum entropy modeling has been discussed in Section 5.2.2. In the paper, four types of features are selected. They are shallow features, higher level features, divider features, and *em* features. Shallow features are statistics on the current sequence and its position related to the target, such as, the sequence, the predicate lemma, the length of the sequence in chunks, the frequency of the sequence, the position related to the predicate, the distance to the predicate, and its embedding depth in comparing with the target. High level features are abstraction of a chunk sequence into one of element in the set {*NP*, *VP*, *PP*, *S*, *ADVP*, *ADJP*, *THING*}. Divider features are shallow features and height level features related to the divider sequence, a sequence nested in another sequence (an argument). The *em* features are features based on *EM*−based clustering. They are the verb-argument pair probabilities. Let C be a set of classes. Let $\mathcal{Y}_1$ be a verb alphabet and $\mathcal{Y}_2$ be an argument alphabet. for each pair $y = (y_1, y_2) \in \mathcal{Y}_1 \times \mathcal{Y}_2$, we define:

$$p(y) = \sum_{c \in C} p(c, y) = \sum_{c \in C} p(y|c)p(c) = \sum_{c \in C} p(y_1|c)p(y_2|c)p$$

The feature is obtained by the *EM* algorithm to maximize the incomplete data log-likelihood $L = \sum_y \tilde{p}(y) log_2 p(y)$.

Chapter 6

Methodology of Handling Texual Data

In this chapter, we discuss techniques for reducing noise in textual data. Moreover, we discuss two smoothing methods for handling unseen events.

6.1 Zipf's Law

In the English language, if we count occurrences of each word in a large corpus and list the words in decreasing order based on their frequencies, Zipf's law states that the relationship between the frequency of a word f and its position in the list is:

$$f(k, s, N) = \frac{1}{k^s \sum_{n=1}^{N} \frac{1}{n^s}} = \frac{1}{k^s H_{N,s}} \tag{6.1}$$

where:

- k is the position (rank) of a word in the list ;
- s is a value of the exponent charactering the distribution;
- N is the number of words in the list;
- $H_{N,s}$ is the N^{th} generalized harmonic number.

If $s = 1, f(k, s = 1, N) = \frac{1}{k \sum_{n=1}^{N} \frac{1}{n}} \propto \frac{1}{k}$. In this case, the frequency $f(k, s, N)$ of the k^{th} most common word is about $\frac{1}{k}$ times the frequency of the most commonly occurring word. This is the classic

version of Zipf's law.

Figure 6.1 shows the distributions of f for $1 \leq s \leq 5$ and $N = 200$ under $k = 1$ to 20. The right subfigure of Figure 6.1 is f on $log - log$ scale. By observing Figure 6.1, the second most common frequency will occur $\frac{1}{2}$ as often as the first. The third most common frequency will occur $\frac{1}{3}$ as often as the first. The n^{th} most common frequency will occur $\frac{1}{n}$ as often as the first. Moreover, there are a few high frequency words, a moderate number of medium frequency words, and many low frequency words. For example, Table 6.1 is empirical evaluation of Zipf's law on *Tom Sawyer*

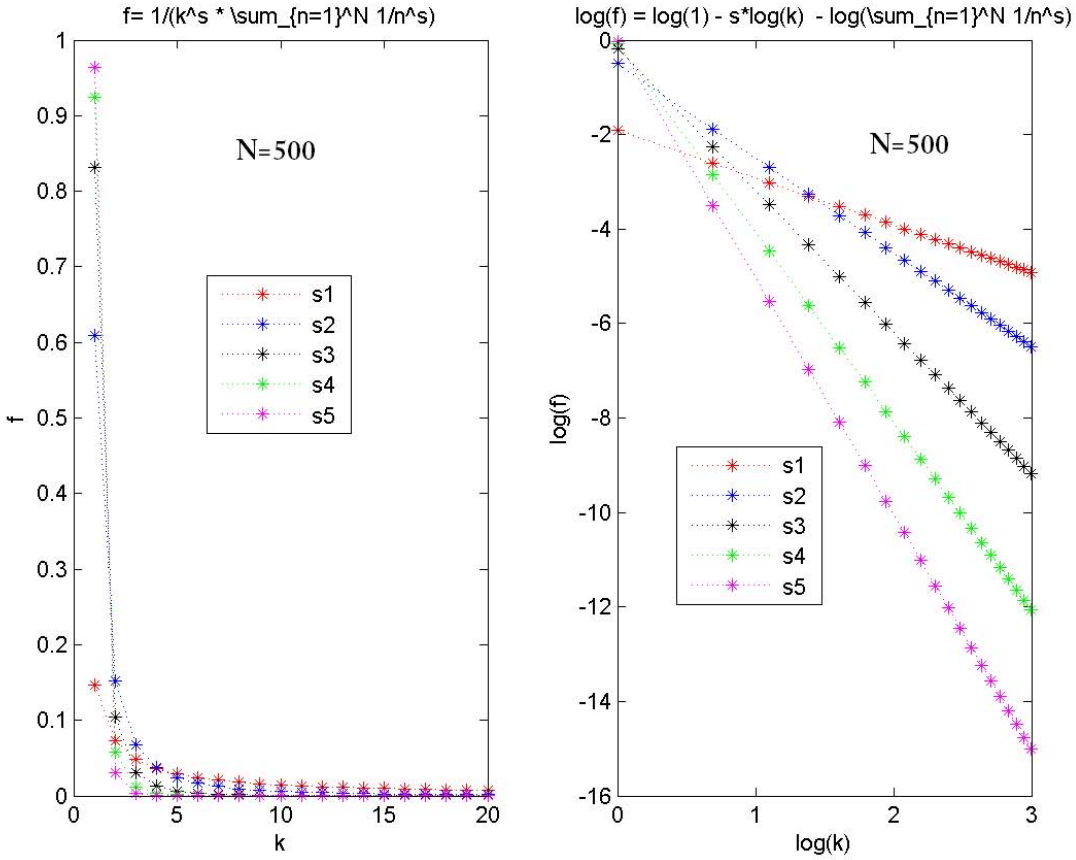


Figure 6.1: Distributions of $f(k, s, N) = \frac{1}{k^s \sum_{n=1}^{N} \frac{1}{n^s}}$, where $1 \leq s \leq 5$, $N = 200$, and $1 \leq k \leq 20$ The *log* is the natural *log*.

data [18].

Table 6.1: Empirical evaluation of Zipf's law on *Tom Sawyer* data

Word	Frequency f	Rank r	$f \cdot r$
the	3332	1	3332
and	2972	2	5944
a	1775	3	5235
he	877	10	8770
but	410	20	8400
be	294	30	8820
there	222	40	8880
one	172	50	8600
about	158	60	9480
more	138	70	9660
never	124	80	9920
oh	116	90	10440
two	104	100	10400
turned	51	200	10200
you'll	30	300	9000
name	21	400	8400
comes	16	500	8000
group	13	600	7800
lead	11	700	7700
friends	10	800	8000
begin	9	900	8100
family	8	1000	8000
brushed	4	2000	8000
sins	2	3000	6000
Could	2	4000	8000
Applausive	1	8000	8000

6.2 Methodology of Reducing Noise

Text preprocessing is a technique of reducing noise in textual data. It basically can be divided into following subcategories: tokenization, removing stop words, stemming, and pruning.

6.2.1 Tokenization

A token is a nonempty sequence of characters, excluding spaces and punctuations, that details all the information of the sentence. Different number and time formats, abbreviations, and acronyms are transformed into standard forms. However, a tokenizer is not always required, it heavily

depends on the tasks and the methods [18]. For example, for the sentence:

S&P today affirmed A + /A − 1 ratings on China Light & Power.

A possible list of tokens can be:

's', '&', 'p', 'today', 'affirmed', 'a', '+', 'a', '1', 'ratings', 'on', 'china', 'light', '&', 'power'

6.2.2 Removing Stopwords

Stop words are function words that can be ignored without a significant effect on output performance of a task. Stop words may be different for different tasks. For example, the following is a small set of stop words. The more complete version of stop words can be found in paper [40].
StopWords = { a, also, an, and, as, at, be, but, by, can, could, do, for, from, go, have, he, her, here, his, how, i, if, in, into, it, its, my, of, on, or, our, say, she, that, the, their, there, therefore, they, this, these, those, through, to, until, we, what, when, where, which, while, who, with, would, you, your }

6.2.3 Stemming

Stemming is a technique to reduce words to their roots [18]. Many words in the English language can be reduced to their base form or stem. For example, *agreed*, *agreeing*, *disagree*, *agreement*, and *disagreement* belong to *agree*. Moreover, names are transformed into the stem by removing the *s*. Foe instance, *Peters* in a sentence is reduced to *Peter* during the stemming process. The following is a selection of suffixes and prefixes for removal during stemming:
suffixes ={ ly, ness, ion, ize, ant, ent , ic, al , ical, able, ance, ary, ate, ce, y, dom , ed, ee, eer, ence, ency, ery, ess, ful, hood, ible, icity, ify, ing, ish, ism, ist, istic, ity, ive, less, let, like, ment, ory, ty, ship, some, ure }
prefixes = { anti, bi, co, contra, counter, de, di, dis, en, extra, in, inter, intra, micro, mid, mini, multi, non, over, para, poly, post, pre, pro, re, semi, sub, super, supra, sur, trans, tri, ultra, un }
However, most stemming algorithms do not remove the prefix of a term. The reason is the huge

impact that the prefix has on the meaning of a sentence. For instance, stemming the word *disagree* to *agree* completely changes the meaning of the sentence. Two frequently used stemmers are *Lovins* [41] and *Porter* [42] stemmers. The two algorithms are based on suffix removal. By setting a set of rules, these algorithms truncate a word by removing inflections that convey parts of speech and tense. For example, *laughing*, *laugh*, *laughs*, and *laughed* are all stemmed to *laugh*; *university* and *universal* both stem to *universe*; *clustering* stems to *cluster*. The difference is that Lovins finds the longest match from a large list of endings while Porter uses an iterative algorithm with a smaller number of suffixes and a set of recoding rules. Two problems with these stemmers are that they conflate semantically different words for example *gallery* and *gall* may both be stemmed to *gall*. Moreover, terms can be unintelligible to users, for example *gallery* is represented by *gall*.

6.2.4 Prunning

Pruning is a scheme for dealing with words appearing rarely or more frequently. These words do not help with identifying objects that are for each task. Rather they add noise and degrade the overall performance. The features that are frequently used for pruning are term frequency, document frequency, and weighted terms.

Term Frequency

The information captured by term frequency is how salient a word is within a given document. The higher the term frequency, the more likely it is that the word is a good description of the content of the document. The term frequency [18] is represented by $tf_{i,j}$. It is the frequency of word w_i in document d_j. It is defined by:

$$tf_{i,j} = \#\{w_i | w_i \in d_j\}$$

There are alternative forms of term frequency. They include $\sqrt{tf_{i,j}}$ or $1 + log\,(tf_{i,j})$.

Document Frequency

The document frequency [18] can be interpreted as an indicator of informativeness. A semantically focussed word will often occur several times in a document while semantically unfocussed words are spread out homogeneously over all documents. Let df_i be the document frequency, which represents the number of documents that term w_i is in. It is defined as:

$$df_i = \#\{d \in D | w_i \in d\}$$

An alternative form of document frequency is $log\,(N) - log\,(df_i)$, where N is the number of documents.

Weighted Term

Let $w(i, j)$ be the weight of term i in document j. It is the combination of term frequency $tf_{i,j}$ and document frequency [18] df_i defined by Equation (6.2). This scheme solves the problem that in reality, not all terms are equally useful. Terms that appear too rarely or too frequently are ranked lower than terms that balance between the two extremes. Higher weight means that the term is more informative to classifiers.

$$w(i,j) = \begin{cases} (1 + log\,(tf_{i,j}))log\,\frac{N}{df_i} & if\ tf_{i,j} \geq 1 \\ 0 & if\ tf_{i,j} = 0 \end{cases} \tag{6.2}$$

6.3 Methodology of Handling Unseen Events

We discuss two smoothing techniques that are frequently used by researchers to assign a probability to events. One is additive smoothing while the other is Good-Turing smoothing.

6.3.1 Additive Smoothing

We have a set of N observations and K bins. An event is that an observation falls into a particular bin. Let x_k be the number of observations that fall into bin k: $x_1 + x_2 +, \ldots, +x_K = N$. Let $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_K\}$ be a set of smoothing parameters, where each $\alpha_k > 0$. Let d_k be the event that an

observation falls into bin k. We compute the probability of d_k given $(x_1, \ldots, x_K)$ as follows:

$$p(d_k|x_1, \ldots, x_K) = \frac{x_k + \alpha_k}{N + \sum_{n=1}^{K} \alpha_n} \tag{6.3}$$

The derivation of Equation (6.3) can be found in Appendix B.1. It is called additive smoothing or Laplace smoothing.

Equation (6.3) produces a non-zero probability event for events that did not occur in the sample. The probability defined by Equation (6.3) guarantees that they sum to one. For example, from an observation *Lisa I am I am Lisa I do not eat*, we have $N = 10$ and $K = 6 + 1 = 7$ (included unseen words). Table 6.2 shows this example. The fourth column shows the probabilities of $p(d_k|N)$ by maximum likelihood estimation (*MLE*). The fifth column show the probabilities of $p(d_k|N)$ by additive smoothing. By comparing these two columns, we found that the more a word occurs, the higher the probability is. If two words occur the same number of times, in the *MLE* computation, we have the same probability. However, in additive smoothing, it is dependent on a value of α. If the α is big, then the probability is big. Otherwise, the probability is small. Row 4 and row 5 of Table 6.2 show these results.

Table 6.2: Computing $p(d_k|N)$
for the example: *Lisa I am I am Lisa I do not eat*

word	k	x_k	α_k	$p(d_k\|N) = \frac{x_k}{N}$ *MLE*	$p(d_k\|N) = \frac{x_k+\alpha_k}{N+\sum_{i=1}^{7}\alpha_i}$ additive smoothing
(unseen word)	1	0	0.8	0	0.043
I	2	3	1.0	0.3	0.215
Lisa	3	2	1.2	0.2	0.172
am	4	2	1.5	0.2	0.188
do	5	1	0.9	0.1	0.102
not	6	1	1.1	0.1	0.113
eat	7	1	2.1	0.1	0.167
				$\sum_{k=1}^{7} p(d_k\|N) = 1$	$\sum_{k=1}^{7} p(d_k\|N) = 1$

Add-one Smoothing

When $\alpha_1 = \alpha_2 = \ldots = \alpha_K = M$, Equation (6.3) is changed to Equation (6.4). Here $p(d_k|x_1, \ldots, x_K) = \frac{x_k+\alpha_k}{N+M\cdot K}$ becomes the symmetric Dirichlet distribution.

$$p(d_k|x_1, \ldots, x_K) = \frac{x_k + M}{N + M \cdot K} \tag{6.4}$$

When $\alpha_1 = \ldots = \alpha_k = 1$, Equation (6.3) is changed into Equation (6.5). In this case, we name it "add-one smoothing".

$$p(d_k|x_1, \ldots, x_K) = \frac{x_k + 1}{N + K} \tag{6.5}$$

We use the previous example to show the procedures of add one smoothing in Table 6.3. The fifth column shows the probabilities of $p(d_k|N)$ for $\alpha_1 = \ldots = \alpha_K = 1$; the sixth column shows the probabilities of $p(d_k|N)$ for $\alpha_1 = \ldots = \alpha_K = 2$; the seventh column shows the probabilities of $p(d_k|N)$ for $\alpha_1 = \ldots = \alpha_K = 3$. By comparing these columns, we see that as the value of α_k increases, the probability of an unseen word increases and the probability of a seen word decreases.

Table 6.3: Computing $p(d_k|N)$ and $p(d_k|I)$
for the example: *Lisa I am I am Lisa I do not eat*

word	k	x_k	$p(d_k\|N) = \frac{x_k}{N}$	$p(d_k\|N) = \frac{x_k+1}{N+K}$	$p(d_k\|I) = \frac{I_k+2}{I+2K}$	$p(d_k\|I) = \frac{I_k+3}{I+3K}$
(unseen word)	1	0	0	0.059	0.083	0.097
I	2	3	0.3	0.235	0.208	0.194
Lisa	3	2	0.2	0.176	0.167	0.161
am	4	2	0.2	0.176	0.167	0.161
do	5	1	0.1	0.118	0.125	0.129
not	6	1	0.1	0.118	0.125	0.129
eat	7	1	0.1	0.118	0.125	0.129

6.3.2 Good-Turing Smoothing

Good-Turing smoothing (Good-Turing frequency estimation) [43] technique is used by [44] to estimate the probability of any word that is unseen in a population. Let number of distinct words

in a population be s. Let the population frequencies of words be $p_1, \ldots, p_s$, s.t.

$$\sum_{i=1}^{s} p_i = 1$$

Let n_r be the number of distinct words occurring exactly r times. Let R be the largest number of distinct words occurring exactly R times. Then

$$n_0 + n_1 + \ldots n_r + \ldots + n_R = s \qquad R \leq s$$

Therefore, the total number of words in the population is:

$$N = \sum_{r=0}^{R} n_r \cdot r = \sum_{r=1}^{R} n_r \cdot r \quad \textit{where} : n_0 \cdot 0 = 0$$

For example, Table 6.4 shows r and n_r for the segment *Lisa I am I am Lisa I do not eat*. There are six words in the population. Three words occur exactly once. Two words occur exactly twice. One word occurs three times. Hence, $s = n_3 + n_2 + n_1 = 1 + 2 + 3 = 6$. The total number of words is $N = \sum_{r=1}^{3} r \cdot n_r = 1 \cdot 3 + 2 \cdot 2 + 3 \cdot 1 = 10$.

Table 6.4: Computing r, p_i, and n_r
for the example of *Lisa I am I am Lisa I do not eat*

i	word	r	p_i	n_r
1	I	3	0.3	n_3=1
2	Lisa	2	0.2	n_2= 2
3	am	2	0.2	
4	do	1	0.1	n_1 = 3
5	not	1	0.1	
6	eat	1	0.1	

Expected value of q_r

Consider the following example. A random sample of size N is drawn from a population of s distinct words. Let q_r be the probability that an arbitrary word is represented r times in the sample. The expected value of q_r is

$$E[q_r] = \frac{(r+1)}{N}\frac{E[n_{r+1}]}{E[n_r]} \approx \frac{(r+1)}{N}\frac{n_{r+1}}{n_r} = \frac{r^*}{N} \tag{6.6}$$

where

$$r^* = (r+1)\frac{n_{r+1}}{n_r} \tag{6.7}$$

The proof of Equation (6.6) can be found in Appendix B.2. From Equation (6.7), the expected probability of all distinct words that are each represented r times in the sample is approximately equals to $(r+1)\frac{n_{r+1}}{N}$. Therefore, the expected probability of all distinct words been represented in the sample is approximately $\frac{(2n_2+3n_3+...+Rn_R)}{N} = 1 - \frac{n_1}{N}$.

Let q_0 be the probability of any word that is unseen. The expected value of q_0 is:

$$E[q_0] = \frac{E[n_1]}{N} \approx \frac{n_1}{N} \tag{6.8}$$

Examples

For the segment *Lisa I am I am Lisa I do not eat*. By observing the table, we have that for $r = 1$, the probability q_r is taken $\frac{4}{9}$ of the original p_r and for $r = 2$, the probability q_r is taken $\frac{3}{8}$ from the original p_r.

$$r^* = (r+1)\frac{E[n_{r+1}]}{E[n_r]}$$

$$q_r = \frac{r^*}{N}$$

Here is another example. In the sentence *We are finishing and have caught 18 fish*, it consists of 10

Table 6.5: Computing r^* and $E[q_r]$
for the example *Lisa I am I am Lisa I do not eat*

word	r	n_r	p_r	r^*	$E[q_r]$
	0				$\frac{3}{10} = 0.3$
am, do, not	1	3	$\frac{3}{10} = 0.3$	$\frac{4}{3}$	$\frac{4}{30} = 0.13$
Lisa, am	2	2	$\frac{4}{10} = 0.4$	$\frac{3}{2}$	$\frac{3}{20} = 0.15$
I	3	1	$\frac{3}{10} = 0.3$		$1 - \frac{3}{10} - \frac{4}{30} - \frac{3}{20} = \frac{5}{12} = 0.42$

carp, 3 perch, 2 whitefish, 1 trout, 1 salmon, and 1 eel. Table 6.6 shows the statistics of frequency r and frequency of frequency N_r. We compute the expected probability of the next fish is a new fish

Table 6.6: A *fish* example

fish	r	n_r
trout	1	
salmon	1	3
eel	1	
whitefish	2	1
:	:	:
carp	10	1

(such as a catfish or a bass) by:

$$E[q_0] = \frac{E[n_1]}{N} = \frac{3}{18}$$

with an error of:

$$Var(q_0) = E[q_0](E[q_1] - E[q_0]) = \frac{3}{18} \cdot abs(\frac{1}{27} - \frac{3}{18}) = 0.022$$

After this consideration, we want to compute the probability, for example, that the next fish is a trout by

$$\begin{aligned} r^* &= (r+1)\frac{E[n_{r+1}]}{E(n_r)} = (1+1) \cdot \frac{1}{3} = \frac{2}{3} \\ E[q_1] &= \frac{r^*}{N} = \frac{2}{3 * 18} = \frac{1}{27} \end{aligned}$$

$$Var(q_1) \quad = \quad E[q_1](E[q_2] - E[q_1]) = \frac{1}{27} \cdot abs(0 - \frac{1}{27}) = 0.001$$

Chapter 7

Developing Algorithms and Experiments

In this chapter, based on the models developed in Chapter 3, we create an algorithm for identifying the semantic arguments of a verb, an algorithm for identifying the sense of a polysemous word, and an algorithm for identifying the noun phrases of a sentence. We discuss the evaluation matrices and the statistical testing methods. We test the data sequence dependence model (*DSDM*), the context independent model (*CIM*), and the class sequence dependence model (*CSDM*) for identifying these text patterns on the standard data sets. We also test the hidden Markov model (*HMM*) discussed in Section 4.2.1 for identifying *NP* chunks and the maximum entropy model (*MEM*) discussed in Section 5.2.2 for identifying word senses on the standard data sets.

