

VOTING BETWEEN MULTIPLE DATA REPRESENTATIONS FOR TEXT CHUNKING

by

Hong Shen

B.Eng. Shanghai University 1995
B.Sc. The University of Manitoba 2002



A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

In the School

of

Computing Science



© Hong Shen 2004



SIMON FRASER UNIVERSITY



June 2004

All rights reserved. This work may not be
reproduced in whole or in part, by photocopy
or other means, without permission of the author.



APPROVAL

Name: Hong Shen

Degree: Master of Science (Computing Science)

Title of Project: Voting Between Multiple Data Representations for Text Chunking

Examining Committee:

Chair: Dr. Richard (Hao) Zhang
Assistant Professor of Computing Science



Dr. Anoop Sarkar
Senior Supervisor
Assistant Professor of Computing Science



Dr. Martin Ester
Supervisor
Associate Professor of Computing Science



Dr. Fred Popowich
Internal Examiner
Professor, School of Computing Science
Simon Fraser University

Date Approved: _____



ABSTRACT

One major goal of research on Natural Language Processing (NLP) is to process and understand multiple languages. There is arguably a close link between understanding language and the hierarchical analysis of linguistic utterances or sentences. To achieve this goal, much research in NLP has focused on an intermediate task, text chunking, which is the task of finding non-recursive phrases in a given sentence of natural language text. Most of the successful text chunking methods proposed in the last decade have been achieved using machine learning techniques.

Recent research shows the combination approach, using simple majority voting or more complex techniques like boosting, is more successful than a single learning model. Voting can be in terms of system combination or data representation (DR) combination. In this project, we consider the hypothesis that voting between *multiple data representations* can be more accurate than voting between *multiple learning models*. To show the power of the data representation combination, we present that a simple learning method, in our case a simple trigram Hidden Markov Model (HMM), combined with DR voting techniques can achieve a result better than the best on the CoNLL-2000 text chunking data set. Without using any additional knowledge sources, we achieved 94.01 $F_{\beta=1}$ score for arbitrary phrase identification which is equal to previous best comparable score of 93.91 and 95.23 $F_{\beta=1}$ score for Base NP phrase identification which is better than the current comparable state-of-the-art score of 94.22. In addition, our chunker is

considerably faster and simpler than comparably accurate methods in training as well as decoding.

ACKNOWLEDGEMENTS

I would like to thank my senior supervisor, Dr. Anoop Sarkar. He provided me with creative idea and insightful directions for this work. I have to say I could not complete my degree so quickly without his kind help and patience.


Thanks to my supervisor Dr. Martin Ester and my Examiner Dr. Fred Popowich who spent considerable time reading my project and made several thoughtful suggestions.

Many thanks to Dr. Richard Zhang for chairing my defence and providing useful advice.

Thanks to Val for helping me with the administration details.

Finally, I thank my wife and parents for their unconditional love and support, and thank to my Aunt Ping Shen and Uncle Dr. LiYan Yuan, who provided me great help in my life.

TABLE OF CONTENTS


Approval	1
Abstract	2
Acknowledgements	4
Table of Contents	5
List of Figures	7
List of Tables	8
 Chapter One: Introduction	9
1.1 Motivation	9
1.2 Shallow Parsing	10
1.2.1 Part-of-Speech (PoS) Tagging	11
1.2.2 Text Chunking	12
1.3 Project Contribution	15
1.4 Project Organization	16
Chapter Two: Overview of CoNLL-2000 Shared Task	17
2.1 Task Background.....	17
2.2 Dataset	17
2.3 Performance Evaluation Metric.....	18
2.4 Chunk Types.....	19
2.5 Approach Summary	19
2.6 Results	21
2.7 Base NP Chunking Background.....	21
2.7.1 Data and Evaluation.....	21
2.7.2 Results	22
2.8 Chapter Summary	22
Chapter Three: Background to the Approach	24
3.1 Markov Chains	24
3.2 Hidden Markov Model	24
3.2.1 Viterbi Algorithm	26
3.3 Data Representation.....	28
3.3.1 Inside/Outside	28
3.3.2 Start/End (O+C).....	29
3.4 Voting Techniques.....	30
3.4.1 Majority Voting	31
3.5 Chapter Summary	32
Chapter Four: Text Chunking Approach	34
4.1 Specialized HMM Chunking.....	34

4.2 Voting Between Multiple Data Representations	38
4.3 Chapter Summary	39
Chapter Five: Chunking Evaluation.....	40
5.1 Dataset	41
5.1.1 Arbitrary Chunking Dataset (CoNLL-2000 Dataset)	41
5.1.2 Base NP Dataset (Base NP Chunking Dataset)	41
5.2 TnT Tagger.....	41
5.2.1 File Formats	42
5.2.2 Running TnT.....	44
5.2.3 Evaluation	44
5.3 Experimental Results.....	44
5.3.1 Text Chunking (Arbitrary Phrase Chunking)	44
5.3.2 Base NP Chunking (Noun Phrase Chunking).....	48
5.4 Results Comparison.....	48
5.4.1 Text Chunking Comparison.....	48
5.4.2 Base NP Chunking Comparison	50
5.4.3 Comparison with Kudo's Approach	51
5.5 Analysis	53
5.6 Chapter Summary	55
Chapter Six: Conclusion.....	57
Chapter Seven: Future Work	58
Bibliography	59

LIST OF FIGURES

Figure 1 Example of Chinese word segmentation.	11
Figure 2 Example of Hidden Markov Model.....	25
Figure 3 Example of Viterbi Algorithm.....	28
Figure 4 Running time comparison for single data representation between SP+Lex-WCH and [KM01] on arbitrary chunking task.	53
Figure 5 Example of a new representation.	58

LIST OF TABLES

Table 1 PoS and chunk tagging example.....	18
Table 2 Results based on CoNLL-2000 shared task.....	21
Table 3 Results based on Base NP chunking task.....	22
Table 4 The noun chunk tag sequences for the example sentence, <i>In early trading in Hong Kong Monday, gold was quoted at \$366.50 an ounce.</i>	30
Table 5 Example of majority voting results among five data representations (DRs)	32
Table 6 Example of specialization where the words belong to the predefined  lexical set W_s	36
Table 7 Format of lexicon, untagged and tagged files.....	43
Table 8 Format of lexicon files.....	43
Table 9 Format of n-gram files.....	43
Table 10 Text chunking results for each setting.....	45
Table 11 Text chunking results of 5DR majority voting with SP+Lex-WCH in IOB2 format.....	46
Table 12 Text chunking results of 3DR majority voting with SP+Lex-WCH in IOB2 format.....	46
Table 13 Text chunking accuracy for all DRs in five evaluation formats. Note each column represents the evaluation format and each row represents the training and testing format.....	47
Table 14 Text chunking accuracy for all DRs evaluated in IOB2 format. Note that voting format is the format when conducting majority voting, all the DRs are converted into this format.....	47
Table 15 Text chunking accuracy for all DRs evaluated in IOE1 format.....	48
Table 16 Base NP chunking accuracy for all DRs evaluated in IOB1 format.....	48
Table 17 Comparison of text chunking accuracy with major approaches.....	49
Table 18 Comparison of Base NP chunking accuracy with major approaches.....	50
Table 19 Text chunking Error distribution between SP+Lex-WCH w/voting and [KM01].....	52
Table 20 Base NP chunking Error distribution between SP+Lex-WCH w/voting and [KM01].....	52
Table 21 McNemar’s test between Specialized HMM w/ voting and [KM01] on two chunking tasks.....	52

CHAPTER ONE: INTRODUCTION

1.1 Motivation

A major goal of research on Natural Language Processing (NLP) is to process and understand multiple languages. However, not all NLP applications require a complete syntactic analysis. A full parse often provides more information than needed and sometimes less. For example, in Information Retrieval, it may be enough to find simple noun phrases and verb phrases. In Information Extraction, Language Summarization, and Question Answering, researchers are only interested in information about some specific syntactic or semantic relations such as agent, object, location, time, etc (basically, who did what to whom, when, where and why), rather than elaborate hierarchical or recursive syntactic analyses. The CoNLL-2003 shared task, for example, is only interested in persons, locations, organizations, and other entities. E.g.:

[U.N._ORG] [official_O] [Ekeus_PER] [heads_O] [for_O] [Baghdad_LOC].

In above example, there are four Name Entity Recognition (NER) tags. Tag *ORG* is for organizations; Tag *PER* is for persons; Tag *LOC* is for locations; and Tag *O* is for others we are not interested in.

To achieve above goals, much research in NLP has focused on intermediate tasks that make sense of some of the structure inherent in language without requiring complete understanding.