7.1 Evaluation Metrics

7.1.1 Contingency table

Let C be the true class of an instance. Let $\neg C$ designate that an instance is not C. Table 7.1 represents a contingency table. In the table, we have

- $\alpha = p_{TA}(C, C)$ represents the fraction of instances where the true class is C and the assigned class is C;
- $\beta = p_{TA}(C, \neg C)$ represents the fraction of instances where the true class is C but the assigned class is $\neg C$;

Table 7.1: A contingency table

$true \rightarrow$ $assigned \downarrow$	C	$\neg C$
C	α	β
$\neg C$	γ	δ

- $\gamma = p_{TA}(\neg C, C)$ represents the fraction of instances where the true class is $\neg C$ but the assigned class are C;
- $\delta = p_{TA}(\neg C, \neg C)$ represents the fraction of instances where the true class is $\neg C$ and the assigned class is $\neg C$.

7.1.2 *precision, recall,* and *f-measure*

The *precision*, *recall*, and $f - measure$ are three probabilities that evaluate the performance of our methods.

The *precision* is the fraction of instances whose assigned class is C given that the true class is C. Based on Table 7.1, the *precision* is defined as:

$$precision = p_{A|T}(C|C) = \frac{\alpha}{\alpha + \beta} \tag{7.1}$$

The *recall* is the fraction of instances whose true class is C given that its assigned class is C. Based on Table 7.1, the *recall* is defined as:

$$recall = p_{T|A}(C|C) = \frac{\alpha}{\alpha + \gamma} \tag{7.2}$$

The *f-measure* is the probability that considers both the *precision* and *recall*. In general, it is the weighted harmonic mean of *precision* and *recall*. Based on Table 7.1, the *f-measure* is defined as:

$$\begin{aligned} f - measure \quad &= \quad p_{TA}(C|C) = \frac{1}{\eta \frac{1}{precisiobn} + (1 - \eta)\frac{1}{recall}} \\ \eta \in (0, 1) \end{aligned} \tag{7.3}$$

When η approaches zero, the $f - measure$ is dominated by the *recall*. When η approaches one, the $f - measure$ is dominated by the *precision*. When $\eta = \frac{1}{2}$, the *f-measure* is the harmonic mean of *precision* and *recall*.

7.1.3 *accuracy*

The *accuracy* evaluates the performance of our methods. It is the fraction of instances for which the assigned class equals to the true class. Based on Table 7.1, the *accuracy* is defined as:

$$accuracy = p_{TA}(C, C) + p_{TA}(\neg C, \neg C) = \frac{\alpha + \delta}{\alpha + \beta + \gamma + \delta} = \alpha + \delta \tag{7.4}$$

7.2 Testing Methods

7.2.1 $N - fold$ cross validation

In $N - fold$ cross validation, the original sample is randomly partitioned into N subsamples; each of the subsamples is called a fold. Among the N subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $N - 1$ subsamples are used as training data. The cross validation process is then repeated N times, with each of the N subsamples used exactly once as the validation data. The N results from the folds then can be averaged to produce estimations for *precision*, *recall*, $f - measure$, or *accuracy*.

7.2.2 Monte Carlo Methods

A Monte Carlo method estimates the parameter of a hypothetical population, using a random sequence of numbers to construct a sample of the population, from which statistical estimates of the parameter can be obtained. For example, let X be a random variable with a probability distribution p. The expected value of X is written as $A = E[X]$. If we can generate $X_1, \ldots, X_N$, N independent random variables with the same probability distribution p, then, we can estimate A

by the sample mean $\bar{A}$:

$$\bar{A} = \frac{1}{N}\sum_{i=1}^{N} X_i$$

By the central limit theorem, as $n \to \infty$, $\bar{A} \to A$. In this case, the random numbers are X_i, $1 \leq i \leq N$, the sample is $X_1, \ldots, X_N$, the parameter is $E[X]$, and the estimate for $E[X]$ is $\bar{A} = \frac{1}{N}\sum_{i=1}^{N} X_i$. Monte Carlo methods exist in a variety of forms. However, a general pattern is followed by these methods. This pattern can be described in four steps. First, a random variable X with a probability distribution p is defined. Second, over the distribution p, a set independent random variables $X_1, X_2, \ldots, X_N$ is generated. Third, a deterministic computation is performed on these random variables to estimate the required parameters. Finally, the results is obtained based on the step three.

7.2.3 Hypothesis Testing

Let $d_1, \ldots, d_n$ and $d_{n+1}, \ldots, d_N$ be a random sample of observations from R^K. Let H_0 be the hypothesis that $d_1, \ldots, d_n$ and $d_{n+1}, \ldots, d_N$ come from the same distribution. Let H_1 be the alternative hypothesis that $d_1, \ldots, d_n$ and $d_{n+1}, \ldots, d_N$ come from different distributions. In hypothesis testing, we test the null hypothesis H_0 against the alternative hypothesis H_1. The test is based on a computation of $p - value$.

$p - value$

The $p - value$ is the probability of obtaining a test statistic at least as extreme as the one that was actually observed, assuming that the null hypothesis H_0 is true. To compute $p - value$, from the observations, we compute a set of permutation scores. Let $\{s_0, \ldots, s_M\}$ be a set of such scores. The score s_0 is obtained from the original data, the others are the scores obtained from randomly permuting the data.

$$s_0 = \sum_{i=1}^{n} \min_{n+1 \leq m \leq N} \{\rho(d_m, d_i)\} + \sum_{j=n+1}^{N} \min_{1 \leq m \leq n} \{\rho(d_m, d_j)\}$$

Where $\rho(d_x, d_y)$ is a metric measuring the distance between d_x to d_y. For example, ρ could be the Euclidean distance.

$$\rho(d_x, d_y) = \sqrt{(d_x^1 - d_y^1)^2 + \ldots + (d_x^K - d_y^K)^2}$$

Let $p(s)$ be the estimated probability of a permutation score greater than or equal to s for the observations assuming the null hypothesis H_0 is true, $p(s)$ is defined as:

$$p(s) = \frac{\#\{m|s_m > s\} + \frac{1}{2} * \#\{m|s_m = s\}}{M + 1}$$

Therefore, the estimated $p - value$ is computed as:

$$p - value = \frac{\#\{m|s_m > s_0\} + \frac{1}{2} * \#\{m|s_m = s_0\}}{M + 1} \tag{7.5}$$

Significance Level α

Let α be the significance level; α is the probability of rejecting H_0 when H_0 is true. The popular significance levels are 5%(0.05), 1%(0.01), 0.5%(0.005), and 0.1%(0.001). For example, if $\alpha = 0.05$, if the $p - value$ obtained by the randomly shuffled data (the null hypotheses is true) is less than or equal to α, we reject the null hypothesis. In this case, the probability of rejecting the null hypothesis when in fact the null hypothesis is true is α. We say the result is statistically significant if the $p - value$ obtained by assuming the null hypothesis being true is great than α, then we fail to reject the null hypothesis H_0, and we say that the result is statistically nonsignificant, or statisticlly not significant.

7.3 Experiments of NP Chunking

7.3.1 Developing an Algorithm

Let S be a sequence of symbols associated with a sentence, $S =< s_1, ..., s_i, ..., s_N >$, where s_i is the pair (word, its speech tag). Let C be a set of classes, $C = \{C_1, C_2, C_3\}$, where C_1 represents that a symbol is inside a noun phrase, C_2 represents that a symbol is not in a noun phrase, and C_3 represents that

a symbol starts at a new noun phrase.

Building Blocks

$\mathcal{B}$ is a block if and only if:

1. For some $i \leq j$, $\mathcal{B} = < c_i, c_{i+1}, \ldots, c_j >$
2. $c_i \in \{C_1, C_3\}$
3. $c_n = C_1, n = i+1, \ldots, j$
4. For some $\mathcal{B}'$, if $\mathcal{B}' \supseteq \mathcal{B}$ and $\mathcal{B}'$ satisfies 1, 2, 3 then $\mathcal{B}' \subseteq \mathcal{B}$

The Procedure

1. Find a sequence of class assignments $< c_1^*, \ldots, c_N^* >$ for a sequence of input symbols $< s_1, \ldots, s_N >$, s.t.

$$\begin{aligned} < c_1^*, \ldots, c_N^* > &= \underset{c_1,\ldots,c_N}{argmax}\, p(c_1, \ldots, c_N | s_1, \ldots, s_N) \\ &\propto \underset{c_1,\ldots,c_N}{argmax}\, p(c_1, \ldots, c_N, s_1, \ldots, s_N) \end{aligned}$$

2. Find $\{B_1, \ldots, B_M\}$, where each B_m is a block satisfying the definition of $\mathcal{B}$.
3. Extract an NP from each B_m.

7.3.2 Complexities

Time Complexity

Based on Section 3.4.5, to find a sequence of N optimal class assignments for a sequence of N symbols, if we have a set of M classes, we need to have $O(NM)$ time. Moreover, in order to find a set of blocks, we need to have $O(N)$ time. Furthermore, to extract all NPs from these blocks, we

need to have $O(N)$ time. Therefore, the time complexity of the algorithm is

$$T_c = O(MN) + O(N) + O(N) = O(NM).$$

Memory Complexity

To find a sequence of N optimal class assignments for a sequence of N input symbols, if we have a set of M classes, we need to have $O(M)$ memory space. In order to find a set of blocks, we need to have $O(N)$ memory space. Moreover, to extract all NPs from these blocks, we need to have $O(N)$ memory space. Furthermore, for obtaining a probability table, we need to have $O(N^2M)$ memory space. Therefore, the memory complexity of the algorithm is

$$M_c = O(M) + O(N) + O(N) = O(M) + O(N) + O(N^2M) = O(N^2M).$$

7.3.3 Experiments

In the experiments, we used *WSJ* data from Penn Treebank and $CoNLL - 2000$ shared task data to test the three models. In the first step, we designed three kinds of symbols in a symbol sequence to test *DSDM* on $CoNLL-2000$ shared task data. From this experiment, we learned that a symbol with the lexicon plus the *POS* tag of a word achieved the best performance. Therefore, this symbol type was used for further testing. Moreover, on the same data set, models were tested and compared with the methods that were published by other researchers. Further, these models were tested on *WSJ* data from Penn Treebank. A Monte Carl testing method was employed for finding the best model.

Testing the Data Sequence Dependence Model on $CoNLL - 2000$ **Data**

We tested the second model on $CoNLL - 2000$ Shared Task data. In this experiment, each symbol sequence was associated with a sentence. Three types of symbols in a symbol sequence were utilized. They were the lexicon of a word, the *POS* tag of a word, and the lexicon and the part of speech (*POS*) tag of a word. We used $10 - fold$ cross validation techniques to distribute a training

set and a testing set. For every sentence in the training set, noun phrases, no noun phrases, and break point phrases for two consecutive noun phrases were extracted. The set C_1 contained all noun phrases, the set C_2 contained all no noun phrases, and the set C_3 contained all break point phrases. We set $C = \{C_1, C_2, C_3\}$

We estimated $\hat{p}(s_i|c_i)$ by the number of symbols s_i in c_i divided by the total number of symbols c_i in C. Note, for each c_i, we need to consider $c_i = C_1$, $c_i = C_2$, and $c_i = C_3$. We estimate $\hat{p}(s_{i+1}|s_i, c_i)$ by the number of symbols phrases $s_i s_{i+1}$ in c_i divided by the number of symbol s_i in c_i. We estimate $\hat{p}(s_{i-1}|s_i, c_i)$ by the number of symbol phrases $s_i s_{i-1}$ in c_i divided by the number of symbols s_i in c_i. For a symbol in each testing symbol sequence in the testing set, we assign a class C_k for s_i, when

$$\frac{\hat{p}(s_{i-1}|s_i, C_k)\hat{p}(s_{i-1}|s_i, C_k)\hat{p}(s_i|C_k)\hat{p}(C_k)}{\hat{p}(s_{i-1}|s_i, C_j)\hat{p}(s_{i-1}|s_i, C_j)\hat{p}(s_i|C_j)\hat{p}(C_j)} > \theta \tag{7.6}$$

Note, θ is a threshold. The default value of θ was 1.

We find the optimal class sequence by

$$\begin{aligned} & < c_1^*, c_2^*, ..., c_N^* > \\ = \; & log_2 \prod_{n=1}^{N} \underset{c_n \in C}{argmax}\, \hat{p}(s_{n-1}|s_n, c_n)\hat{p}(s_{n-1}|s_n c_n)\hat{p}(s_n|c_n)\hat{p}(c_n) \\ = \; & \sum_{n=1}^{N} log_2 \underset{c_n \in C}{argmax}\, \hat{p}(s_{n-1}|s_n, c_n)\hat{p}(s_{n-1}|s_n c_n)\hat{p}(s_n|c_n)\hat{p}(c_n) \end{aligned}$$

The training instance distributions are shown in Table 7.2. The evaluation metrics were *precision*,

Table 7.2: The training instance distributions on $CoNLL - 2000$ data

# of possible symbols	# of possible categories	Total amount space taken for the probability table for seen events
15710	3	1.7×10^8

recall, and $f - measure$. The test result was shown in Figure 7.3. For example, for the symbol type being *POS*, the *precision* showed the ratio of the correctly assigned NP chunks over the total number of true NP chunks was 92.27%. The *recall* showed that the correctly assigned NP chunks

over the total number of assigned NP chunks was 93.76%, and the $f-measure$ showed that the average of the *precision* and *recall* was 92.76%. By comparing the $f-measure$s for these three types of symbols, we noticed that if the model was built only on the lexical information, it had the lowest performance 89.75%. The model's performance improved 3% if it was constructed by *POS* tags. The model achieved the best performance of 95.60% if both lexicon and *POS* tags were included.

Table 7.3: Testing *DSDM* for identifying NP chunks on $CoNLL-2000$ data

DSDM	*symbol type*		
	lexicon	*POS*	*POS + lexicon*
precision %	86.27	92.27	95.15
recall %	93.35	93.76	96.05
$f-measure$ %	89.75	92.76	95.60

Another experiment we have done on the CoNLL-2000 shared task data set was to test the relationship between precision and recall under different thresholds of θ as defined in Equation (7.6). The symbol type was still *POS + lexicon* of a word. We set the range of $\theta = 0.01$ to $\theta = 160$. The result was shown in Figure 7.1. The $x-axis$ represents the values of precision, while the $y-axis$ represents the values of recall. From this experiment, we learned that as θ increases, the precision increases and the recall decreases. For example, When $\theta = 1$, the *precision* equals to 95.2% and the *recall* equals to 96.1%. When $\theta = 5$, the *precision* equals to 95.6% and the *recall* equals to 95.6%. When $\theta = 10$, the *precision* equals to 94.8% and the *recall* equals to 95.7%. When θ equals 5, the precision = the recall = 95.6%. Thus, if we want to select a higher precision, we can select $\theta > 5$. If we want to select a higher recall, we can select $\theta < 5$.

Testing Three Models on $CoNLL-2000$ **Data**

We tested the three models on $CoNll-2000$ Shared Task data set. Based on the result obtained from the previous test, the only symbol type in a symbol sequence was *POS + lexicon* of a word. In this test, we still used $10-fold$ cross validation. The results are shown in the Table 7.4. Comparing with the $f-measure$s on three models, we see that *DSDM* achieved the best performance. It is 1.91% better than *CIM* and 2.05% better than *CSDM*. Note, the results in the fourth column are results for

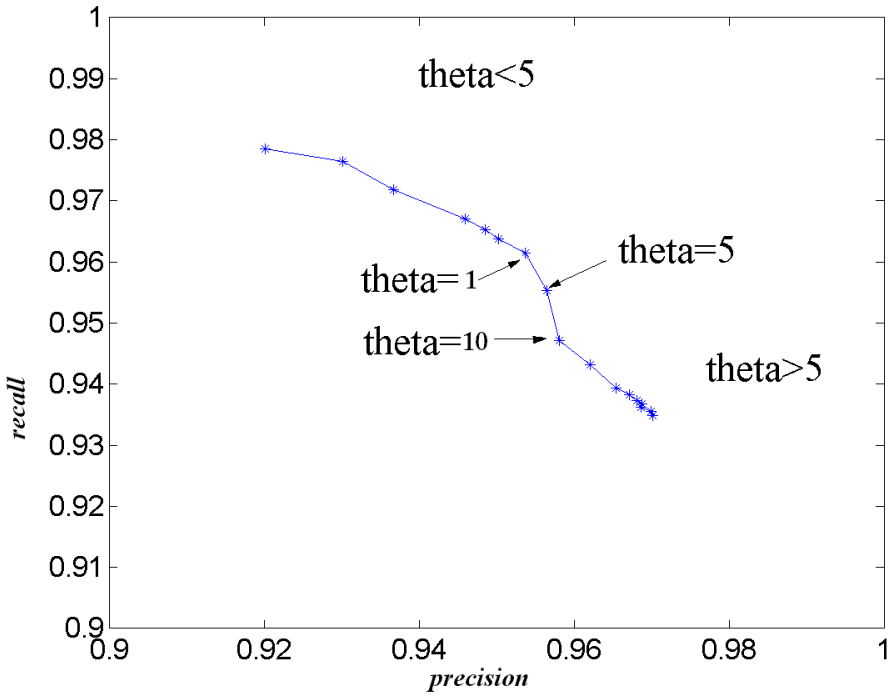


Figure 7.1: Precision and recall change as θ changes

We test the data sequence dependence model on *CoNLL* − 2000 shared task data. The $x - axis$ represents the values of *precision* while the $y - axis$ represents the values of *recall*. The range of θ is from 0.01 to 160.

a hidden Markov model (*HMM*) discussed in Section 4.2.1. We apply Viterbi algorithm discussed in Section 4.2.4 for computing $p(s_1, \ldots, s_N, c_1, \ldots, c_N) = \prod_{i=1}^{N} p(s_i|c_i)p(c_i|c_{i-1})$. By comparing the $f - measure$, we found that *CSDM* is 0.14% better that the *HMM*.

Table 7.4: Testing *CIM*, *DSDM*, *CSDM*, and *HMM* for identifying NP chunks on the *CoNLL* − 2000 data

	CIM	*DSDM*	*CSDM*	*HMM*1[1]
precision %	93.11	95.15	92.99	92.83
recall %	94.27	96.05	94.11	93.99
$f - measure$ %	93.69	95.60	93.55	93.41

1.*Viterbi's algorithm is used to compute* $p(s|c) = \prod_{i=1}^{N} p(s_i|c_i)p(c_i|c_{i-1})$

Comparisons

Table 7.5 shows the best performances of the related methods on the CoNLL-2000 shared task data. The $f-measures$ are ordered in descending order. *CIM* is the context independence model, *DSDM* is the data sequence dependence model, *CSDM* is the class sequence dependence model, and *HMM*1 is the hidden Markov model. Other methods can be found in papers [1] [17] [45] [32] [46] [6]. Although several of these papers report their results for identifying not only noun phrases but also prepositional phrases and verb phrases, we argue that the results are dominated by finding noun phrases. Other phrases can be determined from noun phrases by simply following the syntactic rules. For example, we can determine the prepositional phrases by the POS tag *IN* being in front of noun phrases. We can determine the verb phrases from not noun phrases by finding a sequence of POS tags related verbs, such as *VBZ*, *VBN*, *VBG*, ... Moreover, our testing results on *HMM*, *CIM*, and *CSDM* experiments support our argument. Therefore, comparing with the best performances of the related methods reported from theses papers, we found that *RBL* [46] achieves the worst $f-measure$ performance. The $f-measure$ performances of *CIM* and *CSDM* are better than *HMM*. Moreover, the *DSDM* achieves the best $f-measure$ performance.

Table 7.5: Comparisons of different methods on the $CoNLL-2000$ data

M	*RBL* [46]	*HMM*1	*HMM* [1]	*CSDM*	*CIM*	*MEMM* [17]	*PL* [45]	*CRF* [17]	*SVM* [32]	*DSDM*
f %	91.54	93.41	93.52	93.55	93.69	93.70	93.74	94.38	94.45	95.74

- M: method
- f: $f-measure$
- *RBL*: role based learning [46]
- *HMM*1: hidden Markov model
- *HMM*: hidden Markov model [1]
- *CSDM*: class sequence dependence model
- *CIM*: context dependence model
- *MEMM*: maximum entropy Markov model [17]
- *PL*: perceptron learning [45]
- *CRF*: conditions random fields [17]
- *SVM*: support vector machine [32]
- *DSDM*: data sequence dependence model

Testing Three Models on *WSJ* Data

We further tested three models on WSJ data from Penn Treebank. The main reason for using this data set was that we wanted to see whether the performance of our models can be improved when they were built on more data. The files *W*0200 – *W*0999 and *W*1100 – *W*2099 were used in this test. The symbol type in each symbol sequence was still *POS* + *lexicon* of a word. The training set contained 800 files: *W*0200 – *W*0999. The training set was seven times larger than the CoNLL-2000 shared task training data set. The training instance distributions are shown in Table 7.6. The

Table 7.6: The training instance distributions on *WSJ* data

# of possible symbols	# of possible categories	Total amount space taken for the probability table for the seen events
25930	3	4.6×10^8

testing set contained 100 files. For example, files *W*1100 – *W*1199 was one of the testing files. From this arrangement, we obtained ten estimates. Then, an average output was obtained from these ten estimates. The evaluating metrics were *precision*, *recall*, and $f - measure$. For example, for the data sequence dependence model (column 2), the *precision* showed the ratio of the correctly assigned NP chunks over the total number of true NP chunks was 97.7%. The *recall* showed that the correctly assigned NP chunks over the total number of assigned NP chunks was 98.5%, and the $f - measure$ showed that the average of the *precision* and *recall* was 98.1%. Table 7.7 shows the results. Appendix C shows the accuracy curves. By comparing with the results obtained from *CoNLL* – 2000 data, we noticed that as the training set became larger, our models achieved the better performance.

Monte Carl Testing of Significant Difference

From the experiment on *WSJ* data, for each model, we have obtained a set of $f-measure$s. Therefore, we have three sets of $f - measure$s for *CIM*, *DSDM*, and *CSDM*. In this experiment, we use Monte

Table 7.7: Testing *CIM*, *DSDM*, *CSDM* for identifying NP chunks on *WSJ* data

	CIM	*DSDM*	*CSDM*
precision %	95.6	97.7	95.2
recall %	96.5	98.5	95.8
$f - measure$ %	96.1	98.1	95.5

Carl testing to estimate the statistical significance of the difference between any pair of average $f - measure$s.

Let $f_x = \{f_x^1, \ldots, f_x^{10}\}$ and $f_y = \{f_y^1, \ldots, f_y^{10}\}$ be two sets of $f - measure$s obtained from the model x and the model y. Let $mean(f_x)$ be the average of $\{f_x^1, \ldots, f_x^{10}\}$ and $mean(f_y)$ be the average of $\{f_y^1, \ldots, f_y^{10}\}$. Let d_0 be the difference of $mean(f_x)$ and $mean(f_y)$:

$$d_0 = \frac{1}{10}\sum_{i=1}^{10} f_x^i - \frac{1}{10}\sum_{j=1}^{10} f_y^j$$

Let H_0 be the hypothesis that the mean of (f_x) equals to the mean of (f_y). Let H_1 be the alternative hypothesis that the mean of (f_x) is bigger than the mean of (f_y).

Let v be a vector, s.t.

$$v = (f_x^1, \ldots, f_x^{10}, f_y^1, \ldots, f_y^{10})$$

Let $\{I_1, \ldots, I_K\}$ be a set of K permutations obtained by randomly shuffling the components of the vector v K times. For each permutation $I_k = (f_{x,k}^1, \ldots, f_{x,k}^{10}, f_{y,k}^1, \ldots, f_{y,k}^{10})$, the distance d_k is defined as;

$$d_k = \frac{1}{10}\sum_{i=1}^{10} f_{x,k}^i - \frac{1}{10}\sum_{j=1}^{10} f_{y,k}^j$$

Therefore, the $p - value$ associated with the null hypothesis H_0 against the alternative hypothesis H_1 can be estimated as:

$$p - value \quad = \quad \frac{\#\{k|d_k > d_0\} + \frac{1}{2} * \#\{k|d_k = d_0\}}{K + 1}$$

Figure 7.2 shows the relations of $p-value$, d_0, and d_k on the experiment of *DSDM* vs. *CIM*. The

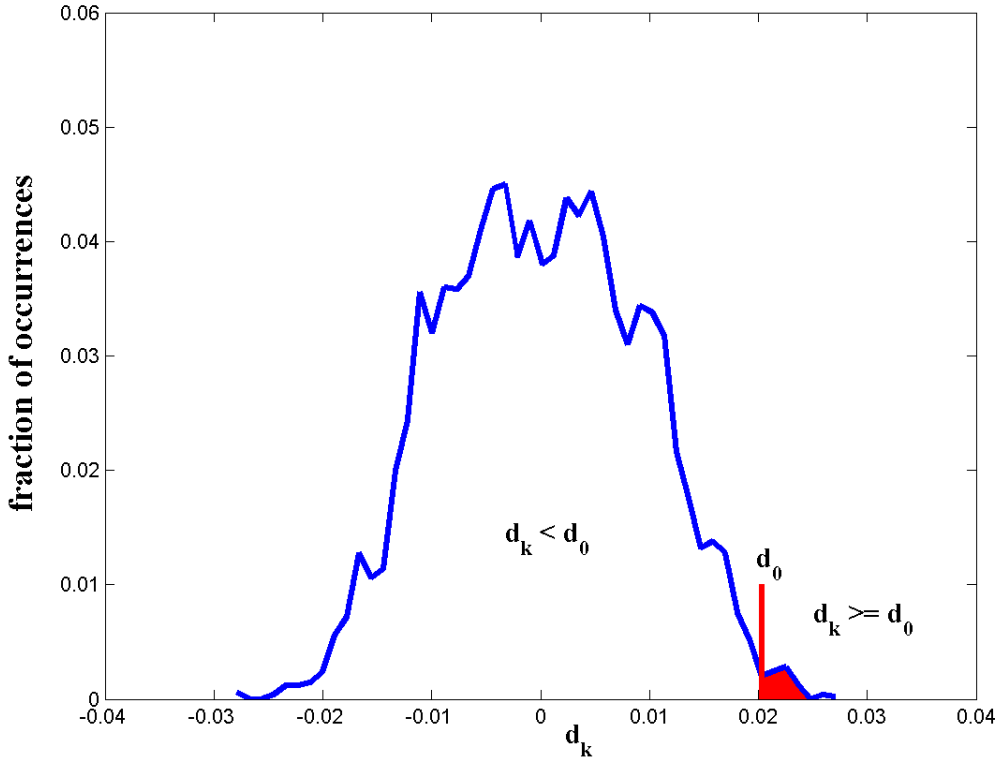


Figure 7.2: $p-value$ on *DSDM* vs. *CIM*

$p-value = \frac{\#\{k|d_k>d_0\}+\frac{1}{2}*\#\{k|d_k=d_0\}}{K+1}$. It is represented by the red area, where K is the # of permutations.

$x-axis$ represents d_k while the $y-axis$ represents the number of occurrences of d_k. The $p-value$ is represented by the red area.

In the experiments, we set the significance level $\alpha = 0.05$, the probability that we reject the null hypothesis when the null hypothesis is true. If $p-value < \alpha$, we reject the null hypothesis H_0.

Each time, we randomly shuffle the vector v to get a permutation I_k. We compute the mean of $f_{x,k}$ by averaging the first ten elements in the shuffled v and the mean of $f_{y,k}$ by averaging the last ten elements in the shuffled v. Further, we compute the difference d_i between these two means. We count the number of occurrences of d_k that are greater than and equal to d_0. We run this procedure for five thousand times. Figures 7.3, 7.4, and 7.5 show the results. In these Figures, each graph has three subgraphs. In each subgraph, the $x-axis$ indicates d_k and the $y-axis$ indicates the fraction of

number of occurrences of d_k. The upper left hand side subgraph represents the distribution of d_k less than d_0 while the upper right hand side subgraph represents the distribution of d_k great than and equal to d_0. The subgraph on the bottom represents the distribution of d_k.

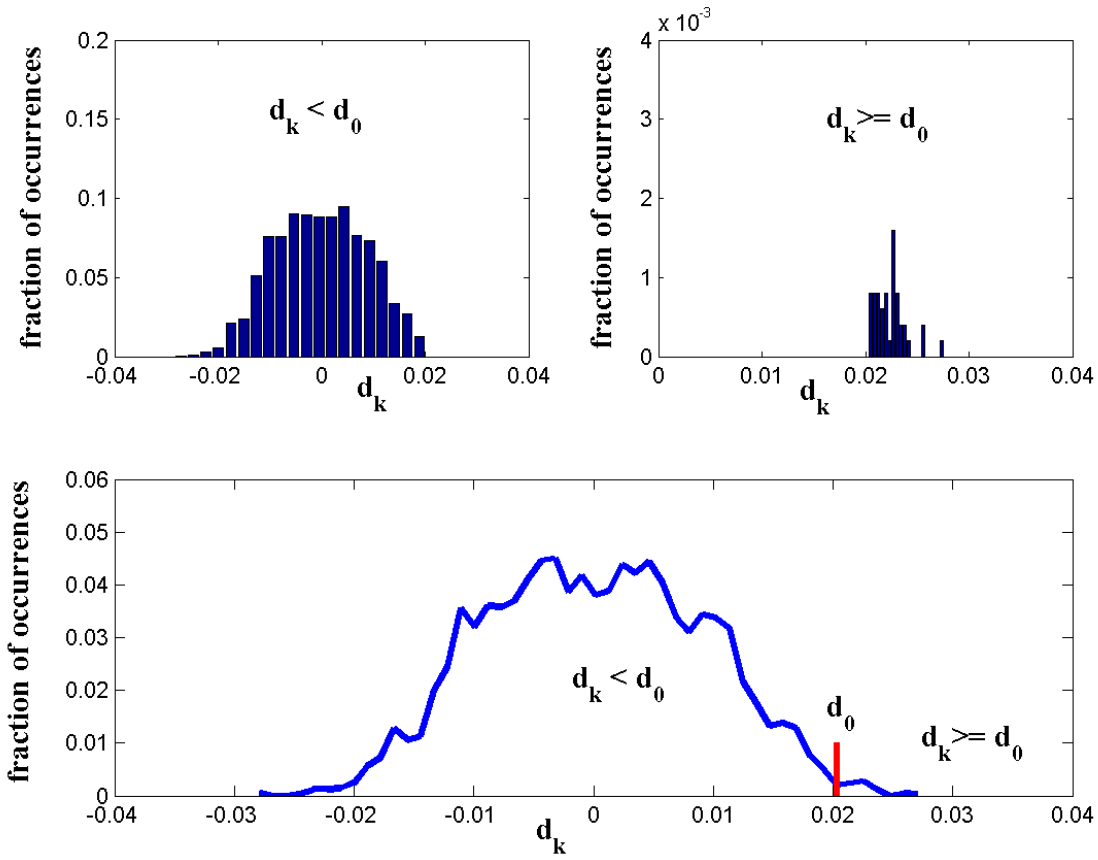


Figure 7.3: Monte Carl testing *DSDM* vs. *CIM*. The $p - value$ equals to 0.0074.

Figure 7.3 shows the result of testing H_0 against H_1 where the null hypothesis H_0 is that the mean of $f - measure$ for method *DSDM* is the same as the mean of $f - measure$ for method *CIM*. The alternative hypothesis H_1 is that the mean of $f - measure$ for method *DSDM* is larger than the mean of $f - measure$ for method *CIM*.

Figure 7.4 shows the result of testing H_0 against H_1 where the null hypothesis H_0 is that the mean of $f - measure$ for method *DSDM* is the same as the mean of $f - measure$ for method *CSDM*. The alternative hypothesis H_1 is that the mean of $f - measure$ for method *DSDM* is larger than the mean of $f - measure$ for method *CSDM*.

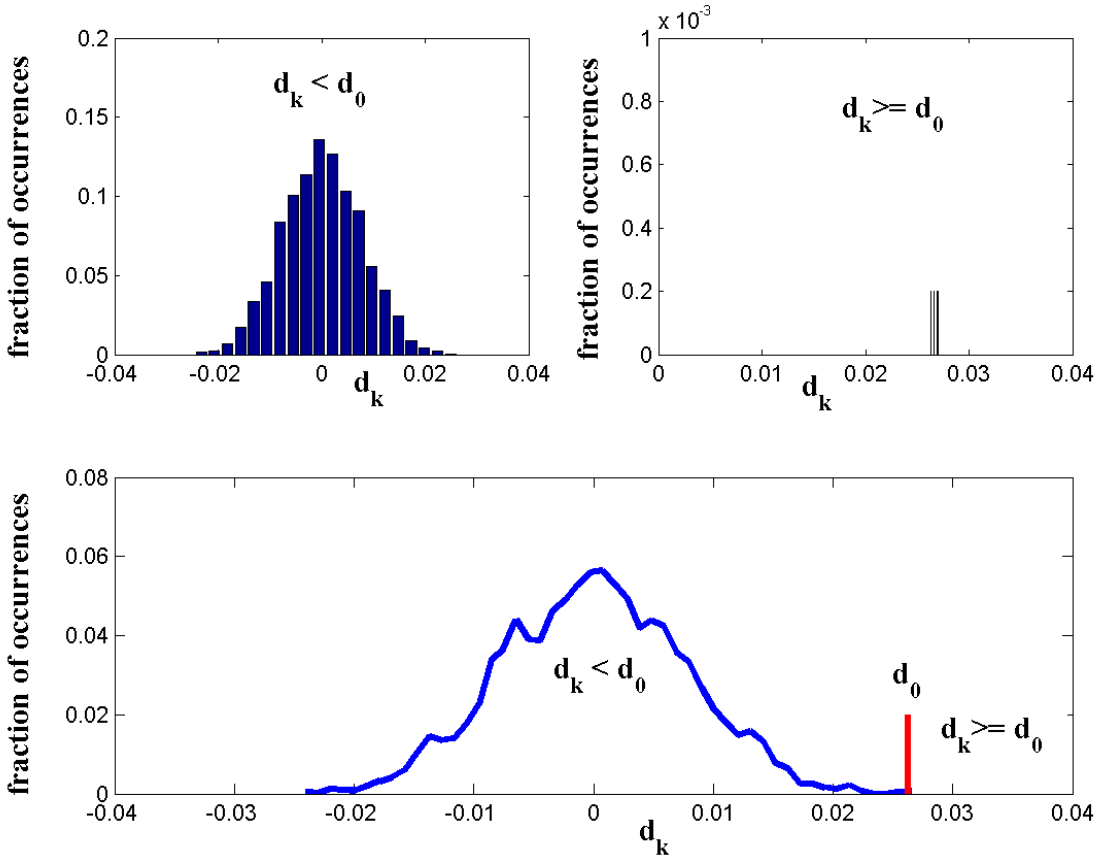


Figure 7.4: Monte Carl testing on *DSDM* vs. *CSDM*. The $p - value$ equals to 0.0006.

Figure 7.5 shows the result of testing H_0 against H_1 where the null hypothesis H_0 is that the mean of $f - measure$ for method *CIM* is the same as the mean of $f - measure$ for method *CSDM*. The alternative hypothesis H_1 is that the mean of $f - measure$ for method *CIM* is larger than the mean of $f - measure$ for method *CSDM*.

As stated previously, we have set $\alpha = 0.05$. For the test of $CIM = DSDM$ against $CIM < DSDM$, a $p - value$ of 0.0074 is obtained. The $p - value$ is less than α. Therefore, we reject the null hypothesis that the mean of $f - measure$ for method *DSDM* is the same as the mean of $f - measure$ for method *CIM*. We say that the mean of $f - measure$ for method *DSDM* is larger than the mean of $f - measure$ for method *CIM* at the significance level of 0.05. For the test of $CSDM = DSDM$ against $CSDM < DSDM$, a $p - value$ of 0.0006 is obtained. The $p - value$ is less than α. Therefore, we reject the null hypothesis that the mean of $f - measure$ for method *DSDM* is the same as the

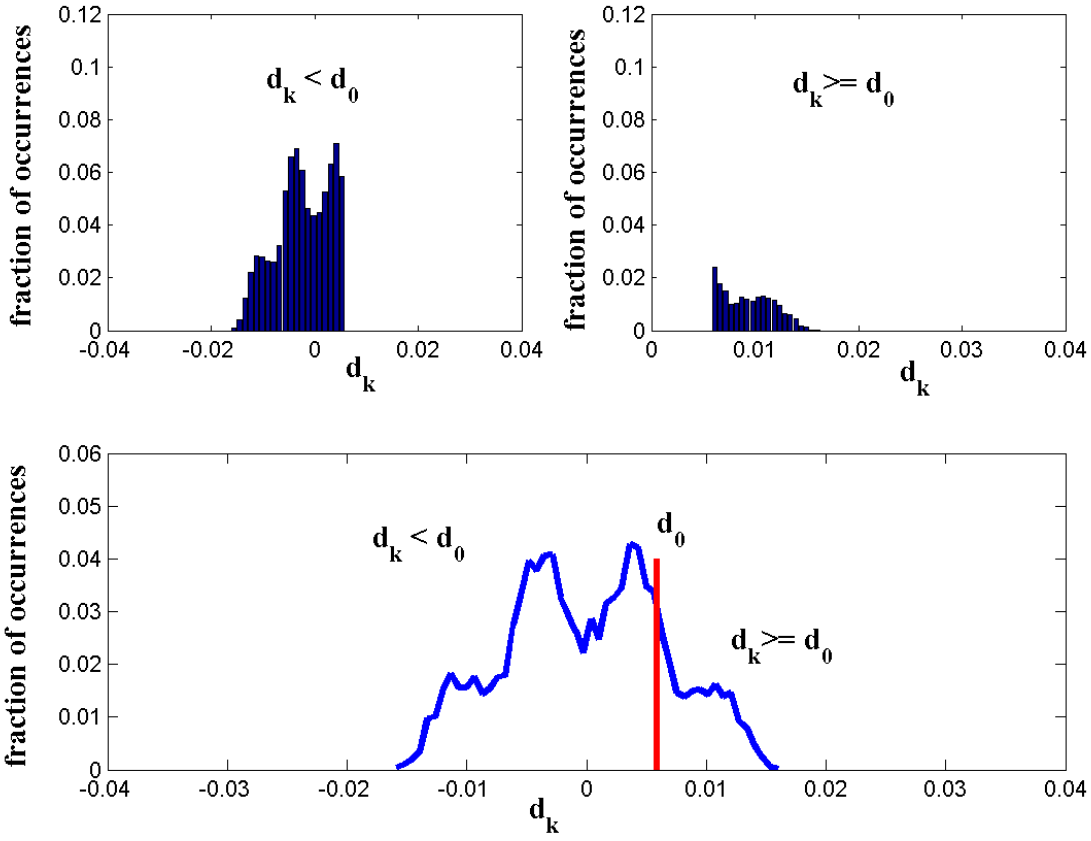


Figure 7.5: Monte Carl testing on *CIM* vs. *CSDM*. The $p - value$ equals to 0.1966.

mean of $f - measure$ for method *CSDM*. We say that the mean of $f - measure$ for method *DSDM* is larger than the mean of $f - measure$ for method *CSDM* at the significance level of 0.05. For the test of $CIM = CSDM$ against $CIM < CSDM$, a $p - value$ of 0.1966 is obtained. So, in this case, we can not reject the null hypothesis that the mean of $f - measure$ for method *CIM* is the same as the mean of $f - measure$ for method *CSDM*. We say that the difference in performance of *CIM* and *CSDM* is not statistically significant at the significance level of 0.05.

7.4 Experiments of Word Sense Disambiguation

7.4.1 Developing an Algorithm

Let $S =< s_1, .., s_t, .., s_N >$ be a sequence of symbols associated with a sentence. Each symbol is a pair consisting of a word α_n and its part of speech tag β_n, such that, $s_n =< \alpha_n, \beta_n >$. Let $s_t \in S$ be a given ambiguous symbol in which α_t needs to be disambiguated. Let $C = \{C_m | m = 1, ..., M\}$ be a set of predefined senses of the ambiguous word α_t.

Defining Contexts

The Context of an ambiguous symbol s_t is a $k - tuple$, represented by T_t. Each element in T_t is a symbol in S, $T_t = (t_1, ..., t_K)$, $t_k \in S$, and $K \leq N$.

Identifying the Sense of a Word

- Find the context T_t for s_t.
- Find a sequence of classes $< c_1^*, \ldots, c_K^* >$ for $T_t = (t_1, ..., t_K)$, s.t.

$$< c_1^*, \ldots, c_K^* >= \underset{c_1,...,c_K}{argmax}\, p(c_1, \ldots, c_K | t_1, \ldots, t_K)$$

- Assign C_j to s_t if and only if

$$\#\{k \mid c_k = C_j\} \geq \#\{k \mid c_k = C_m\},\ m = 1, \ldots, M$$

7.4.2 Complexities

Time Complexity

For a sequence of N input symbols and a set of M classes (the number of senses of a polysemous word), we need to have $O(N)$ time to find the contexts for a polysemous word. From Section 3.4.5,

to find an optimal class assignment sequence for the contexts, we need to have $O(NM)$ time. Moreover, to assign a class to the polysemous word, we need to have $O(N)$ time. Therefore, the time complexity of the algorithm is

$$T_c = O(MN) + O(N) + O(N) = O(NM).$$

Memory Complexity

To find the context for a polysemous word, we need to have $O(N)$ memory space. From Section 3.4.5, to find an optimal class assignment sequence for the contexts, we need to have $O(M)$ memory space. Moreover, to assign a class to the polysemous word, we need to have $O(N)$ memory space. Furthermore, for obtaining a probability table, we need to have $O(N^2M)$ memory space. Therefore, the memory complexity of the algorithm is

$$M_c = O(M) + O(N) + O(N) = O(M) + O(N) + O(N^2M) = O(N^2M).$$

7.4.3 Experiments

We test the data sequence dependence model on four data sets [47] [48]. They are the polysemous noun *line* data, the polysemous noun *interest* data, the polysemous adjective *hard* data, and the polysemous verb *serve* data. These data sets can also be found in *SENSVAL 2* [49]. Because the number of instances of each sense for these polysemous words is larger, it provides us more opportunities to check our methods. Moreover, other researchers have already published their results on these data sets so that it is easier for us to do the comparisons. For these data sets, the details of senses' descriptions and instances' distributions can be found in the original papers [47] [48]. Here, we just list the number of instances. The left hand side table of the Table 7.8 shows the number of instances of each sense of the polysemous words in the original sets.

In our case, for each polysemous word, we reduce the number of instances for senses which have the larger number in order to make all number of instances for each class to be equal. For example, sense 1 of the word *line* has a large number of instances: 2218. However, the number of instances for sense 2 to sense 6 are from 429 to 349. If the *accuracy* is obtained based on this data distribution

according to Equation (7.4), it will be more affected by sense 1 rather than others. From this perspective, we need to reduce the number of instances for sense 1 to 460. In Section *Testing on Line Use Full Data*, we will use original data. We solve this problem by setting a equal prior probability for each class.

We show our refined data sets in the right handed side table of the Table 7.8. Note, for the polysemous word *interest*, the number of instances for the sense 4 to the sense 6 are too small to compare with other senses. Therefore, we exclude the three senses in this experiment.

Table 7.8: Instance distructions of words *line*, *interest*, *hard*, and *serve*.

sense	*line*	*serve*	*hard*	*interest*
1	2218	1841	3345	1272
2	429	1272	502	500
3	404	853	376	361
4	376	439		178
5	373			66
6	349			11

sense	*line*	*serve*	*hard*	*interest*
1	460	450	400	400
2	429	450	400	400
3	404	450	376	361
4	376	439		
5	373			
6	349			

After refining our data sets, we randomly select $\frac{9}{10}$ of the instances as the training set and the remaining $\frac{1}{10}$ of the instances as the testing set. In each following experiment, we have created total 30 different training sets and 30 different testing sets.

For each sentence in which the polysemous word is located, we construct a word sequence by carrying only four words to the left and four words to the right of the polysemous words [1]. If there are no such words existing, we build the sequence by randomly selecting any word from the sentence. For example, if a sentence ends up with the polysemous word, that is, the four right words are missing. We randomly select a word from the sentence as the first right word respectively. Moreover, a symbol sequence is constructed in which each symbol is *POS* + *lexicon* of a word. The test metric we have used is *accuracy*.

Testing the Polysemous Noun *line*

For this experiment, we have two kinds of tests. let $C = \{C_1, C_2, C_3, C_4, C_5, C_6\}$= {*project* , *phone*, *text*, *division*, *cord*, *formation*}. By the procedure described in Section 7.4.3, We obtain 30 accuracies on the different training sets and the testing sets. The training instances of *line* is shown in Table 7.9.