1.2 Shallow Parsing

Shallow parsing, the task of recovering only a limited amount of syntactic information from natural language sentences – has proved to be a useful technology for written and spoken language domains. For example, within the Verbmobil project, shallow parsers were used to add robustness to a large speech-to-speech translation system [Wah00]. Shallow parsers are also typically used to reduce the search space for full-blown, ‘deep’ parsers [Col96]. Yet another application of shallow parsing is question-answering on the World Wide Web, where there is a need to efficiently process large quantities of (potentially) ill-formed documents [BD01] [SL99]. More generally all text mining applications can be viewed as applications of shallow parsing, e.g. from biology and health literature [SPT98] [HOA+02].

Due to the fact that the phrases are assumed to be non-overlapping, the phrase boundaries can be treated as labels, one per word in the input sentence, and *sequence learning* or *sequence prediction* techniques such as the viterbi algorithm can be used to find the most likely sequence of such labels. Hence, we can consider shallow parsing as a *sequence learning* task.

Sequence learning is simply defined as assigning a sequence of classes to some given sequences. The following well-known tasks are considered sequence learning tasks.

- Part-of-Speech tagging
- Chunking
- Name Entity Recognition
- Word Segmentation

For example, in Chinese, there is no space between words. To parse this Chinese sentence, we have to identify the boundaries between words. The technique developed to deal with this problem is called Chinese word segmentation. We can treat Chinese word segmentation as a sequence learning problem by tagging each Chinese character with a certain symbol to show its word boundary. The following sentence is the example result after segmentation, where symbol **B** is for the beginning of a Chinese word and symbol **I** is for the rest character within a Chinese word.

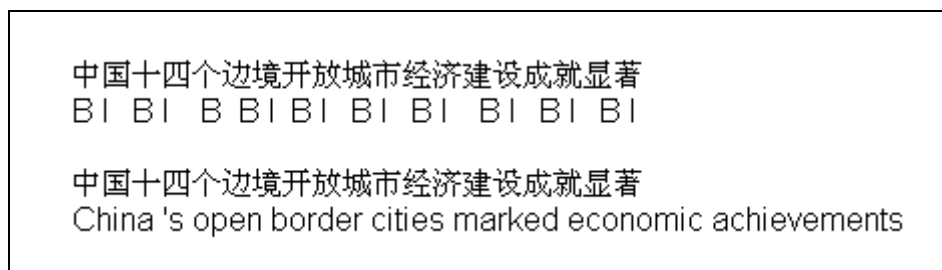


Figure 1 Example of Chinese word segmentation.

1.2.1 Part-of-Speech (PoS) Tagging

Part-of-speech (PoS) tagging, or simply tagging, is the task of labelling (or tagging) each word in a sentence with its appropriate part of speech. E.g.:

[He_PRP] [closes_VBZ] [the_DT] [door_NN].



Data-driven PoS tagging has benefited a lot from machine learning techniques, i.e. the annotation of words with the contextually appropriate PoS tags, often including morphological features. The main advantage with data-driven PoS taggers is that they are language and tag set independent and thereby methods are easily applicable to new languages and domains [Meg02]. The average accuracy reported for state-of-the-art data-driven PoS taggers lies between 95% and 98% depending on the corpus and language. However, it is important to realize that this impressive accuracy figure is not quite as

good as it looks, because it is evaluated on a per-word basis. For instance, in many genres such as newspapers, the average sentence is over twenty words, and on such sentences, even with a tagging accuracy of 96%, this means that there will be on average over one tagging error per sentence. Even though it is limited, the information we get from tagging can be quite useful. For example, tagging results can be used to solve shallow parsing tasks.

1.2.2 Text Chunking

In the past years, some attempts have been made to build data-driven shallow parsers. The main goal of the data-driven shallow parsers is, above all, to find the phrase structure of the sentence. As a first step in building corpus-based parsers, a considerable amount of research has been carried out to find syntactically related non-overlapping groups of words, so-called *chunks*. A chunk is a major phrase category consisting of the phrasal head and its modifiers on the left hand side. The example below illustrates three different chunk types NP (noun phrase), VP (verb phrase) and PP (prepositional phrase) for the sentence “He reckons the current account deficit will narrow to only \$1.8 billion in September.”

[He_IN] [reckons_VP] [the current account deficit_NP] [will narrow_VP]
[to_PP] [only \$1.8 billion_NP] [in_PP] [September_NP].

Text chunking was suggested as a pre-processing step of a parser by [Abn91]. Ten years later, most statistical parsers contained a chunking phase (e.g. [Rat98]).

Text chunking is defined as given the words and their morphosyntactic class, decide which words can be grouped as chunks (noun phrases, verb phrases, complete

clauses, etc). The goal of text chunking is to divide each sentence of a given text into non-overlap syntactic units. Text chunking can help solve many NLP tasks, such as information extraction, text summarization and spoken language understanding.

The chunking task is divided into two subtasks: finding only noun phrases (*Base NP chunking* or *Noun Phrase Chunking*) and identifying arbitrary chunks (*Text Chunking* or *Arbitrary Phrase Chunking*).

Machine learning approaches towards noun phrase chunking started with work by [Chu88] who used bracket frequencies associated with POS tags for finding noun phrase boundaries in text. In an influential paper about chunking, [RM95] show that chunking can be regarded as a tagging task. Even more importantly, the authors propose a training and test data set that is still being used for comparing different text chunking methods. These data sets were extracted from the Wall Street Journal part of the Penn Treebank II corpus (Marcus et al., 1993). Sections 15-18 are used as training data and section 20 as test data. The Penn Treebank Project annotates naturally-occurring text for linguistic structure. Most notably, they produce skeletal parses showing rough syntactic and semantic information -- a *bank* of linguistic *trees*. They also annotate text with part-of-speech tags [San90]. The Treebank bracketing style is designed to allow the extraction of simple predicate/argument structure. Over one million words of text are provided with this bracketing applied. In principle, the noun phrase chunks present in the material are noun phrases that do not include other phrases, with initial material (determiners, adjectives, etc.) up to the head but without post-modifying phrases (prepositional phrases or clauses) [RM95].

The noun phrase chunking data produced by [RM95] contains a couple of nontrivial features. First, unlike in the Penn Treebank, possessives between two noun phrases have been attached to the second noun phrase rather than the first. An example in which round brackets mark chunk boundaries:

(*Nigel Lawson*) ('s *restated commitment*): the possessive 's has been moved from *Nigel Lawson* to *restated commitment*.

Second, Treebank annotation may result in unexpected noun phrase annotations: *British Chancellor of (the Exchequer) Nigel Lawson* in which only one noun chunk has been marked. The problem here is that neither *British Chancellor* nor *Nigel Lawson* has been annotated as separate noun phrases in the Treebank. Both *British ... Exchequer* and *British ... Lawson* are annotated as separate noun phrases in the Treebank. But these phrases could not be used as noun chunks because they contain the smaller noun phrase *the Exchequer*.

The major researches of data-driven text chunking have been directed to recognize base NP chunks (e.g. [Chu88], [CP98], [SB98]) and detect other chunk types (e.g. [RM95], [ADK98], [Bra99], [BVD99], [Vee99], [Osb00], [Meg01a], [MP02], [TKS02], [KM01], and [ZDJ02]). The first area is focused on recognition methods for simple, non-recursive noun phrases. These phrases play an important role in many application areas, such as information retrieval, information extraction and question answering. The latter pays attention to develop promising methods to detect other chunk types, such as prepositional phrases (PP), adverb phrases (ADVP), adjective phrases (ADJP) and verb phrases (VP). In general, researchers put their most energy on combining linguistic information (e.g. Lexical information) with chunk detection

methods, extending studies to deal with various language corpora, and applying different learning methods (e.g. Rule-based learning, Transformation-based Learning, Memory-based Learning, Hidden Markov Models, Maximum Entropy, Support Vector Machines, Winnow, etc.).

1.3 Project Contribution

In this project, we consider the hypothesis that voting between *multiple data representations* can be more accurate than voting between *multiple learning models*. The main contribution of this paper is that a single learning method, in our case a simple trigram Hidden Markov Model can use voting between multiple data representations to obtain results equal to the best on the CoNLL-2000 text chunking data set. Using no additional knowledge sources, we achieved 94.01 $F_{\beta=1}$ score for arbitrary phrase identification compared to the previous comparable best score of 93.91 [KM01]. The highest score reported on this data set is 94.17 [ZDJ02], but this result used a full-fledged parser as an additional knowledge source. Without the parser, the result obtained was 93.57. There have been over 30 publications with different methods on this CoNLL-2000 data set with result from 77.07 to 94.17. It is therefore a very competitive data set with small significant difference likely to have an impact on many sequence learning problems. In addition, we achieved 95.23 $F_{\beta=1}$ score for Base NP phrase identification compare to the previous comparable best score of 94.22 [KM01]. By the paired McNemar test, we showed our result is significantly different from [KM01] on this task.


Based on our empirical results, we show that choosing the right representation (or the types of features used) can be a very powerful alternative in sequence prediction, even when used with relatively simple machine learning methods.