[1] We have tried a set of context windows, this is the best choice.

In this training set, the probabilities for an unseen events, such as (s_{i-1}, s_i, c_i), (s_{i+1}, s_i, c_i) and (s_i, c_i) are not stored. They are computed based on Equation (6.5) discussed in Section 6.3.1. The results

Table 7.9: The training instance distribution on the *line* data

# of possible symbols	# of possible categories	Amount of space taken for the seen events	Events
791	6	863440	(s_{i-1}, s_i, c_i)
		863440	(s_{i+1}, s_i, c_i)
		4224	(s_i, c_i)
		6	(c_i)

are shown in Figure 7.6. The $x-axis$ represents an accuracy while the $y-axis$ represents the fraction of occurrences. By averaging these accuracies, we obtained the overall accuracy of 81.12% with a

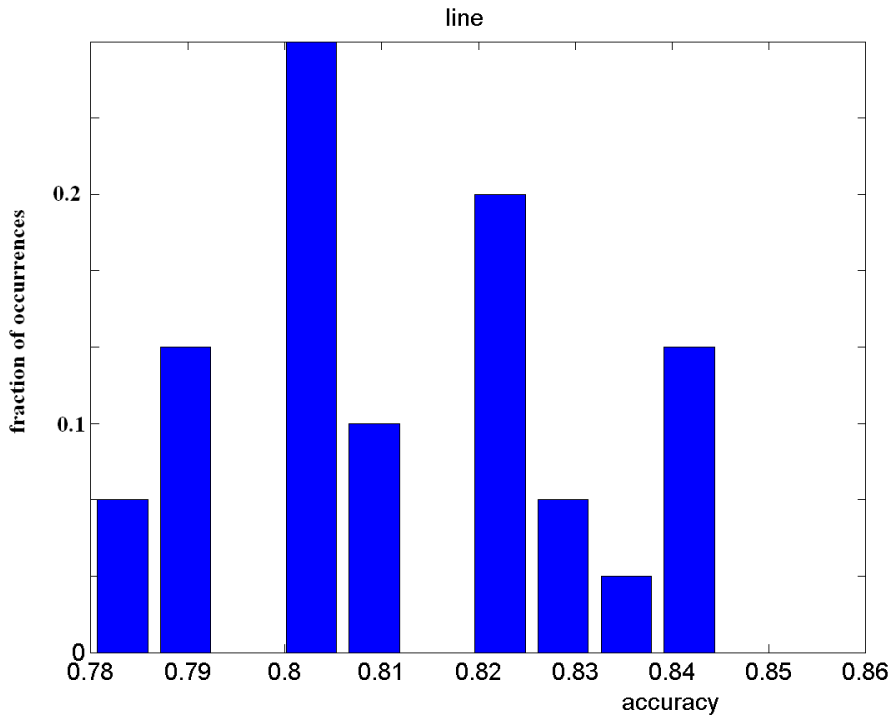


Figure 7.6: The testing results on the polysemous word *line*.
The $x-axis$ represents an accuracy while the $y-axis$ represents the fraction of occurrences. The average accuracy is 81.12% with a standard deviation of 1.89%.

standard deviation of 1.89%.

Table 7.10 shows one of the contingency tables. From the table, we see that of the 241 testing

instances, 196 instances are correctly identified. Among the 46 test instances of C_1, 43 instances are correctly identified. Among the 54 instances assigned to C_1, 43 instances are correctly assigned. Among the 43 test instances of C_2, 39 instances are correctly identified. Among the 43 instances assigned to C_2, 39 instances are correctly assigned. Among the 41 test instances of C_3, 25 instances are correctly identified. Among the 30 instances assigned to C_3, 25 instances are correctly assigned. Among the 38 test instances of C_4, 36 instances are correctly identified. Among the 45 instances assigned to C_4, 36 instances are correctly assigned. Among the 38 test instances of C_5, 26 instances are correctly identified. Among the 39 instances assigned to C_5, 26 instances are correctly assigned. Among the 35 test instances of C_6, 27 instances are correctly identified. Among the 30 instances assigned to C_6, 27 instances are correctly assigned. By the table on the right side of Table 7.10, C_1, C_2, and C_4 have a better *precision* compared to C_3, C_5, and C_6. Moreover, C_2 and C_6 have a better *recall* compared to others. Further, C_5 has the worst *recall* and C_3 has the worst *precision*. Therefor, instances of C_1, C_2, and C_4 have more chances to be assigned correctly while the most assigned instances are instances of C_2 and C_6. Moreover, instances of C_3 have more chances to be assigned wrongly compared with the instances of other classes.

Table 7.10: A contingency table for the six sense word *line*
Computing $p_T(C)$, $pA|T(C|C)$, and $p_{T|A}(C|C)$

	C_1	C_2	C_3	C_4	C_5	C_6	*accuracy*
C_1	43	0	1	1	1	0	
C_2	0	39	0	1	2	1	
C_3	4	3	25	1	6	2	
C_4	0	0	1	36	1	0	
C_5	5	1	2	4	26	0	
C_6	2	0	1	2	3	27	
							0.8133

	$p_T(C)$	$p_{A\|T}(C\|C)$	$p_{T\|A}(C\|C)$
C_1	0.1909	0.9348	0.7963
C_2	0.1784	0.9070	0.9070
C_3	0.1701	0.6098	0.8333
C_4	0.1577	0.9474	0.8000
C_5	0.1577	0.6842	0.6667
C_6	0.1452	0.7717	0.9000

Testing on *line* with Different Size of Classes

In the second experiment on the *line* data, we test our method on the collection of sets of classes with the size of two, three, four, and five. For each set with the different size, we include all combinations from the original set of the size *six*. For example, the number of combinations of two senses among six senses was $\binom{6}{2} = \frac{6!}{4!2!} = 15$, the number of combinations of three senses among

six senses was $\binom{6}{3} = \frac{6!}{3!3!} = 20$, the number of combinations of four senses among six senses was $\binom{6}{4} = \frac{6!}{2!4!} = 15$, and the number of combinations of five senses among six senses was $\binom{6}{5} = \frac{6!}{1!5!} = 6$. Figure 7.7 shows the results of all the combinations.

In each test, only relevant data samples are included. For example, in the case of testing classes C_1 and C_2, the instances of C_3 to C_6 are excluded. For each test, we obtain an average *accuracy* by averaging all accuracies of all combinations. From Figure 7.7, the average *accuracy* for two senses is 90.24% with a standard deviation of 3.96%. The best *accuracy* is 98.81% obtained from the combination C_1C_4 while the worst *accuracy* is 83.53% obtained from the combination C_3C_5. which verifies the result in Table 7.10. The average *accuracy* for three senses is 85.25% with a standard deviation of 3.59%. The best *accuracy* is 91.97% obtained from the combination $C_1C_2C_4$ while the worst *accuracy* is 77.60% obtained from the combination $C_3C_5C_6$ which is the same as discussed from the previous test. The average *accuracy* for four senses is 82.26% with a standard deviation of 2.75%. The best *accuracy* is 87.65% obtained from the combination $C_1C_2C_4C_6$ while the worst *accuracy* is 77.91% obtained from the combination $C_1C_3C_4C_5$. The average *accuracy* for five senses is 81.22% with a standard deviation of 2.38%. The best *accuracy* is 84.00% for the combination $C_1C_2C_4C_5C_6$ while the worst *accuracy* is 78.83% for the combination $C_1C_2C_3C_5C_6$. From this experiment, we noticed that as the number of senses increases, the average *accuracy* decreases. However, the ratio is smaller. Figure 7.8 shows the result. The $x-axis$ represents the number of senses while the $y-axis$ represents the values of probability. The *accuracy* changes from a set of five senses to a set of six senses is 0.01.

Testing *line* Using Full Data

In this test, we use all the data of *line*. That is, we keep 2000 instances for the sense one and instances for other senses are unchanged. Figure 7.11 shows one of the contingency tables.

Based on Section 7.1.3, the $accuracy_1$ can be estimated as:

$$accuracy_1 = \frac{172 + 39 + 25 + 36 + 29 + 26}{396} = 0.8258$$

	Accuracy 6 senses	Accuracy 5 senses	Accuracy 4 sensens	Accuracy 3 senses	Accuracy 2 senses
1	$C_{1,...,6}$ 0.8112	$C_{1,...,5}$ 0.8107	$C_{1,...,4}$ 0.8451	$C_1C_2C_3$ 0.8567	C_1C_2 0.9438
2		$C_{1,...,4}C_6$ 0.8325	$C_{1,...,3}C_5$ 0.7857	$C_1C_2C_4$ 0.9197	C_1C_3 0.8736
3		$C_{1,...,3}C_5C_6$ 0.7883	$C_1C_2C_4C_5$ 0.8485	$C_1C_2C_5$ 0.8583	C_1C_4 0.9881
4		$C_1C_2C_{4,...,6}$ 0.8400	$C_1C_{3,...,5}$ 0.7791	$C_1C_2C_6$ 0.8790	C_1C_5 0.9286
5		$C_1C_{3,...,6}$ 0.7992	$C_{2,...,5}$ 0.8000	$C_1C_3C_4$ 0.8580	C_1C_6 0.8765
6		$C_{2,...,6}$ 0.7897	$C_{1,...,3}C_6$ 0.8121	$C_1C_3C_5$ 0.7760	C_2C_3 0.8690
7			$C_1C_2C_4C_6$ 0.8765	$C_1C_3C_6$ 0.8197	C_2C_4 0.9506
8			$C_1C_3C_4C_6$ 0.8346	$C_1C_4C_5$ 0.8689	C_2C_5 0.8765
9			$C_{2,...,4}C_6$ 0.8408	$C_1C_4C_6$ 0.8908	C_2C_6 0.8974
10			$C_1C_2C_5C_6$ 0.8259	$C_1C_5C_6$ 0.8587	C_3C_4 0.8987
11			$C_1C_3C_5C_6$ 0.7988	$C_2C_3C_4$ 0.8607	C_3C_5 0.8354
12			$C_2C_3C_5C_6$ 0.7989	$C_2C_3C_5$ 0.8033	C_3C_6 0.8684
13			$C_1C_{4,...,6}$ 0.8344	$C_2C_3C_6$ 0.8403	C_4C_5 0.9079
14			$C_2C_{4,...,6}$ 0.8494	$C_2C_4C_5$ 0.8739	C_4C_6 0.9315
15			$C_{3,...,6}$ 0.8096	$C_2C_4C_6$ 0.8966	C_5C_6 0.8904
16				$C_2C_5C_6$ 0.8362	
17				$C_3C_4C_5$ 0.8034	
18				$C_3C_4C_6$ 0.8684	
19				$C_3C_5C_6$ 0.8070	
20				$C_4C_5C_6$ 0.8739	
average	0.8112	0.8122	0.8226	0.8525	0.9024

Figure 7.7: The accuracies for the polysemous word *line*.

We test *DSDM* on the collection of sets of senses with the size of two, three, four, five, and six on the word *line*. The average *accuracy* for two senses is 90.24%, the average *accuracy* for three senses is 85.25%, the average *accuracy* for four senses is 82.26%, the average *accuracy* for five senses is 81.22%, and the average *accuracy* for six senses is 81.12%.

In this case, the number of instances for the sense one is almost four times larger than others. Under this situation, we want to see what the *accuracy* would be if the class prior probabilities were equal.

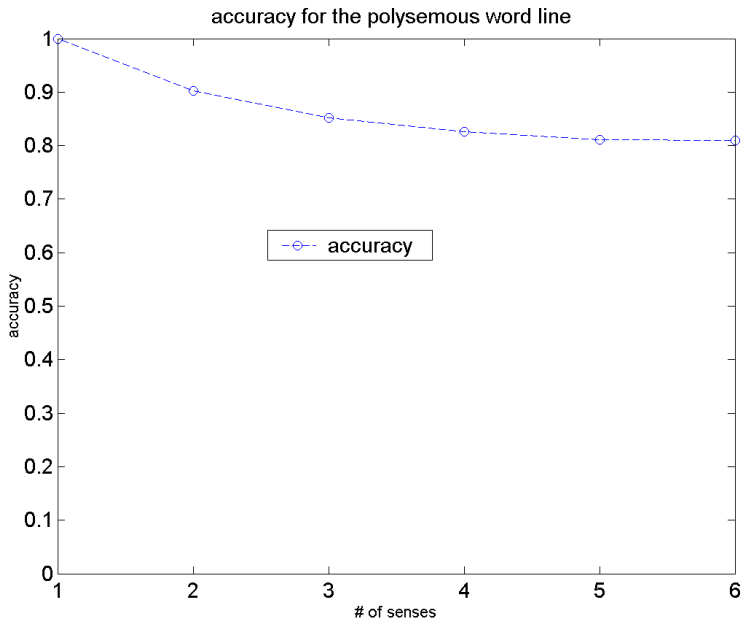


Figure 7.8: The accuracy curve for the polysemous word *line*.
The $x-axis$ represents the different class size while the $y-axis$ represents an accuracy.

Table 7.11: Another contingency table for the six sense word *line*
Computing $p_T(C)$, $p_A(C)$, $pA|T(C|C)$, and $p_{T|A}(C|C)$

	C_1	C_2	C_3	C_4	C_5	C_6
C_1	172	6	5	6	7	4
C_2	0	39	0	1	2	1
C_3	5	2	25	1	6	2
C_4	0	0	1	36	1	0
C_5	3	2	2	3	29	0
C_6	2	0	1	2	4	26

	$p_T(C)$	$p_A(C)$	$p_{A\|T}(C\|C)$	$p_{T\|A}(C\|C)$
C_1	0.5051	0.4596	0.8600	0.9451
C_2	0.1086	0.1237	0.9070	0.7959
C_3	0.1035	0.0859	0.6098	0.7353
C_4	0.0960	0.1237	0.9474	0.7347
C_5	0.0985	0.1237	0.7436	0.5918
C_6	0.0884	0.0833	0.7429	0.7879

Let C be a set of classes. Let $C_T \in C$ be a true class and $C_A \in C$ be an assigned class. Let $\mathcal{M}$ be a contingency matrix. Each entry $n_{ij} = \#\{(C_T, C_A)\}$. So, the probability of each entry $p_{ij} = \frac{n_{ij}}{N}$, where N is the # of test instances. Further, the probability of each row i (Marginal probability) is $p_{i.} = \sum_j p_{ij}$. Therefore, under equal prior probability the *accuracy* can be estimated by:

$$accuracy_2 = \frac{1}{|C|} \sum_{i=1}^{|C|} \frac{p_{ii}}{p_i} \tag{7.7}$$

For the *line* class, Table 7.12 is an contingency table $\mathcal{M}$: From the table, we compute the marginal

Table 7.12: A contingency matrix $\mathcal{M}$ for the word *line*

	C_1	C_2	C_3	C_4	C_5	C_6
C_1	n_{11}	n_{12}	n_{13}	n_{14}	n_{15}	n_{16}
C_2	n_{21}	n_{22}	n_{23}	n_{24}	n_{25}	n_{26}
C_3	n_{31}	n_{32}	n_{33}	n_{34}	n_{35}	n_{36}
C_4	n_{41}	n_{42}	n_{43}	n_{44}	n_{45}	n_{46}
C_5	n_{51}	n_{52}	n_{53}	n_{54}	n_{55}	n_{56}
C_6	n_{61}	n_{62}	n_{63}	n_{64}	n_{65}	n_{66}

	C_1	C_2	C_3	C_4	C_5	C_6
C_1	172	6	5	6	7	4
C_2	0	39	0	1	2	1
C_3	5	2	25	1	6	2
C_4	0	0	1	36	1	0
C_5	3	2	2	3	29	0
C_6	2	0	1	2	4	26

Table 7.13: A marginal probability table for the word *line*

	$p_{.1}$	$p_{.2}$	$p_{.3}$	$p_{.4}$	$p_{.5}$	$p_{.6}$	$p_{i.} = p_T(C)$
$p_{1.}$	0.4343	0.0152	0.0126	0.0152	0.0177	0.0101	0.5051
$p_{2.}$	0.000	0.0985	0.000	0.0025	0.0051	0.0025	0.1086
$p_{3.}$	0.0126	0.0051	0.0631	0.0025	0.0152	0.0051	0.1035
$p_{4.}$	0.000	0.000	0.0025	0.0909	0.0025	0.000	0.0960
$p_{5.}$	0.0076	0.0051	0.0051	0.0076	0.0732	0.000	0.0985
$p_{6.}$	0.0051	0.000	0.0025	0.0051	0.0101	0.0665	0.0884
$p.j = p_A(C)$	0.4596	0.1237	0.0859	0.1237	0.1237	0.0833	1.000

probabilities in Table 7.13. Therefore, based on the table, the $accuracy_2$ can be obtained as:

$$accuracy_2 = \frac{1}{6}(\frac{0.4343}{0.5051} + \frac{0.0985}{0.1086} + \frac{0.0631}{0.1035} + \frac{0.0909}{0.0960} + \frac{0.0732}{0.0985} + \frac{0.0657}{0.0884}) = 0.8018$$

By comparing $accuracy_2$ with the *accuracy* obtained in Section 7.4.3 *Testing on Polysemous Noun line*, we learned that for the given data set *line* with number of instances for each class being not balanced, described in the left table of Table 7.8, we estimate the *accuracy* for the data set by computing the sum of the probability of a class C given the class C ($p_{A|T}(C|C)$) multiply an equal prior probability of the class C ($p_T(C)$). In this case, the *accuracy* is 0.90% less than the *accuracy* obtained in Section *Testing on Polysemous Noun line*. This is because the instances in class C_1 are easier to identify which is shown in the contingency table 7.10 and the contingency table 7.11. Therefore, if the number of instances for class C_1 is difficult to identify, then the accuracy obtained from this method will be better than the accuracy obtained in Section *Testing on Polysemous Noun line Data*. On the other hand, if the number of instances for class C_1 is easier to identify, then the

accuracy obtained by this method will be less equal the accuracy obtained in Section *Testing on Polysemous Noun line*.

Testing the Polysemous Noun *interest*

For the *interest* data experiment, based on Table 7.8, three senses are omitted from the original data. We define $C = \{C_1, C_2, C_3\} = \{$*money paid for the use of the money*, *a share in the company*, *readiness for attention*$\}$. The training instances of *interest* is shown in Table 7.14. In this training set, the probabilities for an unseen events, such as (s_{i-1}, s_i, c_i), (s_{i+1}, s_i, c_i) and (s_i, c_i) are not stored. They are computed based on Equation (6.5) discussed in Section 6.3.1. By the procedure described

Table 7.14: The training instance distribution on the *interest* data

# of possible symbols	# of possible categories	Amount of space taken for the seen events	Events
415	3	124002	(s_{i-1}, s_i, c_i)
		124002	(s_{i+1}, s_i, c_i)
		1121	(s_i, c_i)
		3	(c_i)

in Section 7.4.3, we obtain 30 accuracies on the different training sets and the testing sets. By averaging these accuracies, the overall *accuracy* is 91.93% for this test. From the experiment, we learned that *interest* data is much easier to identify than *line* data. Figure 7.9 shows the results of the thirty tests. The $x-$*axis* represents an accuracy while the $y-$*axis* represents the fraction of occurrences. Table 7.15 is one of the contingency tables. From the table, we notice that of the 117 test instances, 108 instances are correctly identified. Among the 40 test instances of the class C_1, 39 instances are correctly identified. Among the 44 assigned instances to the class C_1, 39 instances are correct. Among the 40 test instances of the class C_2, 38 instances were correctly identified. Among the 41 assigned instances to the class C_2, 38 instances are correct. Among the 37 test instances of the class C_3, 31 instances are correctly identified. Among the 32 assigned instances to the class C_3, 31 instances are correct. By the table on the right side of Table 7.15, we see that instances of C_1 and C_2 have more chances to be assigned correctly while the most assigned instances are instances of C_3.

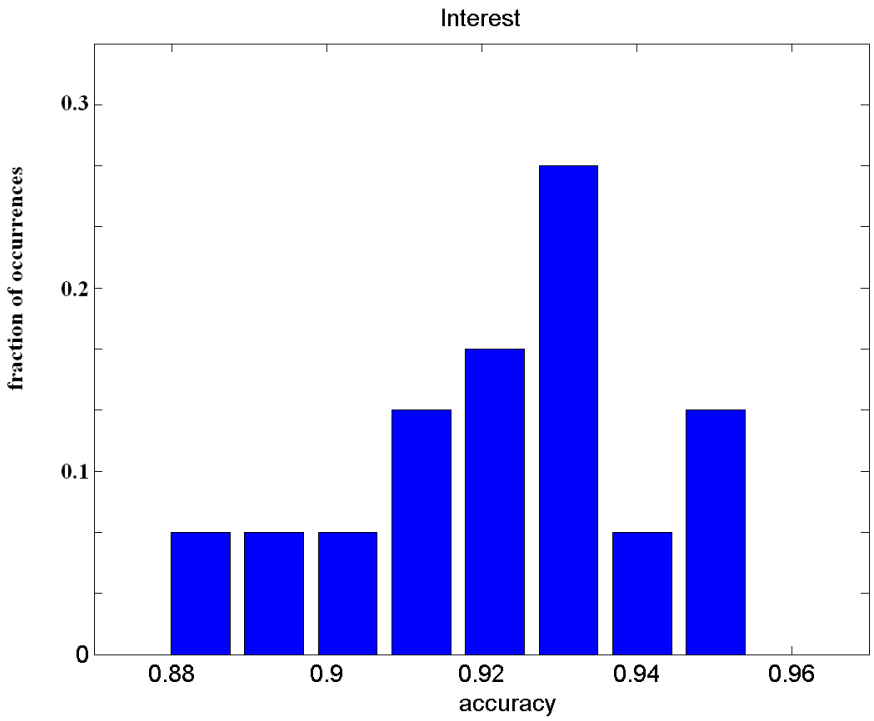


Figure 7.9: The testing results for the polysemous word *interest*.
The $x-axis$ represents an accuracy while the $y-axis$ represents the fraction of occurrences. The average accuracy is 91.93% with a standard deviation of 2.21%.

Testing the polysemous verb *serve*

For the *serve* data experiment, based on Table 7.8, there are four senses from the original data. We define $C = \{C_1, C_2, C_3, C_4\} = \{supply\ with\ food, hold\ a\ office, function\ as\ something, provide\ a\ service\}$. The training instances of *serve* is shown in Table 7.16. In this training set, the probabilities for an unseen events, such as (s_{i-1}, s_i, c_i), (s_{i+1}, s_i, c_i) and (s_i, c_i) are not stored. They are computed based on Equation (6.5) discussed in Section 6.3.1. By the procedure described in Section 7.4.3, we have 30 accuracies on the different training sets and the testing sets. By averaging these accuracies, we have an overall *accuracy* of 79.30% for this test. Figure 7.10 shows the results of the thirty tests. The $x-axis$ represents an accuracy while the $y-axis$ represents the fraction of occurrences. From the experiment, we learned that *serve* data is much more difficult to identify than *line* and *interest* data.

Table 7.17 is one of the contingency tables. From the table, we notice that of 179 test instances,

Table 7.15: A contingency table for the three sense word *interest*
Computing $p_T(C)$, $pA|T(C|C)$, and $p_{T|A}(C|C)$

	C_1	C_2	C_3	accuracy
C_1	39	1	0	
C_2	1	38	1	
C_3	4	2	31	
				0.9231

	$p_T(C)$	$p_{A\|T}(C\|C)$	$p_{T\|A}(C\|C)$
C_1	0.3419	0.9750	0.8864
C_2	0.3419	0.9500	0.9268
C_3	0.3162	0.8378	0.9688

Table 7.16: The training instance distribution on the *serve* data

# of possible symbols	# of possible categories	Amount of space taken for the seen events	Events
522	4	337880 337880 1900 4	(s_{i-1}, s_i, c_i) $(s_i i + 1, s_i, c_i)$ (s_i, c_i) (c_i)

145 instances are correctly identified. Among the 45 test instances of the class C_1, 37 instances are correctly identified. For all of the 43 instances that are assigned to C_1, 37 instances are correct. Among the 45 test instances of the class C_2, 35 instances are correctly identified. For all of the 41 instances that are assigned to C_2, 35 instances are correct. Among the 45 test instances of the class C_3, 41 instances are correctly identified. For all of the 56 instances that are assigned to C_3, 41 instances are correct. Among the 44 test instances of the class C_4, 32 instances are correctly identified. For all of the 39 instances that are assigned to C_4, 32 instances are correct. By the table on the right side of Table 7.17, we see that instances of C_3 have more chances to be assigned correctly while the most assigned instances are instances of C_1 and C_2.

Table 7.17: A contingence table for the four sense word *serve*
Computing $p_T(C)$, $pA|T(C|C)$, and $p_{T|A}(C|C)$

	C_1	C_2	C_3	C_4	accuracy
C_1	37	2	4	2	
C_2	3	35	3	4	
C_3	2	1	41	1	
C_4	1	3	8	32	
					0.8101

	$p_T(C)$	$p_{A\|T}(C\|C)$	$p_{T\|A}(C\|C)$
C_1	0.2541	0.8222	0.8605
C_2	0.2541	0.7778	0.8537
C_3	0.2541	0.9111	0.7321
C_4	0.2458	0.7273	0.8205

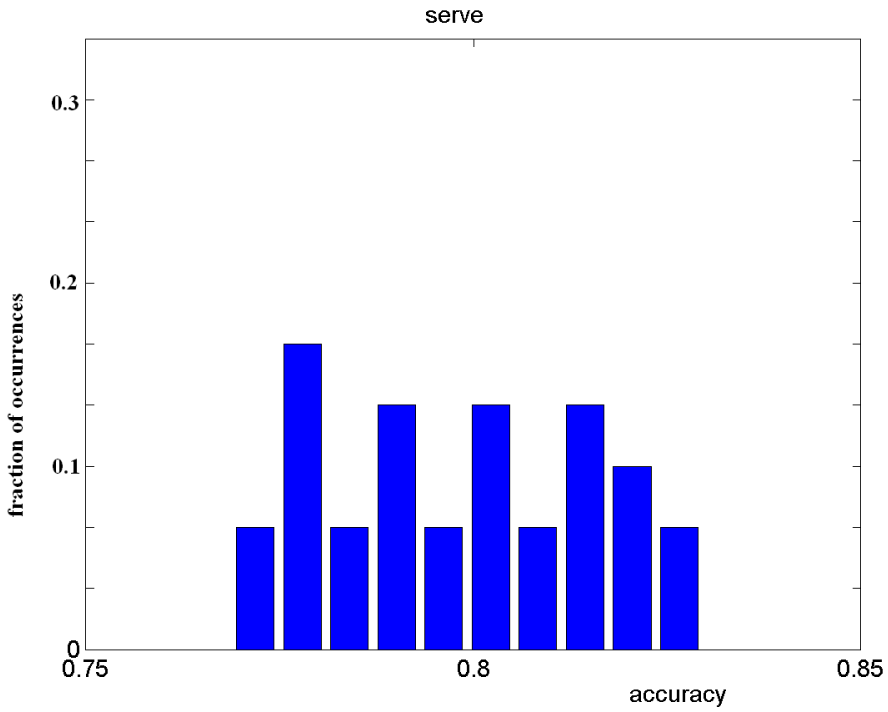


Figure 7.10: The testing results for the polysemous word *serve*.
The $x-axis$ represents an accuracy while the $y-axis$ represents the fraction of occurrences. The average accuracy is 79.30% with a standard deviation of 1.90%.

Testing the polysemous adjective *hard*

For the *hard* data experiment, based on Table 7.8, there are three senses from the original data. We define $C = \{C_1, C_2, C_3\} = \{$*difficult*, *not soft*, *physical not soft*$\}$. By the procedure described in Section 7.4.3, we have 30 accuracies on the different training sets and the testing sets. The training instances of *hard* is shown in Table 7.18. In this training set, the probabilities for an unseen events, such as (s_{i-1}, s_i, c_i), (s_{i+1}, s_i, c_i) and (s_i, c_i) are not stored. They are computed based on Equation (6.5) discussed in Section 6.3.1. By averaging these accuracies, we obtain an overall *accuracy* of 82.73% for this test.

Figure 7.11 shows the results of the thirty tests. The $x-axis$ represents an accuracy while the $y-axis$ represents the fraction of occurrences. Table 7.19 is one of the contingency tables. From the table, we see that of 118 test instances, 98 instances are correctly identified. Among the 40 testing instances of the class C_1, 38 instances are correctly identified. For all the 49 instances that

Table 7.18: The training instance distribution on the *hard* data

# of possible symbols	# of possible categories	Amount of space taken for the seen events	Events
397	3	132392	(s_{i-1}, s_i, c_i)
		132392	(s_{i+1}, s_i, c_i)
		1183	(s_i, c_i)
		3	(c_i)

are assigned to C_1, 38 instances are correct. Among the 40 testing instances of the class C_2, 35 instances are correctly identified. For all of the 42 instances that are assigned to C_2, 35 instances are correct. Among the 38 testing instances of the class C_3, 25 instances are correctly identified. For all of the 27 instances that are assigned to C_3, 25 instances are correct. By the table on the right side of Table 7.19, we see that instances of C_1 and C_2 have more chances to be assigned correctly while the most assigned instances are instances of C_3.

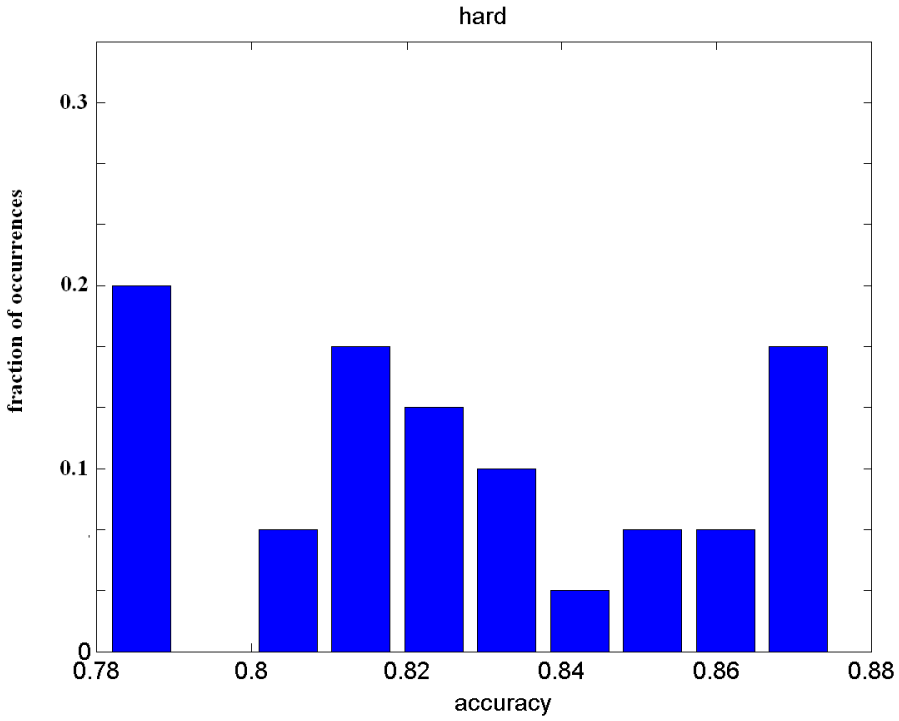


Figure 7.11: The testing results for the polysemous word *hard*.
The $x-axis$ represents an accuracy while the $y-axis$ represents the fraction of occurrences. The average accuracy is 82.73% with a standard deviation of 3.10%.

Table 7.19: A contingency table for the three sense word *hard*
Computing $p_T(C)$, $pA|T(C|C)$, and $p_{T|A}(C|C)$

	C_1	C_2	C_3	accuracy
C_1	38	1	1	
C_2	4	35	1	
C_3	7	6	25	
				0.8305

	$p_T(C)$	$p_{A\|T}(C\|C)$	$p_{T\|A}(C\|C)$
C_1	0.3390	0.9500	0.7755
C_2	0.3390	0.8750	0.8333
C_3	0.3220	0.6579	0.9259

Discussion

From the experiments, we see that whether a polysemous word is a noun, an adjective, or a verb, whether a polysemous word has three senses, four senses, or six senses, our method achieves an average accuracy of about 80%. Table 7.20 shows the results.

Misclassified instances are primarily generated by the ambiguity of the context words. For

Table 7.20: Testing *DSDM* on words *line, serve, hard, interest*

Data	*sense*	*accuracy* %	standard deviation %	*max* %	*min* %
line (noun)	6	81.16	1.92	84.50	78.00
	3	85.25	2.13	91.70	81.05
serve (verb)	4	79.80	1.90	82.92	76.88
hard (adjective)	3	82.88	3.10	87.03	78.11
interest (noun)	3	92.10	2.21	95.50	86.00

example in Table 7.20, comparing with the three sense noun *interest* and the three sense noun *line* obtained by selecting three senses at each time from six senses and examining all twenty combinations, we notice that the *accuracy* of the word *interest* is almost 9% higher than the accuracy for the word *line*. Moreover, by examining accuracies generated from each combination for the word *line* in Figure 7.7, we learn that combination ($C_1C_2C_4$) has the highest average accuracy: 91.7% while combination ($C_1C_3C_5$) has lowest average accuracy: 77.1%. The difference is almost 20%. By carefully checking these misclassified instances, we learn that if two senses are similar to each other, there are more chances that their contexts consist of the same words. As a consequence, the misclassification rate increases.

By examining the outputs of the two polysemous nouns *line* and *interest* on the same number of

senses, we find that our method achieves an average accuracy greater than 85%. However, the average *accuracy* of *interest* is much higher than the average *accuracy line*: 91.93% vs. 85.25%. This result suggests that the senses of *line* are more difficult to identify than *interest*. Features of some senses of *line* might not carry enough distinguishing information compared to *interest*. Moreover, from the second *line* experiment, we find that as the number senses increases, the average accuracy decreases. For example, the average accuracy for three senses is 85.25% while for six senses is 81.12%. However, from Figure 7.8, the rate of decrease gets smaller. Furthermore, whether the polysemous word is noun, or adjective, or verb, one common characteristic that we have noticed is among all senses, some senses are easier to identify while some senses are difficult to recognize. For example, for *line*, the *phone* sense is the easiest one while the *text* sense is the most difficult one. The *division*, *project*, and *formation* senses are relatively easier to identify than the *cord* sense; for *serve*, *readiness to give attention* is the most difficult to identify among other three senses. In conclusion, our method is effective for recognizing polysemous words from texts. It is more effective for recognizing nouns than adjectives and verbs and less effective on verbs.

Test *line, interest, hard, serve* on Maximum Entropy Model

We have tested polysemous words *line*, *interest*, *hard*, and *serve* on maximum entropy models [2] and [18] that we have discussed this model in Section 5.2.2 of chapter 5. In this experiment, we run the Maxent software package implemented in Python [5]. We have formed the context features $N = \pm 1, 2, 3, 4$ of words around the target word. The number of instances for each sense is shown in the left side table of Table 7.8. The maximum entropy estimation requires iterations. For each test, we use one hundred iterations to training a model. Results are shown in Table 7.21. We have seen that the best accuracy for each target word is achieved with the different size of context features. The best accuracy 79.40% for *line* is achieved with the context feature ±1 words, the best accuracy 93.01% for *interest* is achieved with the context feature ±1 words, the best accuracy for *hard* is achieved with the context feature ±3 words, and the best accuracy for *serve* is achieved with the context feature ±3 words.

By the column 6 of Table 7.21, we see that these results are dominated by the results of senses which have the more instances. For example, for the word *line*, the accuracy will be dominated by

Table 7.21: Testing the maximum entropy model for identifying word senses
$accuracy_1$ obtained for words *line*, *interest*, *hard*, and *serve*

	$accuracy_1$ % ± 1 words	$accuracy_1$ % ± 2 words	$accuracy_1$ % ± 3 words	$accuracy_1$ % ± 4 words	$accuracy_1$ % base_line
line	79.40	79.04	76.99	75.90	53.47
interest	93.44	92.32	90.93	90.72	59.63
hard	90.77	90.37	91.00	90.43	79.74
serve	77.05	82.31	84.02	83.79	41.43

C_1 (see the left table of Table 7.8). In order to change this, we compute the accuracy of each word based on Equation 7.7. Table 7.22 shows the results. Moreover, Table 7.23, 7.24, 7.25, and 7.26 are the contingency tables for *line*, *interest*, *hard*, and *serve*.

Table 7.22: Testing the maximum entropy model for identifying word senses
$accuracy_2$ obtained for words *line*, *interest*, *hard*, and *serve* where we set the prior class probability to be equal

	$accuracy_2$ % ± 1 words	$accuracy_2$ % ± 2 words	$accuracy_2$ % ± 3 words	$accuracy_2$ % ± 4 words
line	67.88	65.89	62.07	61.80
interest	91.26	89.69	86.47	86.91
hard	75.44	74.63	75.24	74.63
serve	69.29	75.03	77.37	77.45

Table 7.23: A contingency table for the six sense word *line*
Testing the maximum entropy model for identifying word senses

	C_1	C_2	C_3	C_4	C_5	C_6	accuracy
C_1	421	1	6	2	7	2	
C_2	20	58	1	0	3	2	
C_3	22	4	37	4	5	3	
C_4	8	4	0	64	0	1	
C_5	20	11	3	2	35	2	
C_6	30	3	3	0	5	41	
							0.7904

	p_T(C)	$p_{A\|T}$(C|C)	$p_{T\|A}$(C|C)
C_1	0.5289	0.9590	0.8081
C_2	0.1012	0.6905	0.7160
C_3	0.0904	0.4933	0.7400
C_4	0.0928	0.8312	0.8889
C_3	0.0880	0.4795	0.6364
C_4	0.0988	0.5000	0.8039

Table 7.24: A contingency table for the three sense word *interest*
Testing the maximum entropy model for identifying word senses

	C_1	C_2	C_3	accuracy
C_1	241	4	4	
C_2	6	95	9	
C_3	5	5	61	
				0.9232

	$p_T(C)$	$p_{A\|T}(C\|C)$	$p_{T\|A}(C\|C)$
C_1	0.5791	0.9679	0.9563
C_2	0.2558	0.8636	0.9135
C_3	0.1651	0.8592	0.8243

Table 7.25: A contingency table for the three sense word *hard*
Testing the maximum entropy model for identifying word senses

	C_1	C_2	C_3	accuracy
C_1	678	4	10	
C_2	26	65	4	
C_3	28	6	46	
				0.9037

	$p_T(C)$	$p_{A\|T}(C\|C)$	$p_{T\|A}(C\|C)$
C_1	0.7982	0.9798	0.9262
C_2	0.1096	0.6842	0.8667
C_3	0.0923	0.5750	0.7667

Table 7.26: A contingence table for the four sense word *serve*
Testing the maximum entropy model for identifying word senses

	C_1	C_2	C_3	C_4	$accuracy_1$
C_1	331	14	7	5	
C_2	17	223	16	3	
C_3	15	26	122	2	
C_4	31	10	9	45	
					0.8231

	$p_T(C)$	$p_{A\|T}(C\|C)$	$p_{T\|A}(C\|C)$
C_1	0.4057	0.9272	0.8401
C_2	0.2957	0.8610	0.8168
C_3	0.1884	0.7394	0.7922
C_4	0.1084	0.4737	0.8182

Comparisons

Our results are better than the results reported by other WSD researchers [50] and [51]. Our method achieves an average accuracy of 81.12% for identifying the six sense noun *line* using 2450 training context words, the method discussed by [51] achieves an average accuracy of 73% using 8900 training context words, and maximum entropy model achieves an average accuracy of 79.40% using 6638 training context words. Moreover, an experiment using the Latent Semantic Analysis method conducted by [50] achieves an average accuracy of 75% for identifying only three senses of *line*. The comparisons are shown in Table 7.27.

Table 7.27: Comarisons of different methods for identifying senses of the word *line*

	accuracy (3 senses)	*accuracy* (6 senses)	*#of contexts words per instance*	*base − line* (6 senses)
LSA	75%			
Bayesian	76%	71%	44.5 words	16.67%
Context Vector	73%	72%	44.5 words	16.67%
Neural Network	79%	76%	44.5 words	16.67%
Maximum Entropy Model		79.40%	2 words	53.47%
		67.88%	2 words	16.67%
DSDM	85.25%	81.12%	7 words	16.67%

7.5 Experiments of Semantic Role Labeling

7.5.1 Labeled Rooted Tree

A rooted tree T is a 3-tuple (V, E, r), where V is a finite set of vertices, $E \subseteq V \times V$ is a finite set of edges, and $r \in V$ is the root that all edges of T are directed away from it. The tree-order is the partial ordering on V for any $v, u \in V, u \leq v$ if and only if the unique path from the root r to v passes through u. In T, the root r is a unique minimal vertex and we say it has level 0. An edge in E is an ordered pair $(x, y) \in (V \times V)$ s.t. $x < y$ and there exists no $z \in V$ with $x < z < y$. In this case, x is a parent of y and y is a child of x. If two nodes[2] x, y have the same parent z, x and y are called siblings. Any node y on the unique path from r to x is called an ancestor of x. In this case, x is a descendant of y. The sub-tree rooted at node x is the tree induced by the descendants of x. A node with no children is an external node or a leaf. A node that is not a leaf node is an internal node. The largest depth of a node in T is the *height* of T. A labeled rooted tree is a 5-tuple (V, E, r, A, L). It is a rooted tree with additional two elements: a labeling alphabet A and a labeling function $L : V \rightarrow A$ that assigns labels to vertices.

[2]In a rooted tree, a vertex can be also called a node.

7.5.2 Finding a Path

Let $T = (V, E, r)$ be a rooted tree. For any $a, b \in V, a \neq b$, we say that $a - b$ is a (undirected) path, if we have a sequence

$$a = x_0, x_1, x_2, \ldots, x_{n-1}, x_n = b$$

$$\textit{where:} \quad (x_{i-1}, x_i) \in E, x_i \in V, i = 0, \ldots, n$$

of vertices from T staring from a and ending at b. Moreover, the path $a - b$ can also be written as:

$$a \rightarrow x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_{n-1} \rightarrow b$$

For example, Figure 7.12 shows a path: $VBZ \rightarrow VP \rightarrow ADJP - PRD \rightarrow VBN$, where $a = VBZ$ and $b = VBN$.

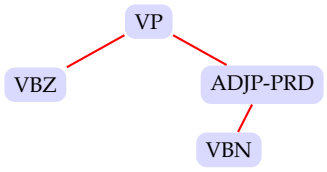


Figure 7.12: A path: $VBZ \rightarrow VP \rightarrow ADJP - PRD \rightarrow VBN$

In order to construct a path, first, we find the start node a based on predefined rules: the label of a is in $\{VB, VBZ, VBD, VBN, VBG, VBP\}$, a is not in any already formed path, and a has an only child that can not be an internal node. From a, we visit each unvisited adjacent node α to determine the second node x_1 in the path. Let $C = \{C_1, C_2\}$ be a set of two class. C_1 represents a node is in a path and C_2 represents a node is not in a path. For any adjacent α, if the joint probability of $p(a, \alpha, C_1, C_2)$ is greater than the joint probability $p(a, \alpha, C_1, C_1)$, this means that the α is not in the path and that we will throw it away. Otherwise, α is a candidate that needs to be considered further. Suppose αs: $\{\alpha_1, \alpha_2, \ldots, \alpha_K\}$ is the set of K possible candidates. We select α_k as x_1 if $p(a, \alpha_k, C_1, C_1)$ is the maximum. From x_1, we can find x_2 by selecting α_k that $p(a, x_1, \alpha_k, C_1, C_1, C_1)$ is maximum. Following this, we can select $x_3, x_4, \ldots,$ to x_n.

7.5.3 A Labeled Rooted Forest

A labeled rooted forest is a set of labeled rooted trees, s.t. $F = \{T_i | i = 1 \dots N\}$ where T_i is a labeled rooted tree.

7.5.4 Developing an Algorithm

Let $T = (V, E, r, A, L)$ be a labeled rooted tree associated with a sentence, where V is a set of vertices, E is a set of edges, $E \subseteq V \times V$, r is the root, A is an alphabet defined by [52], and L is a labeling function $L : V \rightarrow A$ that assigns labels to vertices. The parse tree of the sentence takes the form of T. Let π be a set of labels, s.t. $\pi = \{VB, VBZ, VBD, VBN, VBG, VBP\} \subseteq A$. Let $C = \{C_1, C_2\}$ be a set of classes, where C_1 represents that a current node is in the path ; C_2 represents that a current node is not in the path.