1.4 Project Organization

The remainder of this project is organized as follows: Chapter 2, gives an overview of the CoNLL-2000 (Conference of Natural Language Learning 2000) shared task for data-driven text chunking. Chapter 3, describes some important concepts related in this study. Chapter 4, presents the combined chunking approach. Chapter 5, describes the experiments on various lexical features and combinations. Chapter 6, concludes the project. Chapter 7, draws out the implications for the future work.

CHAPTER TWO: OVERVIEW OF CONLL-2000 SHARED TASK

2.1 Task Background

The early idea of chunking is initially described by Arvin Joshi in 1957. He first developed a parser by using chunking technique. Later on in 1990's, chunking was, again, recognized as an important intermediate approach toward a full parsing. Lance Ramshaw  Mitch Marcus have approached chunking by using a machine learning method [RM95]. Their work has inspired many others to study the application of learning methods to noun phrase chunking. Other chunk types are considered less useful than NP chunks. The most complete work is [BVD99] which presents results for NP, VP, PP, ADJP and ADVP chunks. [Vee99] works with NP, VP and PP chunks. [RM95] have recognized arbitrary chunks but classified every non-NP chunk as a VP chunk. The CoNLL-2000 shared task attempts to fill this gap [CoN03] [NP02].

2.2 Dataset

The CoNLL-2000 dataset for this task is available online [CoN03]. This dataset consists of the same partitions of the Wall Street Journal corpus (WSJ) as the widely used data for noun phrase chunking: sections 15-18 as training data (211727 tokens) and section 20 as test data (47377 tokens). The annotation of the data has been derived from the WSJ corpus by a program written by Sabine Buchholz from Tilburg University, The Netherlands.

The training and test data consist of three columns separated by spaces. Each word has been put on a separate line and there is an empty line after each sentence. The first column contains the current word, the second its part-of-speech tag as derived by the Brill tagger and the third its chunk tag as derived from the WSJ corpus. The chunk tags contain the name of the chunk type, for example I-NP for noun phrase words and I-VP for verb phrase words. Most chunk types have two types of chunk tags, B-CHUNK for the first word of the chunk and I-CHUNK for each other word in the chunk. This chunk representation is in IOB2 format. We will describe it in detail in Section 3.2. Here is an example of the file format:


Word	PoS Tag [San90]	Chunk Tag
He	PRP	B-NP
Reckons	VBZ	B-VP
The	DT	B-NP
Current	JJ	I-NP
Account	NN	I-NP
Deficit	NN	I-VP
Will	MD	B-VP
Narrow	VB	I-VP
	TO	B-PP
Only	RB	B-NP
#	#	I-NP
1.8	CD	I-NP
Billion	CD	I-NP
In	IN	B-PP
September	NNP	B-NP
.	.	O

Table 1 PoS and chunk tagging example.

2.3 Performance Evaluation Metric

The performance on this task is measured with three rates. First, the percentage of detected phrases that are correct - *Precision*. Second, the percentage of phrases in the data that were found by the chunker - *Recall*. And third, the $F_{\beta=1}$ rate which is equal to



$\frac{(\beta^2 + 1) \times \textit{precision} \times \textit{recall}}{\textit{recall} + \beta^2 \times \textit{precision}}$ with $\beta=1$ [Rij75]. The latter rate has been used as the target

for optimization.

2.4 Chunk Types

The chunk types are based on the syntactic category part of the bracket label in the Treebank (cf. [BFK+95] p.35). Roughly, a chunk contains everything to the left of and including the syntactic head of the constituent of the same name. Some Treebank constituents do not have related chunks. The head of S (simple declarative clause) for example is normally thought to be the verb, but as the verb is already part of the VP chunk, no S chunk exists in our example sentence. Besides the head, a chunk also contains pre-modifiers (like determiners and adjectives in NPs), but no post-modifiers or arguments. This is why the PP chunk only contains the preposition, and not the argument NP, and the SBAR chunk consists of only the complementizer [SB00].

2.5 Approach Summary

[SB00] divided the systems that have been applied to the CoNLL-2000 shared task into four groups:

- Rule-based systems: derive a set of rules (or regular expressions), from the training data, which corresponds to chunking decisions to be made.
- Memory-based systems: classify data based on their similarity to data that they have seen earlier. (e.g. MBL)
- Statistical systems: apply various machine learning models to make the chunking decision. (e.g. HMM, ME, Winnow, and SVM)

- Combined systems: apply more than one technique to make chunking decision.

Recent research shows many state-of-the-art works are focused on including more features (e.g. position features between words or/and tags) to train a discriminative model (e.g. SVM and Winnow) rather than to train a generative model (e.g. traditional HMM). This change leads to the big improvement of chunking accuracy. The representative of this approach is [KM01] and [ZDJ02]. The common part of these approaches is that they all treat sequence learning problem as a classification task and involves more features. The difference is that [KM01] proposed to train eight different SVM classifiers, two for each Inside/Outside representation (forward and backward parsing), then vote among their results. They achieved 93.91 $F_{\beta=1}$ score. However, [ZDJ02] applied generalized Winnow with enhanced features in the training process. With the enhanced feature, they achieved 94.17 $F_{\beta=1}$ score, while without the enhanced feature, they obtained 93.57 $F_{\beta=1}$ score. These enhanced features are based on English Slot Grammar (ESG), which does not produce the same bracketed representation as that used in the CoNLL-2000 shared task. Also, ESG has the capability to produce multiple ranked parses for a sentence, a full parser, which is totally different compared with other approaches without the knowledge of a full parse of a sentence. Thus, we consider their method not to be directly comparable with ours.

2.6 Results

Paper	System	Method	Precision	Recall	$F_{\beta=1}$
[ZDJ02]	Statistical	Gen. Winnow w/ full parser	94.28%	94.07%	94.17
[KM01]	Statistical	SVM w/voting	93.89%	93.92%	93.91
[ZDJ02]	Statistical	Gen. Winnow w/o full parser	93.54%	93.60%	93.57
[CM03]	Statistical	Perceptrons	94.19%	93.29%	93.74
[KM00]	Statistical	SVM	93.45%	93.51%	93.48
[Hal00]	Combined	WPDV(Comb)	93.13%	93.51%	93.32
[TKS00]	Combined	MBL(Comb)	94.04%	91.00%	92.50
[ZST02]	Statistical	Specialized HMM	91.96%	92.41%	92.19
[ZST00]	Combined	HMM w/ MBL	91.99%	92.25%	92.12
[Dej00]	Rule	Rule-based	91.87%	92.31%	92.09
[Koe00]	Statistical	ME	92.08%	91.86%	91.97
[Os00]	Statistical	ME	91.65%	92.23%	91.94
[VB00]	Statistical	MBL	91.05%	92.03%	91.54
[PMP00]	Statistical	HMM	90.63%	89.65%	90.14
[Joh00]	Rule	Rule-based	86.24%	88.25%	87.23
[VD00]	Rule	Rule-based	88.82%	82.91%	85.76
Baseline		Most Frequent Chunk Tag	72.58%	82.14%	77.07

Table 2 Results based on CoNLL-2000 shared task.

2.7 Base NP Chunking Background

Unlike arbitrary phrase identification, Base NP phrase identification, or simply NP chunking, deals with identifying the chunks that consist of noun phrases (NPs). NP chunking task is initially introduced by Ramshaw and Marcus in 1995.



2.7.1 Data and Evaluation

NP chunking dataset, like CoNLL-2000 dataset, consists of the section 15-18 of WSJ as training data and the section 20 of WSJ as test data. The PoS is again derived from a tagger written by Sabine Buchholz from Tilburg University, The Netherlands. However, instead of using representation IOB2 in CoNLL-2000 dataset, this dataset use representation IOB1 as its correct chunk tag.

The data format and evaluation method are exactly the same as introduced in CoNLL-2000 dataset section.

2.7.2 Results

Paper	Precision	Recall	$F_{\beta=1}$
[KM01]	94.15%	94.29%	94.22
[TDD+00]	94.18%	93.55%	93.86
[TKS00]	93.63%	92.89%	93.26
[MPR+99]	92.40%	93.01%	92.80
[XTAG99]	91.80%	93.00%	92.40
[TV99]	92.50%	92.25%	92.37
[RM95]	91.80%	92.27%	92.03
Baseline	78.20%	81.87%	79.99

Table 3 Results based on Base NP chunking task.

2.8 Chapter Summary

This chapter introduced **CoNLL-2000 shared task**, *text chunking*, and **Base NP chunking task**, which consists of dividing a sentence into syntactical units. Text chunking is an intermediate step towards full parsing. The goal of this task is to come forward with machine learning methods which after a training phrase can recognize the arbitrary chunk segmentation of the test data as well as possible.