Let $x \in V$ be a node, s.t. $L(x) \in \pi = \{VB, VBZ, VBD, VBN, VBG, VBP\}$, x is not in any previously formed path, and x has an only child that can not be an internal node. For example, in Figure 7.13, we have a list of xs. Searching by applying BFS, we have $[VBD, VBZ, VBN, VBG, VBG]$.

- Search for finding an x, $b_1 = x$.
- Form a path $\mathcal{P}(x) = \tau_1, \rightarrow \dots, \rightarrow \tau_K$.

$$
\begin{aligned}
\tau =< \tau_1, \dots, \tau_K > &= \underset{b_1,\dots,b_K}{argmax}\, p(c_1, \dots, c_K, b_1, \dots, b_K) \\
&= \prod_{k=2}^{K} \underset{c_k=C_1,b_k}{max}\, p(b_{k-1}|c_k, b_k) p(b_{k+1}|c_k, b_k) p(b_k|c_k) p(c_k) \\
&= \prod_{k=2}^{K} \underset{c_k=C_1,b_k}{max}\, p(b_{k-1}|c_k, b_k) \cdot 1 \cdot p(b_k|c_k) p(c_k) \\
&= \prod_{k=2}^{K} \underset{c_k=C_1,b_k}{max}\, f(b_{k-1}, b_k, C_k)
\end{aligned}
$$

  Note, $b_k \in V$, $b_{k-1}b_k \in E$.

- Form a set of roots [3]. We call the set of roots as $R(x)$, where $R(x) = \{r_i | i = 1 \dots M\}$ formed based on $P(x)$.

[3] A semantic argument of x can be found from leaves of a root.

- For each τ_k, $k = 1, \ldots, K$, $R(x) \leftarrow R(x) \cup \{z\}$, if z satisfies
  - z is a sibling of τ_k or a sibling of an ascendent of τ_k,
  - z is not in $\tau =< \tau_1, \ldots, \tau_K >$,
  - $L(z) \notin \pi = \{VB, VBZ, VBD, VBN, VBG, VBP\}$ and $L(z)$ can not be punctuation marks,

- Find a rooted forest $F(x) = \{T_k | i \in \{1, \ldots, K\}\}$,
  - Each T_k is induced from the root r_k by all its codependents.
  - For each $T_k \in F(x)$, the leaves $\{l_k^1, \ldots, l_k^M\}$ correspond to one of the semantic arguments of x.

7.5.5 Complexities

Time Complexity

Let T = (V, E, r, A, L) be a labeled rooted tree. T has N leaves associated with an N word sentence. Assume $od(v \in V) \leq K, v \in V$ [4]. So, this tree is a $K - ary$ tree. The number of internal nodes $I = |V| - N = |E| + 1 - N$.

For each $x \in V$, to find a first node of a path from T, we need to have $O(|E|)$ time [5]. To find the next node from the current node, if we have M classes, we need to have $O(MK)$ time. To form a path for x in T, from Section 3.4.5, we need to have $O(|E|MK)$ time. From the path, to find a set of roots. we need to have $O(|\tau|K)$ time. To form a rooted forest, we need to have $O(|E||R(x)|)$ time. To extract semantic arguments of the verbs from the rooted forest, we need to have $O(|F(x)||E|)$ The time complexity of the algorithm for x is $T_{c,x} = O(|\pi||E|) + O(|E|MK) + O(|\tau|K) + O(|E||R(x)|) + O(|F(x)||E|) = O(|E|)$. Therefore, the time complexity of the algorithm for finding the semantic arguments of a set of L verbs is

$$T_c = L * T_{c,x} = L * O(|E|) = O(|E|)$$

[4] $od(x)$ is the outgoing degree of the vertex x.
[5] It is down by *BFS* search.

Memory Complexity

To find the first node of a path for x from T, we need to have $O(|V| - N) = O(|V|)$ memory space. To find the next node from the current node, if we have M classes, we need to have $O(M)$ memory space. To form a path for x, we need to have $O(|V|)$ memory space. From the path, to find a set of roots, we need to have $O(K)$ memory space. To form a rooted forest, we need to have $O(|V|)$ memory space. To extract semantic arguments of the verbs from the rooted forest, we need to have $O(N)$. For obtaining a probability table, we need to have $O(|V|^2M) = O(|V|^2)$ memory space. So, the memory complexity of the algorithm for the verb x is $M_{c,x} = O(|V|) + O(M) + O(|V|) + O(K) + O(|V|) + O(N) + O(|V|^2) = O(|V|^2)$. Therefore, the memory complexity of the algorithm for finding the semantic arguments of a set of L verbs is

$$M_c = L * M_{c,x} = L * O(|V|^2) = O(|V|^2)$$

7.5.6 An Example

- The sentence: *Mrs. Hills said that the U.S. is still concerned about "disturbing developments in Turkey and continuing slow process in Malaysia".*
- The labeled rooted tree T for this sentence is in Figure 7.13.
- A path for the verb *concern* where $L(x) = VBZ$ is shown in Figure 7.14.
- A labeled rooted forest $F(x) = \{T_1, T_2, T_3\}$ is in Figures 7.15, 7.16, and 7.17.
- The semantic arguments of verb *concern* are:
  - *still*;
  - *the U.S.*;
  - *about "disturbing developments in Turkey and continuing slow process in Malaysia"*

7.5.7 Experiment and Discussion

The data set, the section 00 of WSJ from Penn Treebank and PropBank [52], is used for testing the data sequence dependence model (*DSDM*). Files *20, 37, 49, and 89* contain a total of 223 sentences.

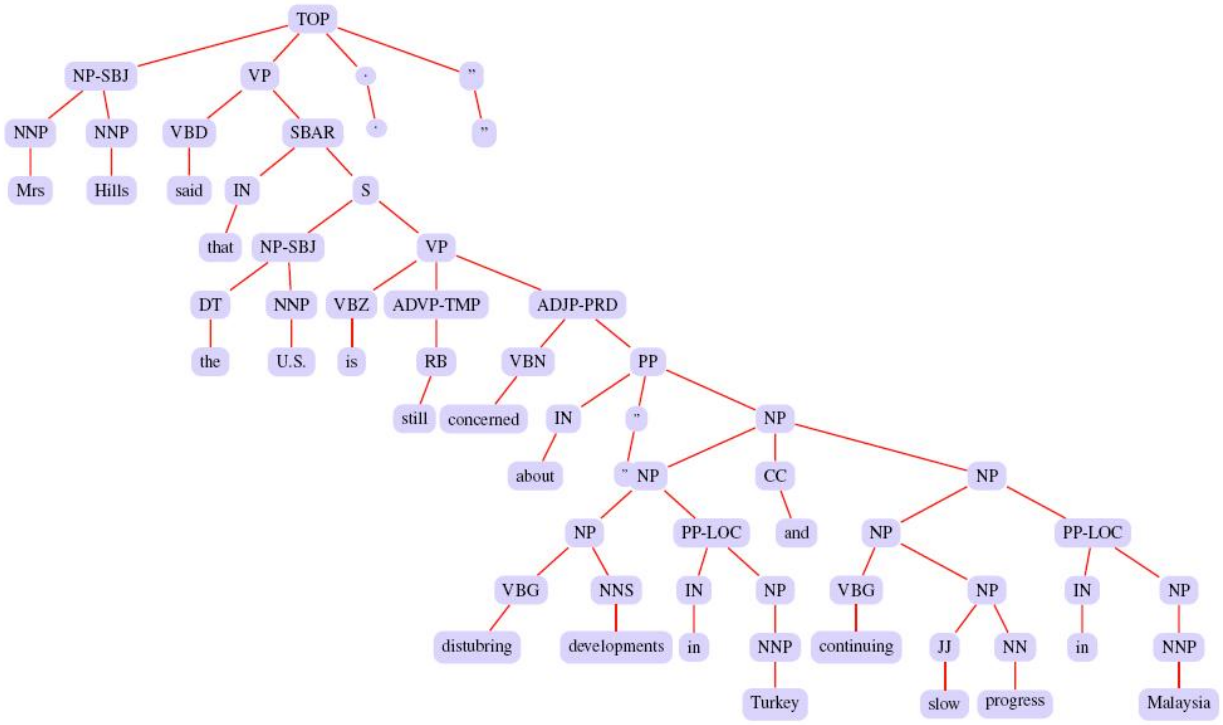


Figure 7.13: A parse tree of the sentence:
Mrs. Hills said that the U.S. is still concerned about "disturbing developments in Turkey and continuing slow process in Malaysia".

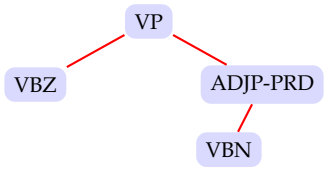


Figure 7.14: A path

All the semantic arguments of the verb *concern* can be extracted from this path.

Associated with each of these sentences, is an automatically determined parse tree provided by Penn Treebank. These parse trees have an average accuracy of 95.0%. Among these sentences, there are 621 verbs. Each verb has an average of three semantic arguments. Hence about 2000 semantic arguments are used. The semantic arguments are provided by PropBank. They are created manually.

Among 621 verbs, about 560 verbs are used for obtaining probability values while about 60

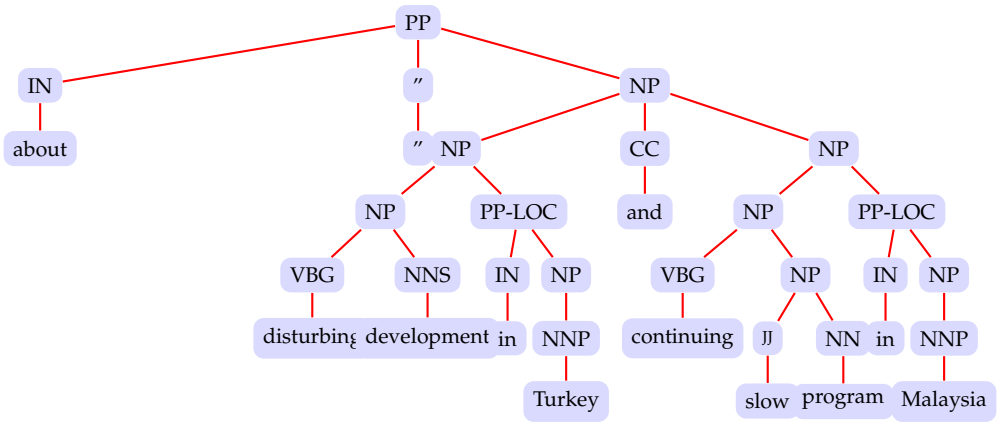


Figure 7.15: Labeled rooted tree T_3

T_3 is associating with one of the semantic arguments of verb *concern*.

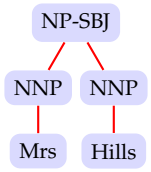


Figure 7.16: Labeled rooted tree T_1

T_1 is associating with one of the semantic arguments of verb *concern*.

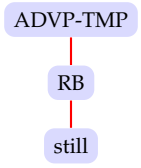


Figure 7.17: Labeled rooted tree T_2

T_2 is associating with one of the semantic arguments of verb *concern*.

verbs are used to form paths based on these probability values. Some of the paths are listed on Table 7.28. They are obtained based on procedure As described in Section 7.2.1 by applying 10-fold cross validation technique. We noticed that 86% paths fell into the first three patterns in Table 7.28. After forming a path for a verb in the test instances, a set of roots are found. From these roots, a

Table 7.28: Six types of paths

NO	%	Path
1	62.1	$VBZ(VBD, VBG, VBP, VBN, VB) \rightarrow VP$
2	14.2	$VB \rightarrow VP \rightarrow VP \rightarrow MD(TO)$
3	10.1	$VBP(VBZ, VBD) \rightarrow VP \rightarrow VP \rightarrow VBN$
4	4.2	$VBD(VBZ, VBN) \rightarrow VP \rightarrow RB \rightarrow VP \rightarrow VB$
5	2.4	$VBN \rightarrow VP \rightarrow VB \rightarrow VP \rightarrow VP \rightarrow TO$
6	2.2	$VBN \rightarrow VP \rightarrow VBP(VB) \rightarrow VP \rightarrow RB \rightarrow VP \rightarrow MD$

set of labeled rooted subtrees, whose leaves are associating with semantic arguments of the verb, are formed. Table 7.29 shows the results. On the average, each time among $\frac{1}{10}$ of the semantic arguments are classified, about 93% semantic arguments are correctly identified and 7% semantic arguments are mistakenly identified. By checking these classified instances, we found that our method is very effective in the case of a semantic argument being a sequence of consecutive words. However, if a semantic argument consists of two or more word fragments, separated by some phrases, our algorithm is less effective. The reason is that these phrases are parts of leaves of a tree induced from a root determined by our algorithm. This suggests that in order to exclude phrases from a semantic argument, we need to develop a method so that a set of subroots needs to be found. Each of them corresponds to a fragment of a semantic argument. Moreover, other misclassified instances are generated by errors carried in original syntactic trees.

Table 7.29: Testing *DSDM* for identifying semantic arguments of a verb on *WSJ* data

Files 20, 37, 49, 89	Precision %	Recall %	F-Measure %
Average	92.335	94.1675	93.2512
Standard Deviation	0.6195	0.5174	0.4605

Chapter 8

Conclusion

By researching on probabilistic graphical models and NLP tasks related semantics, a novel probabilistic graphical model called the data sequence dependence model (*DSDM*) has been introduced for discovering semantics in texts. With the model, a sequence of optimal class assignments for a sequence of input symbols is obtained based on maximizing the expected economic gain and also the joint probability. The maximum joint probability is achieved by finding class assignments that maximize a product of local conditional probabilities. With the model, the sequence of optimal class assignments is obtained in a simple way - no need for dynamic programming and fast - $O(nm)$ [1] compared with other existing models such as HMMs [1], MEMMs [2], or CRFs [3]. In order to do comparisons, we also discussed and implemented the context independence model (*CIM*) and the class sequence dependence model (*CSDM*). We studied and implemented a hidden Markov model (*HMM*). We studied the maximum entropy model (*MEM*). Test results on these models for identifying three types of semantic text patterns: the semantic arguments of a verb, the sense of an ambiguous word, and the noun phrases of a sentence on standard data sets have showed that the data sequence dependence model (*DSDM*) achieves the best performance.

8.1 Summary of Contributions

Our main contributions include:

[1] The length of a input sequence is n and the size of a set of classes is m.

1. A new probabilistic graphical model called the data sequence dependence model (*DSDM*) for a sequence of class assignments given a sequence of input symbols [6] [7] [8].

   It has fewer and different conditional independence assumptions compared with other commonly used probabilistic graphical models. It accounts for the data sequence dependency rather than the class sequence dependency. It gives a partial credit for each correct class assignment. It does not need to employ dynamic programming. It requires less operating time for identifying a new sequence of input symbols than competing techniques.

2. An algorithm for identifying the semantic arguments of a verb [9].

   For each verb in a parse tree $T = (V, E)$ associated with a sentence, the algorithm discovers a path. From the path, a set of roots are extracted. Each root induces a subtree of the parse tree. The leaves of a subtree constitute a semantic argument of the verb. The time complexity of the algorithm is $O(|E|)$ and the memory complexity of the algorithm is $O(|V|^2)$.

3. An algorithm for identifying the sense of a word [10].

   For the sequence of n context symbols of a polysemous word, the algorithm assigns a sequence class categories. Then, it determines the sense of the word by selecting the most frequently assigned class category. The time complexity of the algorithm is $O(nm)$ and the memory complexity of the algorithm is $O(n^2m)$, where m is the number of classes.

4. An algorithm for identifying the noun phrases of a sentence [11].

   For a sequence of n symbols associating with a sentence, the algorithm assigns a sequence of class categories. From the category sequence, the algorithm finds blocks such that each of these blocks is one of the noun phrases. The time complexity of the algorithm is $O(nm)$ and the memory complexity of the algorithm is $O(n^2m)$, where m is the number of classes.

5. Two other probabilistic graphical models [8].

   They are the context independent model (*CIM*) and the class sequence dependence model (*CSDM*). They also give a partial credit for each correct class assignment for a sequence of input symbols. However, the conditional independence assumptions are different from *DSDM*.

6. A method by employing statistical hypothesis testing.

   Papers [12] [14] have discussed statistical permutation tests for NLP tasks. In our case, we compare each pair of models by testing the null hypothesis that two models are equally good at identifying semantic patterns against the alternative hypothesis that one model is better able to identify semantic patterns. The resulting $p - value$ shows which model is better than the other.

8.2 Future Research

I will do more experiments on word sense disambiguation by including 100 polysemous words in the future since we only used limited number of polysemous words in our experiments.

I want to apply the data sequence dependence model (*DSDM*) for identifying name entities, for example, persons, organizations, or locations in texts. Combining with the information obtained from the semantic arguments of a verb, the meaning of an ambiguous word, the noun phrases, and the name entities in a sentence, a verb knowledge base will be built. For example, each concept in the knowledge base is a class of verbs [53]. A set of attributes for each concept might be who, what, where, when, and why. A link between two concepts is a relation between these concepts. Based on the knowledge hierarchy, an intelligent web search engine might be created.

I want to change *DSDM* by modifying some conditional independence assumptions to make it suitable for financial data analysis: predicting stock trends. For example, for a stock, we have a sequence of N trading days, each day is a tuple. The components in the tuple might be an average volume of a trading, an open price, an average price for the stock. Moreover, the components of the tuple can also include the average prices of other stocks and other variables, such as the price of *DJI*, and so on. For each trading day, I need to predict the closing price of the stock: $\{UP, DOWN\}$ compared to the closing price of the previous day. Therefore, each node in the conditional independence graph can be either a tuple or a class. There are three links for each current class node which are the current tuple to the class and the two previous tuples to the class. Based on this conditional independence graph, the joint probability function of a sequence of class for a sequence of N trading days can be computed.

Furthermore, I will do more theoretical studies on probabilistic graphical models. I will study how to build a directed conditional independence graph (Bayesian Network) by extracting relations between random variables from a sample population based on a joint probability distribution over all random variables[54]. By constructing a moral graph from a directed conditional independence graph, a triangulating graph from a moral graph, and an optimal junction tree from a triangulated graph, I will study the probability distribution $p(V)$ for any variable V from such a tree. Moreover, I will study the $p(V|e)$ where e is the context of evidence from the tree [55].

Appendix A

Probabilities and Graphs

In this section, we give a brief tutorial of properbilites and graphs to prepare for the discussion of the next chapter: probabilistic graphcial models.

A.1 Probability

A.1.1 Sample Space

For each random experiment, there are possible outcomes. each of these outcomes is called an elementary event. A set of such all elementary events is called a sample space S. For example, in the experiment of flipping three distinguishable coins, with each individual flip resulting in a head $\{H\}$ or a tail $\{T\}$, we can view the sample space as consisting of a set of 8 elementary events.

$$S = \{TTT, TTH, THT, THH, HTT, HTH, HHT, HHH\}$$

A.1.2 Event

An event is a subset of the sample space S. For example, in the experiment of flipping three coins, an event A of having three heads or three tails is $\{HHH, TTT\}$. The event S is called the certain event and ϕ is called the null event. Two events A and B are mutually exclusive if $A \cap B = \phi$.

A.1.3 Probability Properties

We use a probability function P to measure an event. P is defined as $P : S \to [0, 1]$, where S is a sample space and R is a set of real numbers. In our discussion, S is finite. P satisfies the following properties.

- $P(A) \geq 0$ for any event $A \subseteq S$.
- $P(S) = 1$
- $P(\cup_{i=1}^{N} A_i) = \sum_{i=1}^{N} P(A_i)$, for any $\{A_1, \ldots, A_N\}$ that is a partition of S.

Several other properties:

- $P(\phi) = 0$
- $P(A) \leq P(B)$ if $A \subseteq B$
- $P(\bar{A}) = 1 - P(A)$
- $P(A \cup B) = P(A) + P(B) - P(A \cap B) \leq P(A) + P(B)$

A.1.4 Conditional Probability

Let A and B be two events. If A and B are independent, then

$$P(A \cap B) = P(A)P(B)$$

In general:

$$P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$$

The conditional probability of an event A given another event B is defined as:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A|B)P(A)}{P(A \cap B) + P(\bar{A} \cap B)} = \frac{P(A|B)P(A)}{P(A)P(B|A) + P(\bar{A})P(B|\bar{A})}$$

Chain Rule

$$\begin{aligned} & P(A_1 \cap A_2 \cap A_3 \cap ... \cap A_N) \\ = \; & P(A_1|A_2, ...A_N)P(A_2|A_3, ..., A_N)...P(A_{N-1}|PA_N)P(A_N) \end{aligned}$$

A.1.5 Random Variables

A (discrete) random variable X maps events in a sample space S to the real numbers, $X : S \rightarrow R$. This allows us to work with the probability distribution induced on the resulting set of numbers.

Probability Density Function

For a random variable X and a real number x, we define the event $X = x$ to be $\{s \in S|X(s) = x\}$, then, the probability function $f(x)$ of the random variable X is defined as:

$$f(x) = P(X = x) = \sum_{\{s \in S,\ X(s)=x\}} P(s)$$

A.1.6 Joint Distribution

Two random variables induce a joint distribution

$$P(x_1, x_2) = P(X_1 = x_1, X_2 = x_2)$$

N random variables induce a joint distribution

$$P(x_1, \ldots, x_N) = P(X_1 = x_1, \ldots, X_N = x_N)$$

The joint distribution induces a marginal distribution on each individual random variable. For

random variable X_1,

$$P(x_1) = P(X_1 = x_1) = \sum_{x_2,\ldots,x_N} P(X_1 = x_1, \ldots, X_N = x_N) = \sum_{x_2,\ldots,x_N} P(x_1, \ldots, x_N)$$

Two random variables are independent, $X_1 \perp\!\!\!\perp X_2$, if and only if

$$P(x_1, x_2) = P(X_1 = x_1, X_2 = x_2) = P(X_1 = x_1)P(X_2 = x_2) = P(x_1)P(x_2)$$

Random variables $X_1, \ldots, X_N$ are jointly independent if and only if

$$\begin{aligned} & P(x_1, x_2 \ldots, x_{N-1}, x_N) \\ = \; & P(X_1 = x_1)P(X_2 = x_2) \ldots P(X_{N-1} = x_{N-1})P(X_N = x_N) \\ = \; & P(x_1)P(x_2) \ldots p(x_{N-1})P(x_N) \end{aligned}$$

If X_i and X_j are jointly independent then $X_i \perp\!\!\!\perp X_j$, $i \neq j$.

A.1.7 Conditional Probability

The conditional probability of random variable X_1 given random variable X_2 is defined by:

$$\begin{aligned} & P(x_1|x_2) = P(X_1 = x_1|X_2 = x_2) \\ = \; & \frac{P(X_1 = x_1, X_2 = x_2)}{P(X_2 = x_2)} = \frac{P(x_1, x_2)}{P(x_2)}, \quad \textit{when} \;\; P(x_2) > 0 \end{aligned}$$

If two random variables are independent then

$$\begin{aligned} p(x_1|x_2) &= P(X_1 = x_1 \mid X_2 = x_2) \\ &= P(X_1 = x_1) \\ &= P(x_1) \end{aligned}$$

Random variable X_1 is conditionally independent of random variable X_2 given random variable X_3, $X_1 \perp\!\!\!\perp X_2 \mid X_3$, if and only if.

$$\begin{aligned} p(x_1, x_2|x_3) &= P(X_1 = x_1, X_2 = x_2 \mid X_3 = x_3) \\ &= P(X_1 = x_1|X_3 = x_3)P(X_2 = x_2 \mid X_3 = x_3) \\ &= P(x_1|x_3)P(x_2|x_3) \end{aligned}$$

A.1.8 Expectation

The expected value (mean) of a discrete random variable X is defined by:

$$E[X] = \sum_x x \cdot P(X = x) = \sum_x x \cdot P(x)$$

- $E[X_1 + X_2] = E[X_1] + E[X_2]$
- $E[g(X)] = \sum_x g(x) \cdot P(X = g(x)) = \sum_x g(x) \cdot P(g(x))$, where g is a function, $g(X)$ is a random variable.
- X_1 and X_2 are uncorrelated if and only if $E[X_1X_2] = E[X_1]E[X_2]$. When X_1 and X_2 are independent, then X_1 and X_2 are uncorrelated.

A.1.9 Variance

$$\begin{aligned} V[X] &= E[(X - E[X])^2] \\ &= \sum_x P(X = x)(x - E[X])^2 = E[X^2] - (E[X])^2 \end{aligned}$$

- $E[X^2] = V[X] + E^2[X]$
- $V[aX] = a^2V[X]$
- $V[X_1 + X_2] = V[X_1] + V[X_2]$ when X_1 is uncorrelated with X_2 and certainly when X_1 is independent of X_2.

- $V[\sum_{i=1}^{N} X_i] = \sum_{i=1}^{N} V[X_i]$ when X_1, X_2, ..., X_N are jointly independent.

A.2 Graph

Let $G = (V, E)$ be a graph, where V is a set of nodes and E is a set of edges. G can be either a directed graph or a undirected graph. If G is a directed graph, G contains the directed edges, $E \subseteq V \times V$. In Figure A.1, graph G_1 is a directed graph while graph G_2 is an undirected graph.

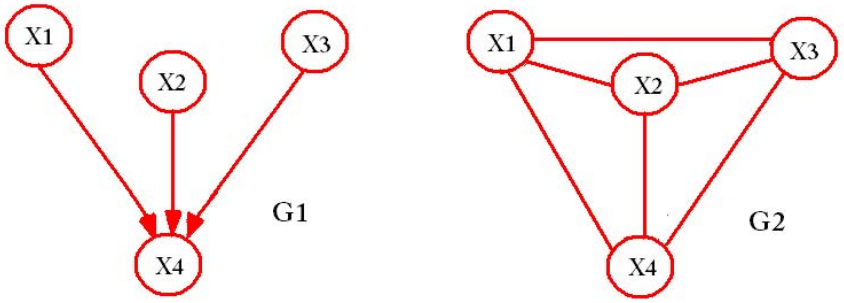


Figure A.1: $G_1 = (V_1, E_1)$ is a directed graph and $G_2 = (V_2, E_2)$ is an undirected graph. Where: $V_1 = \{x_1, x_2, x_3, x_4\}$ and $E_1 = \{(x_1, x_4), (x_2, x_4), (x_3, x_4)\}$; $V_2 = \{x_1, x_2, x_3, x_4\}$ and $E_2 = \{\{x_1, x_2\}, \{x_1, x_3\}, \{x_1, x_4\}, \{x_2, x_3\}, \{x_2, x_4\}, \{x_3, x_4\}\}$

A.2.1 Path

A path in a graph $G = (V, E)$ is a sequence of nodes $v_1, v_2, \ldots, v_{K-1}, v_K$ (or $v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_{K-1} \rightarrow v_K$) such that from each of its node v_k, there is an edge e_k to the next node v_{k+1} in the sequence: $\{v_k, v_{k+1}\} \in E$ (or $(v_k, v_{k+1}) \in E$), $i \neq j$.

A.2.2 Complete Graph

Let $G = (V, E)$ be a graph. G is a complete graph if and only if $E = \{\{a, b\} \subseteq V | a \neq b\}$. The graphs in Figure A.2 are complete graphs.

A.2.3 Subgraph

Let $G = (V, E)$ be a graph. $G' = (V', E')$ is a subgraph of G if and only if $V' \subseteq V$ and $E' \subseteq E$. For example, in Figure A.2, graph G_2 is a subgraph of G_1.

A.2.4 Induced Subgraph

Let $G = (V, E)$ be a graph. $\mathcal{H} = (U, E')$ is an induced subgraph of G restricted to U if $U \subseteq V$, E' contains all edges $\{a, b\}$ from G, for all $a, b \in U$. That is, $E' = \{\{a, b\} \in E | a, b \in U\}$. We write $\mathcal{H} = G|_U$ to mean $\mathcal{H}$ is the induced subgraph of G restricted to U. For example, in Figure A.2, graph G_2 is an induced subgraph of G_1 when G_1 is restricted to $\{x_1, x_2, x_4\}$.

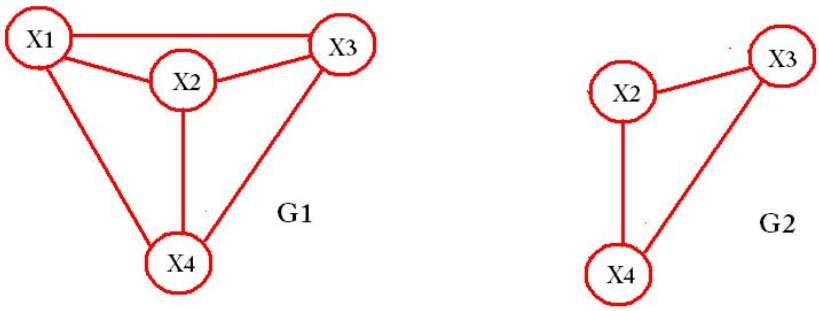


Figure A.2: $G_1 = (V_1, E_1)$ is a complete graph and $G_2 = (V_2, E_2)$ is a subgraph of G_1. G_2 is also an induced subgraph of G_1.

A.2.5 Maximally Complete Subset

Let G = (V, E) be a graph. Let a subset of nodes $U \subseteq V$ be given and $\mathcal{H} = G|_u$. We say that U is a maximally complete subset if and only if $\mathcal{H}$ is a complete graph and $T \supset U$ and $\mathcal{H} = G|_T$ being complete implies $T = U$.

A.2.6 Clique

Let $G = (V, E)$. A maximally complete subset $U \subset V$ is called a clique of G. For example, in Figure A.2, in G_1, $\{x_1, x_2, x_3, x_4\}$ is a clique. In G_2, $\{x_2, x_3, x_4\}$ is a clique. In Figure A.3, in G_2, $\{x_1, x_2, x_3, x_4\}$, $\{x_1, x_3, x_4, x_5\}$, and $\{x_4, x_5, x_7\}$ are cliques.

A.2.7 Directed Acyclic Graph (*DAG*)

From Section A.2 and Section A.2.1, we know that $G = (V, E)$ is a directed graph if and only if $E \subseteq V \times V$ and $v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_K$ is a path in G if and only if $(v_{k-1}, v_k) \in E, k = 1, \ldots, K$. A directed

graph $G = (V, E)$ is acyclic if and only if for each note $v \in V$, there is no path starting from v and getting back to itself. A direct acyclic graph is referred to as *DAC*.

A.2.8 Descendent of a Directed Acyclic Graph

Let $G = (X, E)$ be a *DAC*, the descendent of x is defined by

$$d(x) = \{y | \textit{for some } x_0, \ldots, x_N, x = x_0, (x_{n-1}, x_n) \in E, n = 1, \ldots, N\}$$

The non-descendent of x is defined by

$$nd(x) = X - (d(x) \cup \{x\})$$

A.2.9 Moral Graph

Let $G = (V, E)$ be a directed acyclic graph. The moral graph $G' = (V, E')$ of G is a undirected graph, where V is a set of the same nodes of G, E' is a set of undirected edges obtained by connecting nodes in G that have a common child, and then making all edges of G in G' undirected. For example, in Figure A.3, graph G_2 is the moral graph of G_1. In G, x_1, x_2, and x_3 are parents of x_4. Therefore, in G', we add all links between all parents of x_4.

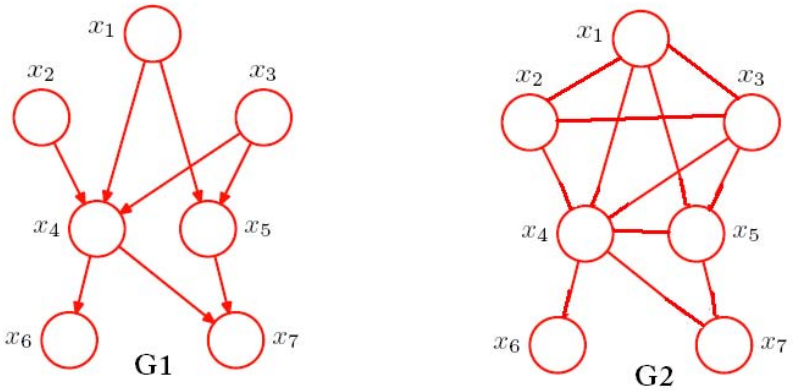


Figure A.3: G_2 is a moral graph of G_1

A.2.10 Chordal (Trianglated) Graph

Let $G = (V, E)$ be a graph. G is chordal if each of its cycles of four or more nodes has a chord, which is an edge joining two nodes that are not adjacent in the cycle. For example, in Figure A.3, graph G_2 is chordal.

A.2.11 Strongly Decomposable Graph

Let $G = (V, E)$ be a graph. G is strongly decomposable if and only if (1) G is triangulated, (2) the cliques of G can be put in running order of cliques $C_1, ..., C_K$ with separators $S_2, ..., S_K$ where

$$S_k = C_k \cap (\cup_{i=1}^{k-1} C_i), \qquad k = 2, ..., K$$

and each S_k is complete. For example, in Figure A.3, G_2 is a strongly decomposable graph. First, it is triangulated. Moreover, a running order of cliques is $C_1, C_2, C_3, C_4 = \{x_1, x_2, x_3, x_4\}, \{x_2, x_3, x_4, x_5\}$, $\{x_4, x_5, x_7\}$, $\{x_4, x_6\}$ with $S_2, S_3, S_4 = \{x_2, x_3, x_4\}$, $\{x_4, x_5\}, \{x_4\}$.

Appendix B

Some Proofs

B.1 Derivation of $p(d_k|x_1, \ldots, x_K)$

B.1.1 Joint Probability over Multinormal Distribution

Suppose we have K bins. The probability that an observation falls into bin k is p_k. To estimate the bin probabilities $p_1, \ldots, p_K$, we take a random sample of N observations. We find that of the N observations, x_1 observations fall into bin 1, x_2 observations fall into bin 2, ..., x_K observations fall into bin K. Under the protocol of the random sampling (multinormal distribution), the probability of observing counts $x_1, \ldots, x_K$ given the bin probabilities $p_1, \ldots p_k$ is

$$P(x_1, \ldots, x_K|p_1, \ldots, p_K) = C \cdot p_1^{x_1} \cdot p_2^{x_2} \cdot \ldots \cdot p_K^{x_K}, \quad \textit{where} \ \sum_{i=1}^{K} x_i = N, \quad \sum_{k=1}^{K} p_k = 1.$$

The coefficient C of $p_1^{x_1} \cdot p_2^{x_2} \cdot \ldots \cdot p_K^{x_K}$ from the expansion $(p_1 + p_2 + \ldots + p_K)^N$ is

$$\binom{N}{x_1, ..., x_K} = \frac{N!}{x_1! \cdot x_2! \cdot \ldots \cdot x_K!}.$$

Therefore,

$$P(x_1, \ldots, x_K|p_1, \ldots, p_K) = \binom{N}{x_1, x_2, ..., x_K} p_1^{x_1} \ldots p_k^{x_k} = \frac{N!}{x_1! \ldots x_K!} p_1^{x_1} \ldots p_K^{x_K}$$

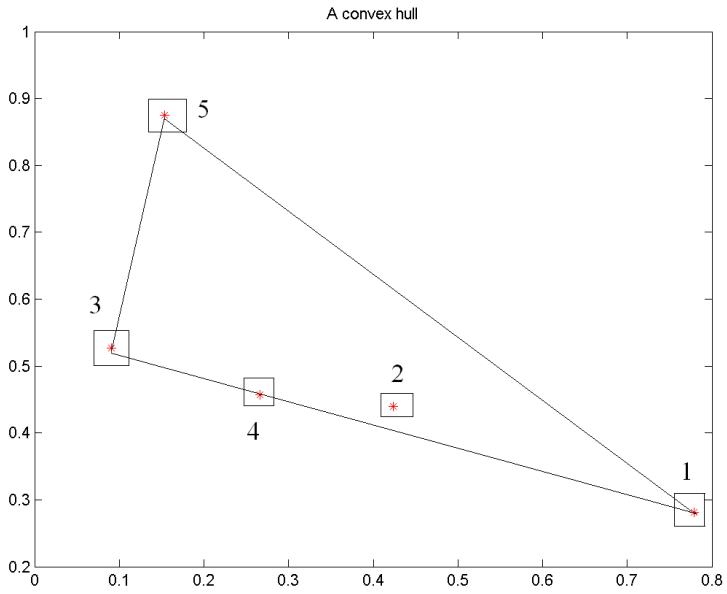


Figure B.1: The convex hull of X
$X = \{(0.7788, 0.2810), (0.4235, 0.4401), (0.0908, 0.5271), (0.2665, 0.4574), (0.1537, 0.8754)\}$

B.1.2 Probability of an Observation Falls into a Specific Bin

Let $x_1, \ldots, x_K$, be observations. We want to determine the probability that an observation falls in bin k. We denote this event by d_k. We want to determine $P(d_k|x_1, \ldots, x_K)$. To do this, we review some concepts.

B.1.3 Convex Hull

The convex hull of a set of points X is the smallest convex region containing all of the points X. For example, suppose $X = \{(0.7788, 0.2810), (0.4235, 0.4401), (0.0908, 0.5271), (0.2665, 0.4574), (0.1537, 0.8754)\}$ is a set of five ordered pairs. The convex hull of X is represented by a set of indices of the points of X, $\{1, 5, 3, 4\}$, that corresponds to the vertices of the convex hull. Figure B.1 shows the convex hull which is the region inside the red lines.

B.1.4 Simplex

A simplex is a generalization of a triangle or tetrahedron to arbitrary dimension. An $n-$simplex is an $n-$dimensional polytope which is the convex hull of its $n+1$ vertices. For example, a single point may be a $0-$simplex, a line may be $1-$simplex, a triangle may be a $2-$simplex, and a tetrahedron may be a $3-$simplex. By Section 6.3, corresponding to K bins, there is a $K-1$ simplex.

Probability of $P(q_1, \ldots, q_{K-1})$ over the $K-1$ Simplex

Let $(q_1, \ldots, q_K)$ be the probabilities distributed uniformly over the $k-1$ simplex, where $q_K = 1 - \sum_{k=1}^{K-1} q_k$. Then,

$$\int_{(q_1,\ldots,q_K)\in S} dq_1 \ldots dq_K = \frac{1}{(K-1)!}$$

and

$$\int_{(q_1,\ldots,q_K)\in S} \prod_{k=1}^{K} q_k^{x_k} d_{q_1} \ldots d_{q_K} = \frac{\prod_{k=1}^{K} x_k!}{(N+(K-1))!}$$

To compute $P(d_k|x_1, \ldots, x_k)$:

$$\begin{aligned}
P(d_k|x_1,\ldots,x_k) &= \frac{P(d_k, x_1, \ldots, x_K)}{P(x_1,\ldots,x_K)} \\
&= \frac{\int_{q_1,\ldots,q_K\in S} P(d_k, x_1, \ldots, x_K, q_1, \ldots, q_K) d_{q_1} \ldots d_{q_K}}{\int_{q_1,\ldots,q_K\in S} P(x_1, \ldots, x_K, q_1, \ldots, q_K) d_{q_1} \ldots d_{q_K}} \\
&= \frac{\int_{q_1,\ldots,q_K\in S} P(d_k, x_1, \ldots, x_k|q_1, \ldots, q_K) P(q_1, \ldots, q_K) d_{q_1} \ldots d_{q_K}}{\int_{q_1,\ldots,q_K\in S} P(x_1, \ldots, x_k|q_1, \ldots, q_K) P(q_1, \ldots, q_K) d_{q_1} \ldots d_{q_K}} \\
&= \frac{\int_{q_1,\ldots,q_K\in S} \frac{N!}{\prod_{k=1}^{K} x_k!} \prod_{k=1}^{K} q_k^{x_k} q_k \frac{1}{(K-1)!} d_{q_1} \ldots d_{q_K}}{\int_{q_1,\ldots,q_K\in S} \frac{N!}{\prod_{k=1}^{K} x_k!} \prod_{k=1}^{K} q_k^{x_k} \frac{1}{(K-1)!} d_{q_1} \ldots d_{q_K}} \\
&= \frac{\int_{q_1,\ldots,q_K\in S} q_1^{x_1} q_2^{x_2} \ldots q_{k-1}^{x_{k-1}} q_k^{x_k+1} q_{k+1}^{x_{k+1}} \ldots q_K^{x_K} d_{q_1} \ldots d_{q_K}}{\int_{q_1,\ldots,q_K\in S} q_1^{x_1} q_2^{x_2} \ldots q_{k-1}^{x_{k-1}} q_k^{x_k} q_{k+1}^{x_{k+1}} \ldots q_K^{x_K} d_{q_1} \ldots d_{q_K}} \\
&= \frac{\frac{x_1!x_2!\ldots(x_k+1)!x_{k+1}!\ldots x_k!}{(N+K)!}}{\frac{x_1!x_2!\ldots x_k!x_{k+1}!\ldots x_k!}{(N+(K-1))!}} = \frac{\frac{(x_k+1)!}{(N+K)!}}{\frac{x_k!}{(N+(K-1))!}} = \frac{x_k+1}{N+K}
\end{aligned} \tag{B.1}$$

The prior distribution over the K bins does not have to be taken as uniform. The natural prior distribution over K bins is the Dirichlet distribution.

$$P(p_1, ..., p_K|\alpha_1, ..., \alpha_K) = \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} p_k^{\alpha_k} - 1$$

The uniform distribution on the $K-$ Simplex is a special case of the Dirichlet distribution where $\alpha_k = 1$, $k = 1, ..., K$. The Beta distribution is a special case of the Dirichlet distribution for $K = 2$. The maximum density occurs at

$$p_k = \frac{\alpha_k - 1}{(\sum_{k=1}^{K} \alpha_k) - K}, \qquad when \quad \alpha_k \geq 1$$

In the case of the Dirichlet prior distribution, the derivation goes in a similar manner.

$$\begin{aligned}
P(d_k|x_1, \ldots, x_K) &= \frac{P(d_k, x_1, \ldots, x_K)}{P(x_1, \ldots, x_k)} \\
&= \frac{\int_{q_1,\ldots,q_K \in S} P(d_k, x_1, \ldots, x_k, q_1, \ldots, q_K) d_{q_1} \ldots d_{q_K}}{\int_{q_1,\ldots,q_K \in S} P(x_1, \ldots, x_k, q_1, \ldots, q_K) d_{q_1} \ldots d_{q_K}} \\
&= \frac{\int_{q_1,\ldots,q_K \in S} P(d_k, x_1, \ldots, x_k|q_1, \ldots, q_K) P(q_1, \ldots, q_K) d_{q_1} \ldots d_{q_K}}{\int_{q_1,\ldots,q_K \in S} P(x_1, \ldots, x_k|q_1, \ldots, q_K) P(q_1, \ldots, q_K) d_{q_1} \ldots d_{q_K}} \\
&= \frac{\frac{(x_k+\alpha_k) \prod_{n=1}^{K} (x_n+\alpha_n-1)!}{(N+\sum_{k=1}^{K} \alpha_k)(N-1+\sum_{k=1}^{K} \alpha_k)!}}{\frac{\prod_{n=1}^{K} (x_n+\alpha_n-1)!}{(N-1+\sum_{k=1}^{K} \alpha_k)!}} = \frac{x_k + \alpha_k}{N + \sum_{n=1}^{K} \alpha_n}
\end{aligned} \tag{B.2}$$

B.2 Proof of Equation (6.6)

In order to show Equation (6.6), suppose we randomly draw a sample of size N. In this sample, a species Δ occurs r times. In the population, the probability of Δ being the i^{th} species in $p_1, \ldots, p_i, \ldots, p_s$, where $p_1, \ldots, p_i, \ldots, p_s$ is an uniform distribution, can be computed as follows:

$$P(p_\Delta = p_i|p_1, \ldots, p_i, \ldots, p_s) = \frac{1}{s}$$

Moreover, for any i, s.t, $(p_i + q_i)^N$, where $q_i = 1 - p_i$. By the Binormal theorem, the probability of observed number of occurrences is r given the species Δ being the i^{th} one can be obtained as:

$$P(r|(p_\Delta = p_i|p_1, \ldots, p_i, \ldots, p_s)) = \binom{N}{r} p_i^r (1 - p_i)^{N-r}$$

Moreover, the sum over all s species of the probabilities $P(r|(p_\Delta = p_i|p_1, \ldots, p_i, \ldots, p_s)) = \binom{N}{r} p_i^r (1 - p_i)^{N-r}$ that each occurs r times, s.t.

$$E[n_r] = \sum_{i=1}^{s} \binom{N}{r} p_i^r (1 - p_i)^{N-r} \tag{B.3}$$

Therefore, by Bayes' theorem, the probability of species Δ is the i^{th} one under the observed number of occurrences being r is:

$$\begin{aligned}
& P((p_\Delta = p_i|p_1, \ldots, p_i, \ldots, p_s)|r) \\
= & \frac{P(r|(p_\Delta = p_i|p_1, \ldots, p_i, \ldots, p_s) \cdot P(p_\Delta = p_i|p_1, \ldots, p_i, \ldots, p_s)}{\sum_{i \in s} P(r|(p_\Delta = p_i|p_1, \ldots, p_i, \ldots, p_s) \cdot P(p_\Delta = p_i|p_1, \ldots, p_i, \ldots, p_s)} \\
= & \frac{\binom{N}{r} p_i^r (1 - p_i)^{N-r} \frac{1}{s}}{\sum_{i=1}^{s} \binom{N}{r} p_i^r (1 - p_i)^{N-r} \frac{1}{s}} \\
= & \frac{\binom{N}{r} p_i^r (1 - p_i)^{N-r}}{\binom{N}{r} \sum_{i=1}^{s} p_i^r (1 - p_i)^{N-r}} \\
= & \frac{p_i^r (1 - p_i)^{N-r}}{\sum_{i=1}^{s} p_i^r (1 - p_i)^{N-r}}
\end{aligned}$$

Let q_r be the probability of an arbitrary species that is presented r times in the sample. Therefore, $P(q_r = p_i|p_1, \ldots, p_i, \ldots, p_s) = P((p_\Delta = p_i|p_1, \ldots, p_i, \ldots, p_s)|r)$. The expected value $E[q_r]$ can be computed as:

$$\begin{aligned}
E[q_r] &= \sum_{k=1}^{s} p_k \cdot \frac{p_k^r (1 - p_k)^{N-r}}{\sum_{i=1}^{s} p_i^r (1 - p_i)^{N-r}} \\
&= \frac{\sum_{k=1}^{s} p_k^{r+1} (1 - p_k)^{(N+1)-(r+1)}}{\sum_{i=1}^{s} p_i^r (1 - p_i)^{N-r}}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{\binom{N}{r}\sum_{k=1}^{s}\binom{N+1}{r+1}p_k^{r+1}(1-p_k)^{(N+1)-(r+1)}}{\binom{N+1}{r+1}\sum_{i=1}^{s}\binom{N}{r}p_i^{r}(1-p_i)^{N-r}} \\
&= \frac{\binom{N}{r}E[n_{r+1}]}{\binom{N+1}{r+1}E[n_r]} \\
&= \frac{\frac{N!}{r!(N-r)!}E[n_{r+1}]}{\frac{(N+1)!}{(r+1)!(N-r)!}E[n_r]} \\
&= \frac{(r+1)E[n_{r+1}]}{(N+1)E[n_r]}
\end{aligned}
$$

Because of $N >> 1$, $N \approx N+1$, we consider $E[q_r]$

$$
E[q_r] = \frac{(r+1)E[n_{r+1}]}{NE[n_r]} \approx \frac{(r+1)}{N}\frac{E[n_{r+1}]}{E[n_r]} \approx \frac{(r+1)}{N}\frac{N_{r+1}}{N_r}
$$

Therefore, we proved that Equation (6.6) is True.