The CoNLL-2000 shared task and Base NP chunking task has the same part of the Wall Street Journal corpus (WSJ), section 15-18 as training data and section 20 as test data. However, their correct chunk type representation is different. CoNLL-2000 dataset uses IOB2, while Base NP dataset uses IOB1.

The chunker for both dataset is evaluated with $F_{\beta=1}$ score,

$$F_{\beta=1} = \frac{2 * precision * recall}{recall + precision} \quad [\text{Rij79}], \text{ where } \mathbf{precision} \text{ is the percentage of detected}$$

phrases that are correct and **recall** is the percentage of phrases in the data that were found by the chunker.

All the **chunking methods** are fall into four groups: *rule-base systems*, *memory based systems*, *statistical systems*, and *combined systems*. The $F_{\beta=1}$ **score** of text chunking is ranged from 77.07 to 94.17 and $F_{\beta=1}$ **score** of Base NP chunking is ranged from 79.99 to 94.22.

CHAPTER THREE: BACKGROUND TO THE APPROACH

3.1 Markov Chains

Markov processes/chains/models were first developed by Andrei A. Markov (a student of Chebyshev). Their first use was actually for a linguistic purpose – modeling the letter sequences in works of Russian literature (Markov 1913) – but Markov models were then developed as a general statistical tool [MS99].

A *Markov chain* is a sequence of random values whose probabilities at a time interval depend on the value of the number at the previous time. Thus, if $X = (X_1, \dots, X_T)$ is a sequence of random variables taking values in some finite set $S = \{s_1, \dots, s_N\}$, the state space, then the Markov chain has following properties:

(1) Limited Horizon, the transition probabilities at a time interval depends on its previous time. $P(X_{t+1} = s_k | X_1, \dots, X_t) = P(X_{t+1} = s_k | X_t)$

(2) Time invariant, the transition probabilities is independent of time interval and does not vary with time. $P(X_{t+1} = s_k | X_1, \dots, X_t) = P(X_2 = s_k | X_1)$

3.2 Hidden Markov Model

Hidden Markov Models was published by Baum-welch. Often we want to consider a sequence (perhaps through time) of random variables that are not independent, but rather the value of each variable depends on previous elements in the sequence. For

many such systems, it seems reasonable to assume that all we need to predict the future random variables is the value of the past random variable, and we do not need to know the values of all the past random variables in the sequence. That is, future elements of the sequence are conditionally independent of past elements, given the present element. In a Visible Markov Model (VMM) or Markov Chain, we know what states the machine is passing through, so the state sequence or some deterministic function of it can be regarded as the output, while in a HMM, you do not know the state sequence that the model passes through, but only some probabilistic function of it.

In the following example, there are two states q and r . Two pairs of *emission probabilities* represented as a and b respectively in the Figure 1. π_i represents the *initial probabilities*. The probabilities on the transition curves are *transition probabilities* [Sar03].

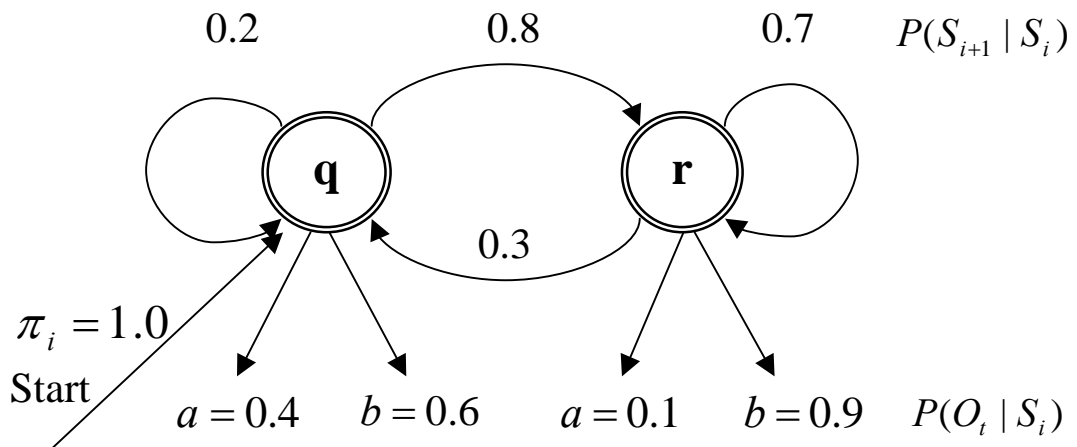


Figure 2 Example of Hidden Markov Model.

$$\sum_x P(q \rightarrow x) = P(q \rightarrow r) + P(q \rightarrow q) = 1.0$$

$$\sum_x P(\text{emit}(q, x)) = P(\text{emit}(q, a)) + P(\text{emit}(q, b)) = 1.0$$

In above two equations, S_i represents *state*; O_i represents *output*; x represents *hidden state* [Sar03] [MS99]. The first equation expresses the total transition probabilities started from the initial state is 1 and, similarly, the second one describes the sum of the total emission probabilities is 1.

Given a certain observation sequence O_{training} , we can find the model parameters μ that maximize $P(O | \mu)$ using Maximum Likelihood Estimation, $\arg \max_{\mu} P(O_{\text{training}} | \mu)$.

This maximization process is often referred to as *training* the model.

Given the observation sequence O and a model μ , we can efficiently compute $P(O | \mu)$ - the probability of the observation given the model. This process is often referred to as *decoding*.

HMMs are useful when one can think of underlying events probabilistically generating surface events. HMM is as a language model: compute probability of a given observation sequence, HMM is as a parser: compute the best sequence of states for a given observation sequence, and HMM is as a learner: given a set of observation sequences, learn its distribution [Sar03] [MS99].

3.2.1 Viterbi Algorithm

The Viterbi algorithm is based on HMMs and used to find the most likely complete path for a given observation sequence through a trellis. The Viterbi algorithm is

linear in input m for n states, $O(mn^2)$. Alternatively, we can enumerate all paths, however, this takes exponential time, $O(n^m)$. One widespread use of the Viterbi algorithm is tagging – assigning parts of speech (or other classes) to the words in a text. We think of there being an underlying Markov chain of parts of speech from which the actual words of the text are generated. This can be applied to text chunking (or shallow parsing task) in the same way. In this project, we used an existing tagger, TnT, which is implemented based on this idea – Best Path (Viterbi) algorithm. In order to find the most likely complete path, that is (where μ represents the model parameters)

$$\arg \max_x P(X | O, \mu)$$

To do this, it is sufficient to maximize for a fixed O ,

$$\arg \max_x P(X, O | \mu)$$

The key idea of this algorithm to compute this trellis is storing the best path up to each state. Using dynamic programming, we can calculate the most probable path through the whole trellis [MS99] [Sar03]. E.g.:

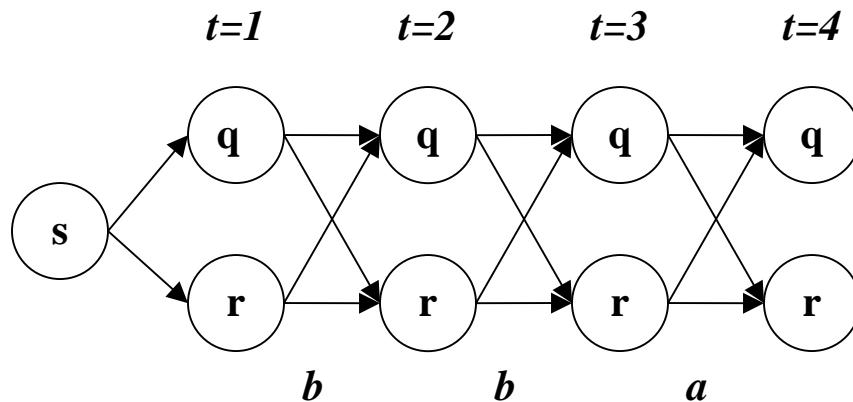


Figure 3 Example of Viterbi Algorithm.

The above trellis is an array of states against times. A node in the trellis can store information about state sequences up to this node. The lines show the possible transitions between states at each time step. Here we have a fully interconnected HMM where one can move from any state to any other at each step [MS99] (This is called having the ergodic property).

Viterbi algorithm can be easily applied to solve text chunking tasks. The running time complexity is the same as Viterbi algorithm mentioned before, $O(mn^2)$, while doing a full parsing with Context-Free Grammar takes at least $O(m^3)$, which is much slower, since the number of input words m is often much larger than the number of the states n .

Another reason is that they are one of a class of models for which there exist efficient methods of training through the use of the Expectation Maximization (EM) algorithm – this algorithm allows us to automatically learn the model parameters that best account for the observed data. In addition, we can use HMM in generating parameters for linear interpolation of n-gram models.

3.3 Data Representation

3.3.1 Inside/Outside

In 1995, Ramshaw and Marcus proposed to encode all chunks with 3 tags, **I**, **O** and **B** [RM95]. This representation enables to solve chunking problem as a trainable PoS tagging task.