Variance of q_r

By (6.6),

$$
\begin{aligned}
E[q_r^m] &\approx \frac{(r+m)^{<m>}}{N^m}\frac{N_{r+m}}{N_r} \qquad 1 \le r \le N, 0 \le m \le N \\
&= \frac{(r+m)(r+m-1)\dots(r+2)(r+1)}{N^m}\frac{N_{r+m}}{N_{r+m-1}}\frac{N_{r+m-1}}{N_{r+m-2}}\cdots\frac{N_{r+2}}{N_{r+1}}\frac{N_{r+1}}{N_r} \\
&= \frac{(r+m)N_{r+m}}{NN_{r+m-1}}\frac{(r+m-1)N_{r+m-1}}{NN_{r+m-1}}\cdots\frac{(r+2)N_{r+2}}{NN_{r+1}}\frac{(r+1)N_{r+1}}{N_r} \\
&= E[q_{r+m-1}]E[q_{r+m-2}]\dots E[q_{r+1}]E[q_r] \qquad \text{(B.4)}
\end{aligned}
$$

The variance of q_r is:

$$
\begin{aligned}
Var(q_k) &= E[q_r^2] - (E[q_r])^2 \\
&= E[q_{r+1}]E[q_r] - (E[q_r])^2 \\
&= E[q_r](E[q_{r+1}] - E[q_r])
\end{aligned}
$$

B.3 Computing the Joint Probability $p(\cdot)$

Computing $p(c_1, \ldots, c_N | s_1, \ldots, s_N)$

From Figure 3.2,

$$
\begin{aligned}
& p(c_1, \ldots, c_N, s_1, \ldots, s_N) \\
= \; & \frac{\prod_{n=1}^{N-1} p(s_n, s_{n+1}, c_n) p(s_n, s_{n+1}, c_{n+1})}{\prod_{m=1}^{N-1} p(s_m, s_{m+1}) \prod_{m=2}^{N-1} p(s_m, c_m)} \\
= \; & \frac{1}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times \frac{\prod_{n=1}^{N-1} p(s_n, s_{n+1}, c_n)}{\prod_{m=2}^{N-1} p(s_m, c_m)} \times \prod_{n=1}^{N-1} p(s_n, s_{n+1}, c_{n+1}) \\
= \; & \frac{p(s_1, s_2, c_1)}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times \frac{\prod_{n=2}^{N-1} p(s_n, s_{n+1}, c_n)}{\prod_{m=2}^{N-1} p(s_m, c_m)} \times \prod_{n=1}^{N-1} p(s_n, s_{n+1}, c_{n+1}) \\
= \; & \frac{p(s_1, s_2, c_1)}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times \prod_{n=2}^{N-1} p(s_{n+1} | s_n, c_n) \times \prod_{m=2}^{N} p(s_{m-1}, s_m, c_m)
\end{aligned}
$$

Setting $n = m - 1$ in the last product:

$$
\begin{aligned}
& p(c_1, \ldots, c_N, s_1, \ldots, s_N) \\
= \; & \frac{p(s_1, s_2, c_1) p(s_{N-1}, s_N, c_N)}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times \prod_{n=2}^{N-1} p(s_{n+1} | s_n, c_n) \times \prod_{m=2}^{N-1} p(s_{m-1}, s_m, c_m) \\
= \; & \frac{p(s_1, s_2, c_1) p(s_{N-1}, s_N, c_N)}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times \prod_{n=2}^{N-1} p(s_{n+1} | s_n, c_n) p(s_{n-1}, s_n, c_n) \\
= \; & \frac{p(s_1, s_2, c_1) p(s_{N-1}, s_N, c_N)}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times \prod_{n=2}^{N-1} p(s_{n+1} | s_n, c_n) p(s_{n-1} | s_n, c_n) p(s_n | c_n) p(c_n) \\
= \; & \frac{1}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times p(s_2 | s_1, c_1) p(s_1 | c_1) p(c_1) \times p(s_{N-1} | s_N, c_N) p(s_N | c_N) p(c_N) \\
& \times \prod_{n=2}^{N-1} p(s_{n+1} | s_n, c_n) p(s_{n-1} | s_n, c_n) p(s_n | c_n) p(c_n)
\end{aligned}
$$

Define $p(s_0 | s_1, c_1) = 1$ and $p(s_{N+1} | s_N, c_N) = 1$, then:

$$
p(c_1, \ldots, c_N, s_1, \ldots, s_N)
$$

$$
\begin{aligned}
&= \frac{1}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times p(s_2|s_1, c_1)p(s_0|s_1, c_1)p(s_1|c_1)p(c_1) \\
&\quad \times p(s_{N+1}|s_N, c_N)p(s_{N-1}|s_N, c_N)p(s_N|c_N)p(c_N) \\
&\quad \times \prod_{n=2}^{N-1} p(s_{n+1}|s_n, c_n)p(s_{n-1}|s_n, c_n)p(s_n|c_n)p(c_n) \\
&= \frac{1}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})} \times \prod_{n=1}^{N} p(s_{n-1}|s_n, c_n)p(s_{n+1}|s_n, c_n)p(s_n|c_n)p(c_n)
\end{aligned}
$$

Define $M_{s_1,\ldots,s_N} = \frac{1}{\prod_{m=1}^{N-1} p(s_m, s_{m+1})}$, then:

$$
\begin{aligned}
& p(c_1, \ldots, c_N, s_1, \ldots, s_N) \\
&= M_{s_1,\ldots,s_N} \times \prod_{n=1}^{N} p(s_{n-1}|s_n, c_n)p(s_{n+1}|s_n, c_n)p(s_n|c_n)p(c_n)
\end{aligned}
$$

Therefore the conditional probability:

$$
\begin{aligned}
& p(c_1, \ldots, c_N|s_1, \ldots, s_N) \\
&= \frac{p(c_1, \ldots, c_N, s_1, \ldots, s_N)}{\sum_{c'_1,\ldots c'N \in C} p(c'_1, \ldots, c'_N, s_1, \ldots, s_N)} \\
&= \frac{\prod_{n=1}^{N} p(s_{n-1}|s_n, c_n)p(s_{n+1}|s_n, c_n)p(s_n|c_n)p(c_n)}{\sum_{c \in C} \prod_{k=1}^{N} p(s_{k-1}|s_k, c_k)p(s_{k+1}|s_k, c_k)p(s_k|c_k)p(c_k)}
\end{aligned}
$$

B.4 Prove $p(s_{n,k-1}|c, s_{n,k})p(s_{n,k+1}|c, s_{n,k})p(s_{n,k}|c)p(c)$ is a Probability Function

Let:

$$
P(c|T_{n,k}) = p(s_{n,k-1}|c, s_{n,k})p(s_{n,k+1}|c, s_{n,k})p(s_{n,k}|c)p(c)
$$

1. It is non-negative for all events of c, $s_{n,k-1}$, $s_{n,k}$, $s_{n,k+1}$
2. $\sum_{c \in C} \sum_{T_{n,k} \in T_{\cdot,k}} P(c|T_{n,k}) = 1$

Proof :

Let $c = X_1$, $s_{n,k-1} = X_2$, $s_{n,k} = X_3$, $s_{n,k+1} = X_4$, $T_{n,k} = Y$

$$
\begin{aligned}
& \sum_{X_1 \in C} \sum_{Y \in T_{\cdot,k}} p(X_1|Y) \\
= & \sum_{X_1} \sum_{X_2} \sum_{X_3} \sum_{X_4} p(X_2|X_1, X_3) p(X_4|X_1, X_3) \\
& p(X_3|X_1) p(X_1) \\
= & \sum_{X_1} \sum_{X_2} \sum_{X_3} p(X_2|X_1, X_3) p(X_3|X_1) p(X_1) \\
& \sum_{X_4} p(X_4|X_1, X_3) \\
= & \sum_{X_1} \sum_{X_2} \sum_{X_3} p(X_2|X_1, X_3) p(X_3|X_1) p(X_1) \\
= & \sum_{X_1} \sum_{X_3} p(X_3|X_1) p(X_1) \sum_{X_2} p(X_2|X_1, X_3) \\
= & \sum_{X_1} \sum_{X_3} p(X_3|X_1) p(X_1) \\
= & \sum_{X_1} p(X_1) \sum_{X_3} p(X_3|X_1) \\
= & \sum_{X_1} p(X_1) \\
= & 1
\end{aligned}
$$

B.5 The Curves of the Three Models

From Section 7.3.3, for each model, we have a set of 20 accuries. We randomly select 10 accuries and make an average. We do this 5000 times. We make a histgram of 30 bins. Figure B.2 shows the results of *CIM*, *DSDM*, and *CSDM*.

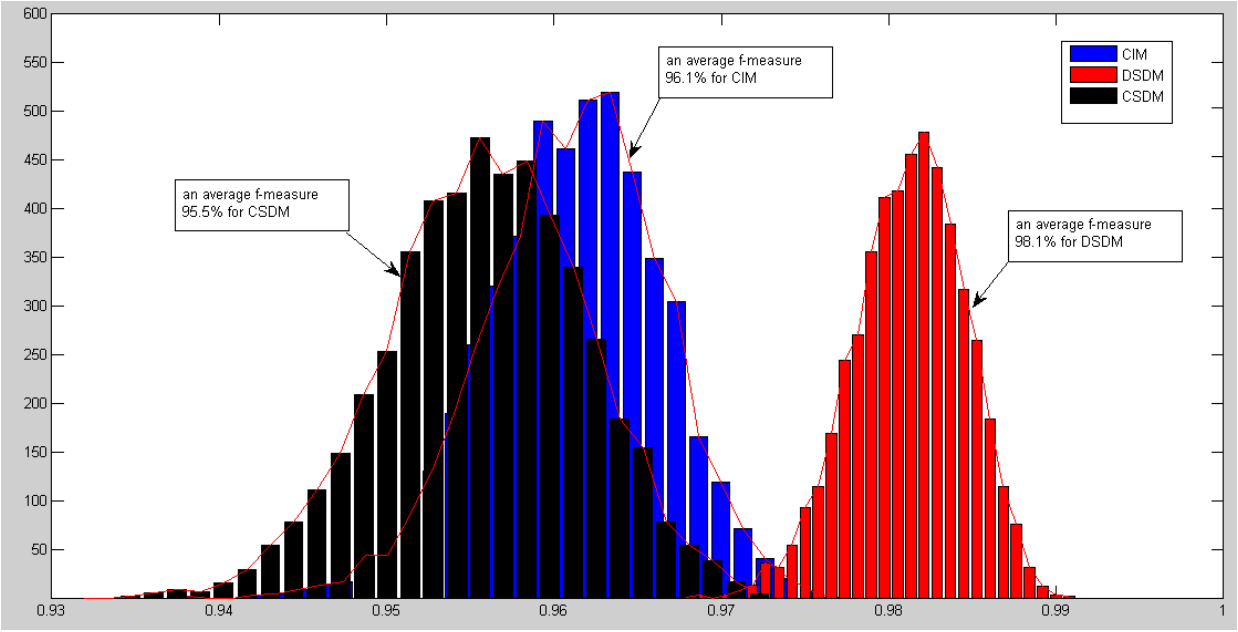


Figure B.2: $f-measure$ curves for *CIM*, *DSDM*, and *CSDM*

Bibliography

[1] Molina, A., Pla, F., tics, D.D.S.I., Hammerton, J., Osborne, M., Armstrong, S., Daelemans, W.: Shallow parsing using specialized hmms. Journal of Machine Learning Research **2** (2002) 595–613

[2] MaCallum, A., Freitag, D., Pereira, F.: Maximum entropy markov models for information extraction and segmentation. In: Proceedings of 17th International Conf. on Machine Learning. (2000) 591–598

[3] Lafferty, J., MaCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of 18th International Conf. on Machine Learning. (2001) 282–289

[4] Haralick, R.M., Shapiro, L.G.: Computer and Robot Vision, Vol. I. Addison-Wesley (1991)

[5] group, P.: Nltk 2.0 documentation. `http://nltk.org/api/nltk.classify.html#module-nltk.classify.maxent` (2012)

[6] Huang, M., Haralick, R.M.: Recognizing Patterns in Texts. River (2010)

[7] Huang, M., Haralick, R.M.: Discovering text patterns by a new graphic model. Transactions on Machine Learning and Data Mining **4**(2) (2011) 55–74

[8] Huang, M., Haralick, R.M.: Developing an algorithm for mining semantics in texts. In: 13th International Conference on Intelligent Text Processing and Computational Linguistics. (2012) 247–260

[9] Huang, M., Haralick, R.M.: An algorithm of identifying semantic arguments of a verb from structured data. In: The International Conference Recent Advances in Natural Language Processing. (2011)

[10] Huang, M., Haralick, R.M.: Discovering semantics of a word from a sentence. In: 2010 International Conference on Artificial Intelligence and Pattern Recognition. (2010) 51–57

[11] Huang, M., Haralick, R.M.: Developing a probabilistic graphic model for identifying semantic patterns in texts. In: In Fifth IEEE International Conference on Semantic Computing. (2011) 197–200

[12] Koehn, P.: Statistical significance tests for machine translation. In: Proceedings of EMNLP. (2004)

[13] Yeh, A.: More accurate tests for the statistical significance of result differences. In: Proceedings of Coling 2000. (2000)

[14] Morgan, W.: Statistical hypothesis tests for nlp. http://masanjin.net/sigtest.pdf (2005)

[15] Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2002)

[16] Lauritzen, S.L.: Graphical Models. Oxford, United Kingdom: Clarendon Press (1996)

[17] Sha, F., Fereira, F.: Shallow parsing with conditional random fields. In: Proceedings of HLT-NAACL. (2003) 213–220

[18] Manning, C., Schutze, H.: Foundations of statistical natural language processing. The MIT Press Cambridge (2003)

[19] Mihalcea, R., D., M.: A highly accurate bootstrapping algorithm for word sense disambiguation. In: International Journal on Artificial Intelligence Tools. Volume 10(1-2). (2001) 5 – 21

[20] Mihalcea, R.: Unsupervised large-vocabulary word sense disambiguation with graph- based algorithms for sequence data labeling. In: The Human Language Technology/Empirical Methods in Natural Language Processing. (2005)

[21] Gale, W., Church, K., Yarowsky, D.: a method for disambiguating word senses in a large corpus. In: Computers and the Humanities. Volume 26. (1992) 415–439

[22] Leacock, C., Towell, C., Voorhees, E.: corpus-based statistical sense resolution. In: The ARPA Workshop on Human Language Technology. (1993) 260–265

[23] Leacock, C., Chodorow, M.: Combining Local Context and WordNet Similarity for Word Sense identification, In WordNet, An Electronic Lexical Database. (1998)

[24] Leacock, C., Chodorow, M., Miller, G.: Using corpus statistics and wordnet relations for sense identification. **24(1)** (1998) 147–165

[25] Yarowsky, D., Florian, R.: evaluating sense disambiguation across diverse parameter spaces. **8(2)** (2002) 293–310

[26] Mooney, R.: Comparative experiments on disambiguating word senses: An illustration of the role of bias in machine learning. In: the 1st Conference on Empirical Methods in Natural Language Processing. (1996) 82–91

[27] Yarowsky, D.: decision lists for lexical ambiguity resolution: Application to accent restoration in spanish. In: The 32nd annual meeting of association for computational Linguistics. (1994)

[28] Quinlan, J.R.: C4.5 Programs for Machine Learning. (1993)

[29] Salton, G., Wong, A., Yang, C.: a vector space model for information retrieval. **18(11)** (1975) 613–620

[30] Church, K.W.: A stochastic parts program and noun phrase parser for unrestricted text. In: Proceedings of the second conference on Applied natural language processing. (1988) 136 – 143

[31] Ratnaparkhi, A.: A maximum entropy model for part-of-speech tagging, booktitle = The Empirical Methods in Natural Language Processing, year = 1996, pages = 133–142

[32] Wu-Chieh, Wu, Lee, Y.S., Yang, J.C.: Robust and efficient multiclass svm models for phrase pattern recognition. Pattern Recognition **41** (2008) 2874–2889

[33] Kudo, T., Mastumoto, Y.: Chunking with support vector machines. In: The second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies. (2001) 1–8

[34] Furnkranz, J.: round robin classification. **2** (2002) 721–747

[35] hyeun Park, S., Frnkranz, J.: Efficient pairwise classification. In: ECML 2007. LNCS (LNAI, Springer (2007) 658–665

[36] Gildea, D., Jurafsky, D.: Automatic labelling of semantic roles. Computational Linguistics (2002) 245–288

[37] Hacioglu, K.: Semantic role labeling using dependency trees. In: Proceedings of Coling 2004, Geneva, Switzerland, COLING (Aug 23–Aug 27 2004) 1273–1276

[38] Cohn, T., Blunsom, P.: Semantic role labelling with tree conditional random fields. In: Proceedings of CoNLL-2005 Shared Task. (2005)

[39] Baldewein, U., Erk, K., Pad, S., Prescher, D.: Semantic role labeling with chunk sequences. In: Proceedings of CoNLL-2004 Shared Task. (2004)

[40] Brahaj, A.: List of english stop words. In: http://www.iusb.edu/ libg/instruction/helpguide/handouts/2005Boolean.shtml. (2009)

[41] Lovins, J.B.: Development of a stemming algorithm. mechanical translation and computational linguistics (1968)

[42] M.F.Porter: An algorithm for suffix stripping. Program (1980) 130–137

[43] Good, I.J.: the population frequencies of species and the estimation of population parameters. **40(3-4)** (1953) 237–264

[44] Gale, W.A.: good-turing smoothing without tears. **2** (1995)

[45] Carreras, X., Mrquez, L.: Phrase recognition by filtering and ranking with perceptrons. In: the International Conference on Recent Advances on Natural Language Processing. (2003)

[46] Veenstra, J., den BoschJ, A.V.: Single-classifier memory-based phrase chunking. In: Preceedings of CoNLL-2000 and LLL-2000. (2000) 157–159

[47] Leacock, C., Miller, G.A., Chodorow, M.: Using corpus statistics and wordnet relations for sense identification. Computational Linguist. **24** (1998) 147–165

[48] Bruce, R., Wiebe, J.: Word-sense disambiguation using decomposable models. In: Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics. (1994) 139–146

[49] Bird, S., Klein, E., Loper, E.: Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit. O'Reilly Media (2009)

[50] Levin, E., Sharifi, M., Ball, J.: Evaluation of utility of lsa for word sense discrimination. In: Preceedings of HLT-NAACL. (2006) 77 – 80

[51] Leacock, C., Towell, G., Voorhees, E.: Corpus based statistical sense resolution. In: Proceedings of the workshop on Human Language Technology. (1993) 260 – 265

[52] Weischedel, R., Palmer, M., Marcus, M., Hovy, E.: Ontonotes release 2.0 with ontonotes db tool v. 0.92 beta and ontoviewer v.0.9 beta. In: http://www.bbn.com/NLP/OntoNotes. (2007)

[53] Levin, B.: English Verb Classes and Alternations. (1993)

[54] Spirtes, P., Glymour, C.: an algorihtm for fast recovery of sparse causal graphs. **9**(1) (1991) 62–72

[55] Huang, C.: inference in belief networks: A procedural guide. **15**(3) (1996) 225–263