I – for words inside a noun chunk




O – for words outside a noun chunk.

B – for the initial word of a noun phrase immediately follows another one.

Thereafter, [SV99] developed three variants based on the Ramshaw and Marcus representation. They named the variants IOB2, IOE1 and IOE2 and used IOB1 as the name for the Ramshaw and Marcus representation. IOB2 is the same as IOB1, except that every initial word of a noun phrase receives tag B. IOE1 differs from IOB1 in the fact that rather than the tag B, a tag E is used for the final word of a noun chunk, which is immediately followed by another noun phrase. IOE2 is a variant of IOE1 in which each final word of a noun phrase is tagged with E.

3.3.2 Start/End (O+C)

[SV99] showed that bracket representations **O** and **C** can also be regarded as two tagging representation with two streams of brackets, where tag **O**, *open bracket*, is for initial word of a chunk. Tag **C**, *close bracket*, is for final word of a chunk. Tag **.**, *period*, is for words outside of any chunk. After merging these two representations, we get a new  representation with 5 tags. This representation was renamed as *Start/End* data representation in [KM01]. In this project, we follow tag naming convention used in [KM01]. This representation is defined as follows.

B – for chunk-initial words

E – for chunk-final words

I – for chunk-inside words

S – for single word within a chunk

O – for words outside of any chunk

Table 4 contains example tag sequences for all five tag sequences for the example sentence.




Word	IOB1	IOB2	IOE1	IOE2	O+C
In	O	O	O	O	O
early	I	B	I	I	B
trading	I	I	I	E	E
in	O	O	O	O	O
Hong	I	B	I	I	B
Kong	I	I	E	E	E
Monday	B	B	I	E	S
	O	O	O	O	O
	I	B	I	E	S
was	O	O	O	O	O
quoted	O	O	O	O	O
at	O	O	O	O	O
\$	I	B	I	I	B
366.50	I	I	E	E	E
an	B	B	I	I	B
ounce	I	I	I	E	E
.	O	O	O	O	O

Table 4 The noun chunk tag sequences for the example sentence, *In early trading in Hong Kong Monday, gold was quoted at \$366.50 an ounce.*

3.4 Voting Techniques


Multiple data representations are interesting because a learner will make different errors when trained with data encoded in a different representation. This means we can improve nking performance with combination techniques.

[Hal98] explored five voting methods. They assign weights to the output of the individual systems and use these weights to determine the most probable output tag. Since the classifier generate different output formats, all classifier output has been converted to the O and the C representations. The simplest voting method assigns uniform weights and picks the tag that occurs most often - *Majority Voting*. A more advanced method is to use as a weight the accuracy of the classifier on some held-out part

of the training data, the tuning data - *TotPrecision*. One can also use the precision obtained by a classifier for a specific output value as a weight - *TagPrecision*. Alternatively, [TKS02] use as a weight a combination of the precision score for the output tag in combination with the recall score for competing tags - *Precision-Recall*. The most advanced voting method examines output values of pairs of classifiers and assigns weights to tags based on how often they appear with this pair in the tuning data - *TagPair* [Hal98].

[TKS02] showed system combination improved performance: the worst result of the combination techniques is still better than the best result of the individual systems. Furthermore, data encoded with Inside/Outside data representations trained by a learner leads to similar results, while data encoded with Start/End (O+C) data representation trained by a learner consistently obtains higher $F_{\beta=1}$ rates. They also found the performance differences among the different voting techniques are small. Thus, Majority Voting becomes attractive, since it is the simplest of the voting techniques

3.4.1 Majority Voting

When different machine learning systems are applied to the same task, they will make different errors. The combined results of these systems can be used for generating an analysis for the task that is usually better than that of any of the participating system. For example: suppose we have five different data representations, DR1-5, which assign y classes to patterns. Their output for five patterns, pattern1-5, is as follows:

	DR1	DR2	DR3	DR4	DR5	Correct
Pattern1	0	0	1	0	0	0
Pattern2	1	1	1	0	1	1
Pattern3	1	1	1	0	1	1
Pattern4	1	0	0	0	0	0
Pattern5	1	1	1	1	0	1

Table 5 Example of majority voting results among five data representations (DRs) .



Each of the five representations makes an error. We then use a combination of the five by choosing the class that has been predicted most frequently for each pattern. This means that we can train one learner with five data representations and obtain five different analyses of the data that we can combine with majority voting techniques. Thus different data representations can enable us to improve the performance of the chunker, and the combined results of these data representations (DRs) can be used for generating an analysis for the task that is usually better than that of any of the participating data representations. This approach will eliminate errors that made by a minority of the data representations. The table 5 showed that combined systems are usually better than single system.

3.5 Chapter Summary

This chapter introduced some **background information to our basic approach.**

Hidden Markov Models (HMMs) are useful when one can think of underlying events probabilistically generating surface events. HMM is as a language model: compute probability of given observation sequence, a *decoding* process, HMM is as a parser: compute the best sequence of states for a given observation sequence, a *training* process, and HMM is as a learner: given a set of observation sequences, learn its distribution. One

application of HMMs is **Viterbi algorithm**, used to find the most likely complete path through a trellis by dynamic programming in linear time to its input.

Data representations (DRs) for text chunking are divided into two subgroups: **Inside/Outside (IOB1/IOB2/IOE1/IOE2)** and **Start/End (O+C)**. Inside/Outside DRs have three chunk tags each, where *I* for words inside a chunk, *B* for words starting a chunk, *O* for words outside any chunk, *E* for words ending a chunk. Start/End DR has five chunk tags, including all the tags in Inside/Outside, In addition, *S* for single word inside a chunk.

Two voting techniques is described in this chapter: simple **Majority Voting**, assigning equal weights to all representations, or various **Weighted Voting**, assigning different weights to each representation based on different conditions.

CHAPTER FOUR: TEXT CHUNKING APPROACH

Our approach is based on two ideas. First, solving chunking tasks as a PoS tagging problem for each data representation based on Specialized Hidden Markov Model (HMM) developed by [MP02]. Second, voting between multiple data representations.



4.1 Specialized HMM Chunking

For each individual data representation, we followed the approach of [MP02]. They considered text chunking to be a tagging problem and then solved tagging as a maximization problem.

Let \mathbf{O} be a set of output tags and \mathbf{I} the input vocabulary of the application. Given an input sentence $I = i_1, \dots, i_T$, where $i_j \in \mathbf{I}: \forall j$, the process consists of finding the sequence of states of maximum probability on the model. That is, the sequence of output tags, $O = o_1, \dots, o_T$, where $o_j \in \mathbf{O}: \forall j$. This process can be formalized as follows:

$$\hat{O} = \arg \max_o P(O | I) = \arg \max_o \left(\frac{P(O) \cdot P(I | O)}{P(I)} \right); O \in \mathbf{O} \quad (1)$$

Due to the fact that this maximization process is independent of the sequence, and taking into account the Markov assumptions, the problem is reduced to solving the following equation (2):

$$\arg \max_{o_1 \dots o_T} \left(\prod_{j=1 \dots T} P(o_j | o_{j-1}, o_{j-2}) \cdot P(i_j | o_j) \right) \quad (2)$$

The parameters of equation (2) can be represented as a *second-order HMM* whose states correspond to a tag pair. In a *first-order HMM* or simply *HMM*, each state corresponds to one tag, which means predicting current tag only depends on the previous tag. In second-order HMM, predicting the current tag depends on the previous two tags. Contextual probabilities, $P(o_j | o_{j-1}, o_{j-2})$, represent the transition probabilities between states and $P(i_j | o_j)$ represents the output probabilities.

The formalism has been widely used to efficiently solve part-of-speech (PoS) tagging in ([Chu88], [Mer94], [Bra00]), etc. In PoS tagging, the input vocabulary is composed of words and the output tags are PoS or morphosyntactic tags. The segmentation produced by some different shallow parsing tasks, such as text chunking or clause identification, can be represented as a sequence of tags as mentioned above. Therefore, these problems can also be carried out in a way similar to PoS tagging.

PoS tagging considers only words in the input. In contrast, chunking can consider words and PoS tags. However, if all this input information is considered, the input vocabulary of the application could become very large, and the model would be poorly estimated.

On the other hand, in order to avoid generating an inaccurate model due to the generic output tag set. They consider a more fine-grained output tag set by adding lexical and PoS information to the output tags. This aspect has also been tackled in PoS tagging ([KLR99], [LTR00], [PM01]), by lexicalizing the models, that is, by incorporating words into the contextual model.

In the work, they proposed a simple function that encoded the given data format into the model without changing the learning or the tagging processes. This method consists of modifying the original training data set in order to consider only the relevant lexical and POS information and to extend the output tags with additional information, since adding all the words leads to poor performance and no improve at all.

This transformation is the result of applying a specialization function f_s on the original training set to produce a new one. This function transforms every training tuple $\langle w_i, p_i, ch_i \rangle$ to a new tuple $\langle p_i, ch_i \rangle$ or $\langle p_i \cdot ch_i, w_i \cdot p_i \cdot ch_i \rangle$. That is, only a set of certain relevant words belong to certain lexical set (W_s) were considered in the contextual language model and defined the following specialization function:

$$f_s(\langle w_i \cdot p_i, ch_i \rangle) = \begin{cases} \langle w_i \cdot p_i, w_i \cdot p_i \cdot ch_i \rangle & \text{if } w_i \in W_s \\ \langle p_i, p_i \cdot ch_i \rangle & \text{if } w_i \notin W_s \end{cases}$$

Input			Output = f_s (Input)	
w_i	p_i	ch_i	p_i or $w_i \cdot p_i$	$p_i \cdot ch_i$ or $w_i \cdot p_i \cdot ch_i$
you	PRP	B-NP	PRP	PRP-B-NP
will	MD	B-VP	MD	MD-B-VP
start	VB	I-VP	VB	VB-I-VP
to	TO	I-VP	TO	TO-I-VP
see	VB	I-VP	VB	VB-I-VP
shows	NNS	B-NP	NNS	NNS-B-NP
where	WRB	B-ADVP	where-WRB	where-WRB-B-ADVP
viewers	NNS	B-NP	NNS	NNS-B-NP
program	VBP	B-VP	VBP	VBP-B-VP
the	DT	B-NP	DT	DT-B-NP
program	NN	I-NP	NN	NN-I-NP

Table 6 Example of specialization where the words belong to the predefined lexical set W_s .



Table 6 shows an example of the application of this function on a sample of the training set used in the chunking task, where symbol “-” is a connection symbol used to connect between words, PoS, or chunk type. For example, the tuple $\langle You - PRP, B - NP \rangle$ is transformed to the new tuple $\langle PRP, PRP - B - NP \rangle$, considering only POS information. On the other hand, the tuple $\langle where, WRB, B - ADVP \rangle$, considering also lexical information, is transformed to the new tuple $\langle where - WRB, where - WRB - B - ADVP \rangle$.

From this new training set, we can learn the Specialized HMM by maximum likelihood in the usual way. The tagging process is carried out by Dynamic Programming Decoding using the Viterbi algorithm. This decoding process is not touched. Thus, the only thing we need to worry about is the decisions taken into account in the specialization process. That is, to consider the relevant information in the output of specialization.

The model is called **SP** if only the part-of-speech tag is involved in specialization, while the model is called **SP+Lex-XXX** if there is more lexical information involved based on some lexical rule **Lex-XXX** defined by [MP02].

The selection of the set W_s produces various kinds of lexicalized HMM models. To generate lexical set **Lex-WTE**, we use a development set consisting of a heldout or deleted set of 10% from the training set in order to pick elements for W_s . The heldout set consists of every 10th sentence. The remaining set is used as the training data.

We used the following lexical specialization models defined by [MP02].

- **Lex-WHE:** W_s contains the words whose frequency in the training set was higher than a certain *threshold*. In order to determine which threshold maximized the performance of the model (that is, the best set of words to specialize the model), we tuned it on the development partition with word sets of different sizes. The threshold obtained in my experiments was 100.
- **Lex-WCH:** W_s contains the words that belong to certain chunk types with higher frequency threshold. In our work, we pick chunk types NP,VP,PP and ADVP with a threshold of 50.
- **Lex-WTE:** W_s contains the words whose chunk tagging error rate was higher than a certain threshold in development set. Based on the experiments in [MP02], we pick a threshold of 2.

The experiments in [MP02] showed that specialization can improve performance considerably. By combining the **Lex-WCH** and **Lex-WTE** conditions, the output tag set increases from the original set of 22 to 1341, with 225 words being used as lexical material in the model and the accuracy on the CoNLL-2000 data increases to 92.19% using exactly the same trigram-based HMM model.

4.2 Voting Between Multiple Data Representations

The notion of specialization is a good example of how the data representation can lead to higher accuracy. We extend this idea further by voting between multiple specialized data representations.

The model we evaluate in this paper is simple majority voting on the output of various specialized HMM models (described above). The HMM model is trained on

different data representations, and the test data is decoded by each model. The output on the test data is converted into a single data representation, and the final label on the test data is produced by a majority vote.

We experimented with various weighted voting schemes -- setting weights for different representations based on accuracy on the heldout set and using a *don't care* tag to ignore certain chunk type for certain data representations. However, no weighting scheme provided us with a significant increase in accuracy over simple majority voting.

4.3 Chapter Summary

This chapter introduced the chunking approach used in this project. It is divided into two parts: **Specialized HMM Chunking** and **Majority Voting**.

- In **Specialized HMM Chunking** process, for each data representation, we first apply a specialization function f_s on the original training set and outputs a new one and then use this output to train a trigram HMM model. This function transforms every training tuple $\langle w_i, p_i, ch_i \rangle$ to a new tuple $\langle p_i, ch_i \rangle$ or $\langle p_i \cdot ch_i, w_i \cdot p_i \cdot ch_i \rangle$. That is, only a set of certain relevant words belong to certain lexical set (W_s) were considered in the contextual language model and defined the following specialization function:

$$f_s(\langle w_i \cdot p_i, ch_i \rangle) = \begin{cases} \langle w_i \cdot p_i, w_i \cdot p_i \cdot ch_i \rangle & \text{if } w_i \in W_s \\ \langle p_i, p_i \cdot ch_i \rangle & \text{if } w_i \notin W_s \end{cases}$$

- In **Majority Voting** process, we simply take equal weight vote among all these five data representations (IOB1/IOB2/IOE1/IOE2/O+C).

CHAPTER FIVE: CHUNKING EVALUATION

Chapter four introduced how specialized HMM approach with voting between multiple data representations works. In this Chapter, we will test this approach on the CoNLL-2000 dataset and Base NP dataset, then we compare our results with the ones of other major approaches.



In first experiment, arbitrary phrase chunking (or *text chunking*), to obtain various data representations, we convert the corpus in IOB2 format into the other four data representations (IOB1, IOE1, IOE2 and O+C), where O+C is Start/End representation, and then convert each data representation into the format defined by the specialized HMM approach.

We tested various specialization models discussed in last chapter. In addition, to generate a specialized model – SP+Lex-WHE, we split original training set into a new training set (90% of the original training set) and a development set (10% of the original training set) for each data representation. Also, to keep the sentence distribution, we set ten sentences as a unit, and for each unit, put first nine sentences into the new training set and the last one into the development set. Now we are ready to chunk.

Once we have all five different data representations chunked, we start to use majority voting technique to combine them into one file. In order to evaluate the accuracy, we have to transform the result into the CoNLL-2000 output format. Then, we

remove the enriched information from the output and convert the results into the CoNLL-2000 output format – IOB2.

In second experiment, Noun Phrase Chunking (*or Base NP chunking*), the process is similar as the previous one, except that the original chunk representation is in IOB1 format, thus we convert it into other four data representations and finally evaluate the result in this format.

5.1 Dataset

5.1.1 Arbitrary Chunking Dataset (CoNLL-2000 Dataset)

We used the dataset defined in the shared task of CoNLL-2000. The characteristics of this task were described by [TB00]. It used the same Wall Street Journal (WSJ) corpus sections defined by [RM95]. The set of chunks was derived from the full parsing taking into account certain assumptions and simplification. The PoS tagging was obtained using the Brill tagger without correcting the tagger output.

5.1.2 Base NP Dataset (Base NP Chunking Dataset)

Base NP dataset is defined in [NP02] and was first introduced by Ramshaw and Marcus [RM95]. Same as the text chunking dataset, it consists of section 15-18 of WSJ of the Penn Treebank as training data, section 20 of that as test data, and the PoS tagging was obtained using the Brill tagger. However, the correct chunk format is defined in IOB1 representation instead of IOB2 in chunking dataset of CoNLL-2000 shared task.

5.2 TnT Tagger

All the training and tagging tasks were conducted by using the TnT tagger developed by [Bra00] without making any modification to its source code.

TnT, developed by Brants, is an efficient statistical part-of-speech tagger independent of language, domain and tagset. TnT is the short form of *Trigrams'n Tags*. The component for parameter generation trains on tagged corpora.

The TnT tagger is an implementation of the Viterbi algorithm for second order Markov models. The main paradigm used for smoothing is linear interpolation, the respective weights are determined by deleted interpolation. Unknown words are handled by a suffix trie and successive abstraction. In [MP02], they used this smoothing setting in their chunking process. In order to compare with their results, we will follow this setting in our experiments. The programs are run under Solaris in ANSI C using the GNC C compiler.

TnT comes with three models, one for German and two for English. The German model is trained on the *Saarbrücker*, German newspaper corpus, using the *Stuttgart-Tübingen-Tagset*. The English model are trained on the *Susanne Corpus* and the *Penn Treebank* respectively [Bra00].

5.2.1 File Formats

There are four types of files: n-gram file, lexicon file, the untagged input file and the tagged output file. Optionally, the user can specify a mapping of tags used by the tagger to an output tagset.

Each line starting with two percentage signs (%%) marks a comments and is ignored by the programs. The tokens are encoded using all characters with codes 0x21...0xFF and each line with white space is ignored by the programs [Bra00].

Untagged format	Tagged format	
%% Brown Corpus	%% Brown Corpus	
%% File N11, Sent 3	%% File N11, Sent 3	
You	you	PRP
Will	will	MD
Start	start	VB
To	to	TO
See	see	VB
Shows	shows	NNS
Where	where	WRB
viewers	viewers	NNS
program	program	VBP
The	the	DT
program	program	NN
.	.	.

Table 7 Format of lexicon, untagged and tagged files.

Lexicon format					
%% Lexicon created from the Brown corpus					
%% ...					
thaw	6	NN	3	VB	3
thawed	3	VCN	3		
thawing	2	VBG	2		
The	625	DT	624	IN	1
theaf	1	NN	1		
%% ...					

Table 8 Format of lexicon files.

%% n-gram, Brown corpus				
%%				
NNP	62022			
NNP	CC	2888		
NNP	CC	CD	26	
NNP	CC	NN	76	
%% ...				
NNP	CD	912		
NNP	CD	CC	23	
NNP	CD	CD	7	
%% ...				
%% ...				

Table 9 Format of n-gram files.

5.2.2 Running TnT

The application of TnT consists of two steps [Bra00]:

- Parameter generation, creates the model from a tagged training corpus.
E.g.: `tnt-para [options] <corpusfile>`
- Tagging, applies the model to new text and performs the actual tagging.
E.g.: `tnt [options] model corpus`

5.2.3 Evaluation

TnT's performance is evaluated under following aspects [Bra00].

- Determining the averaged accuracy over ten iterations, overall accuracy and separate accuracies for known and unknown words are measured.
- Presenting learning curves to indicate the performance comparison with different corpus.
- Assigning tags to words with optional probabilities to rank different assignment.

5.3 Experimental Results

5.3.1 Text Chunking (Arbitrary Phrase Chunking)

In order to obtain various data representations, we converted the corpus in IOB2 format into other four data representations: IOB1, IOE1, IOE2 and O+C. We then converted each data representation into the format defined by specialized HMM approach.

In the results shown in this section,

- *SP* represents specialized HMM approach without lexical information.

- *SP+Lex-WCH* represents specialized HMM approach with lexical information defined based on Lex-WCH.
- *5DR* represents five data representations (DR), which is IOB1, IOB2, IOE1, IOE2, O+C and we pick O+C as the default DR.
- *3DR* represents IOB1, IOB2, IOE1 and we pick IOB2 as the default DR.
- Majority represents majority voting.

Table 10 gives the text chunking (arbitrary phrase chunking) results for each setting. Table 11 and 12 shows the results of the specialized model *SP+Lex-WCH* in IOB2 and IOE2 evaluation format respectively, where *all* represents the results obtained after 3DR or 5DR majority voting respectively. Our results show the performance of 5DR voting is better than 3DR voting.

Specialization criteria	Precision(%)	Recall(%)	$F_{\beta=1}$
Baseline	72.58	82.14	77.07
Trigram HMM (no words)	84.31	84.35	84.33
SP	89.57	89.54	89.56
SP+Lex-WTE (3DR, Majority)	92.49	93.00	92.75
SP+Lex-WCH (3DR, Majority)	93.54	92.97	93.25
SP+Lex-WCH (5DR, Majority)	93.89	94.12	94.01

Table 10 Text chunking results for each setting.

Chunk type	Precision(%)	Recall(%)	$F_{\beta=1}$
ADJP	75.54	71.92	73.68
ADVP	80.80	79.21	80.00
CONJP	60.00	66.67	63.16
INTJ	50.00	50.00	50.00
NP	95.46	95.67	95.57
PP	97.69	96.61	97.15
PRT	66.02	64.15	65.07
SBAR	77.25	85.05	80.96
VP	92.69	94.16	93.42
All	93.89	94.12	94.01

Table 11 Text chunking results of 5DR majority voting with SP+Lex-WCH in IOB2 format.

Chunk type	Precision(%)	Recall(%)	$F_{\beta=1}$
ADJP	77.94	71.00	74.31
ADVP	80.12	78.18	79.14
CONJP	66.67	66.67	66.67
INTJ	50.00	50.00	50.00
NP	94.85	94.03	94.44
PP	97.52	96.47	96.99
PRT	64.29	59.43	61.76
SBAR	76.11	83.36	79.57
VP	92.65	93.39	93.02
all	93.54	92.97	93.25

Table 12 Text chunking results of 3DR majority voting with SP+Lex-WCH in IOB2 format.

In Table 13, we find when an Inside/Outside representation is converted into Start/End representation, the accuracy is increased and if we do the other way, the accuracy will decrease. [TKS00] also reported O+C (Start/End) obtained higher $F_{\beta=1}$ accuracy with high precision and lower recall and [XS03] presented a so called LMR tagging to solve Chinese word segmentation problem is another example Start/End representation improves the performance, since the role of LMR in Chinese word segment is just like that of Start/End in Text chunking. The reason is because Start/End representation with five tags catches more context information, while Inside/Outside

representation only has three tags. Hence, Start/End representation is more discriminative than the inside/outside representations.

Also we find even the difference among different representations within Inside/Outside representations is smaller than that with the Start/End representation, we still can observe the representation format conversion will affect the accuracy. Thus, picking a best result other than standard test format IOB2 as their final result is incorrect, since it is not comparable with other approaches. Moreover, the testing corpus should not be touched somehow. Converting the test corpus is not the right way to do the testing. Errors may be introduced and lead to incomparable results.

	IOB1	IOB2	IOE1	IOE2	O+C
IOB1	92.68	93.07	92.66	92.68	94.72
IOB2	92.82	92.63	92.82	92.82	94.47
IOE1	92.82	92.82	92.87	92.87	94.64
IOE2	92.53	92.53	92.53	92.53	94.43
O+C	92.45	92.45	92.49	92.35	94.28
3DR	93.03	93.25	92.82	93.07	94.92
5DR	93.92	93.76	93.90	94.01	95.05

Table 13 Text chunking accuracy for all DRs in five evaluation formats. Note each column represents the evaluation format and each row represents the training and testing format.

Table 14 and 15 give the final results in IOB2 and IOE2 respectively.

Voting format	Precision(%)	Recall(%)	$F_{\beta=1}$
IOB1	93.89	93.95	93.92
IOB2	93.69	93.82	93.76
IOE1	93.79	93.77	93.78
IOE2	93.89	94.12	94.01
O+C	93.84	93.98	93.91

Table 14 Text chunking accuracy for all DRs evaluated in IOB2 format. Note that voting format is the format when conducting majority voting, all the DRs are converted into this format.

Voting format	Precision(%)	Recall(%)	$F_{\beta=1}$
IOB1	93.81	93.79	93.80
IOB2	93.69	93.82	93.76
IOE1	93.87	93.93	93.90
IOE2	93.89	94.12	94.01
O+C	93.84	94.00	93.92

Table 15 Text chunking accuracy for all DRs evaluated in IOE1 format.

5.3.2 Base NP Chunking (Noun Phrase Chunking)

We first convert dataset in IOB1 format into IOB2/IOE1/IOE2/O+C. For each representation, we perform specialization based on lexical rule SP+Lex-WCH before learning process. Finally, we apply voting in each format and evaluate it in IOB1 format.

Table 16 shows the final results in IOB1 representation after 5DR voting. However, some other experiment results are obtained in other representations. For example, in [KM01], they picked IOB2 as their final evaluation representation. We know there is no significant difference between IOB1 and IOB2, but the chunk representation of original training and test data from [NP02] is defined in IOB1, thus we decide to pick IOB1 as our final evaluation representation.

Voting format	Precision(%)	Recall(%)	$F_{\beta=1}$
IOB1	95.11	95.35	95.23
IOB2	95.05	95.34	95.19
IOE1	94.96	95.11	95.04
IOE2	94.96	95.21	95.08
O+C	95.04	95.30	95.17

Table 16 Base NP chunking accuracy for all DRs evaluated in IOB1 format.

5.4 Results Comparison

5.4.1 Text Chunking Comparison

Table 17 compares the results with other major approaches. We achieved 94.01 on $F_{\beta=1}$ score for both formats, which is slightly higher than [KM01], but still lower than [ZDJ02] in Table 17. However, [ZDJ02] used a full parser, detailed in Section 2.5, which we do not use in our experiments.

Approach	$F_{\beta=1}$
Generalized Winnow w/ full parser [ZDJ02]	94.17
Specialized HMM w/voting between multiple DR	94.01
SVM w/voting between multiple DR [KM01]	93.91
Generalized Winnow w/o full parser [ZDJ02]	93.57
WPDV w/voting between multiple models [Hal00]	93.32
MBL w/voting between multiple models [TKS00]	92.50
Specialized HMM [MP02]	92.19

Table 17 Comparison of text chunking accuracy with major approaches.

The above table shows the text chunking results rank on CoNLL-2000 dataset. All approaches, except [Hal00] and [TKS00], used a single learner and among of them, those approaches with voting between multiple data representations obtained better results than other approaches ([ZDJ02] is the only exceptional, since they used a full parser, which is not comparable with other's). The reason single learner with voting between multiple data representation performs better than that without voting is obvious, since voting between multiple data representations can correct minority errors. Moreover, the above results also showed single learner with voting between multiple data representations seems better than voting between multiple learning models. The reason why voting works is because the partners involved in voting are information compensable. Also, we know creating multiple syntactically complementary data representations is much easier than developing multiple complementary learning models. Hence, our finding -- a single

learner with voting between multiple data representations outperforms voting between multiple learning models seems reasonable.

5.4.2 Base NP Chunking Comparison

Table 18 compares the Base NP chunking results with other major approaches.

We achieved 95.23 on $F_{\beta=1}$ score, which is the best state-of-the-art score so far.

Approach	$F_{\beta=1}$
Specialized HMM w/voting between multiple DR	95.23
SVM w/voting between multiple DR [KM01]	94.22
Voting between multiple learning model[TDD+00]	93.86
Voting between multiple learning model[TKS00]	93.26

Table 18 Comparison of Base NP chunking accuracy with major approaches.

[SP02] developed a discriminative model, conditional random field (CRF). Like other discriminative models, such as [KM01], they also involve position features between tags and words. To remove the overfitting, they used an addition development set (Section 21 of WSJ) to tune the results. Strictly speaking, their dataset is not the same as the other Base NP chunking approaches, since section 21 is not a part of standard Base NP chunking dataset. They achieved 94.38 on $F_{\beta=1}$ score. In [SP02], they do not indicate if they apply a Base NP chunking or an arbitrary phrase chunking. We guess they take the arbitrary chunking process, since they pick CoNLL-2000 dataset and compare their NP results with others extracted from arbitrary phrase chunking approaches. Thus, we consider their score is not comparable with our Base NP results, but their results are comparable with an even higher $F_{\beta=1}$ score, 95.57, which we obtained through an arbitrary chunking process in the previous section. In addition, we find NP F-score in an arbitrary phrase chunking is slightly higher than that in a standard Base NP chunking and

[KM01] has the same phenomenon. We can easily explain this phenomenon from a classification perspective. That is, an arbitrary phrase chunking model (multiple-class model) describes the dataset more accurate than a Base NP chunking model (two-class model). This is obviously true if we look at the syntactic content of the dataset. However, we clearly know this change is not significant.

5.4.3 Comparison with Kudo’s Approach

The common parts between our approach and Kudo’s approach are that we all train a statistical learner with voting between multiple data representations, treat chunk tagging task as a sequence learning problem, and achieve equal state-of-the-art performance. The differences are that our approach uses a simpler learner based on specialized HMM, which runs linear time on input words, while [KM01] trains eight different SVM classifiers, the algorithm requires $k \times (k - 1) / 2$ classifiers considering all pairs of k classes. Each SVM training uses a quadratic programming step. Secondly, we apply simple majority voting between five data representations (Inside/Outside and Start/End), while Kudo’s approach only apply weighted voting between Inside/Outside representations, since their learner restricted them to vote between different data representation types. In our experiments, we find Start/End representation usually catch more information than that of Inside/Outside representations and in turn improve our performance.

To examine the assumption that our approach and [KM01] are different is valid, we applied *McNemar Test* and assumed the errors are independent. The distributions of the errors for arbitrary phrase chunking (text chunking) and Noun phrase chunking (Base NP chunking) are listed in the following Table 19 and 20

		[KM01]		Total
		Correct	Incorrect	
SP+Lex-WCH w/ voting	Correct	22093 (n_{00})	356 (n_{01})	22449
	Incorrect	309 (n_{10})	1094 (n_{11})	1403
Total		22402	1450	23852

Table 19 Text chunking Error distribution between SP+Lex-WCH w/voting and [KM01].

		[KM01]		Total
		Correct	Incorrect	
SP+Lex-WCH w/ voting	Correct	11557 (n_{00})	202 (n_{01})	11759
	Incorrect	74 (n_{10})	491 (n_{11})	565
Total		11631	693	12324

Table 20 Base NP chunking Error distribution between SP+Lex-WCH w/voting and [KM01].

With this distribution, we can directly compute a 2-tailed P value based on the following formula defined in [GC89],

$$P = 2 \sum_{m=0}^{n_{10}} \binom{k}{m} \left(\frac{1}{2}\right)^k, \text{ where } k = n_{10} + n_{01}$$

Task	Null hypothesis	p-value
Arbitrary Chunking	Specialized HMM w/voting vs. [KM01]	0.0745
Base NP Chunking	Specialized HMM w/voting vs. [KM01]	<0.001

Table 21 McNemar's test between Specialized HMM w/ voting and [KM01] on two chunking tasks.

In arbitrary phrase chunking task, we have to say the difference between our approach and [KM01] is not statistically significant, which means this two approaches have the equal performance on the arbitrary phrase chunking task, while Table 21 shows Specialized HMM w/voting and [KM01] are significantly different on Base NP chunking task.

Moreover, our approach is much faster. As showed in figure 4, our approach takes more than 15 times faster than Kudo’s approach for a data representation chunking process. Furthermore, [KM01] trained eight SVM classifiers, thus it took much longer time overall. Thus, our simple voting system is considerably faster and produces comparable results.

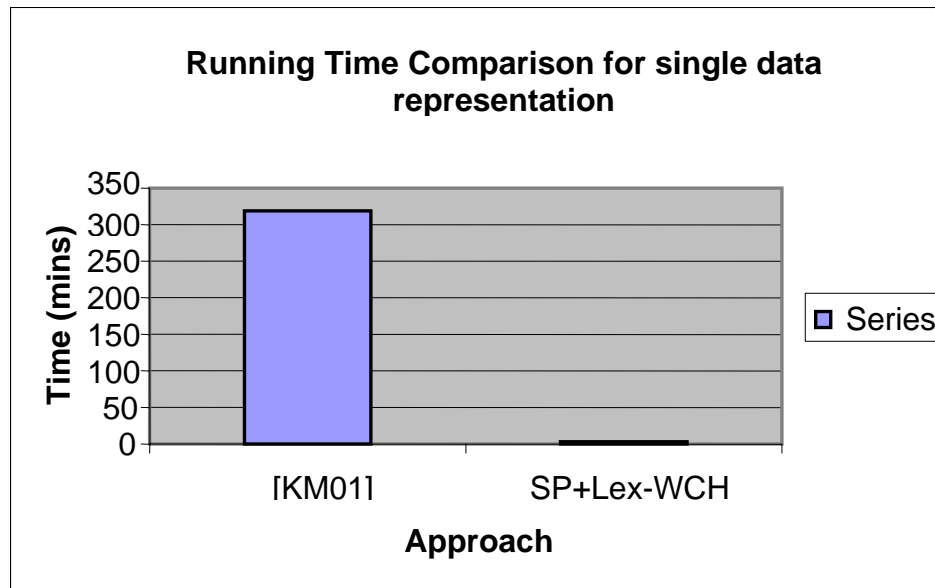


Figure 4 Running time comparison for single data representation between SP+Lex-WCH and [KM01] on arbitrary chunking task.

5.5 Analysis

Previous approaches that use voting have all used voting as a means of system combination, i.e. taking multiple machine learning methods and taking a multiple machine learning methods and taking a majority vote or weighted vote combining their output [TKS00]. This kind of system combination can be done using voting or stacking.

Voting as system combination has been applied to CoNLL-2000 data set as well: [Hal00] obtains an $F_{\beta=1}$ of 93.92. [TKS02] combines the output of several systems but also does voting by exploiting different data representations. However, to our knowledge, there has not been a study of voting purely between multiple data representations using a single machine learning method. Our results seem to indicate that even simple majority voting between multiple data representations does better than voting for system combination.

Superficially, it seems that [KM01] also does voting on multiple data representations. However, the multiple data representations are only used to discover which representation works better with the SVM classifier. The approach uses multiple classifiers (as stated earlier), in order to enable multi-class classification using a two-class SVM classifier (instead of using error-correcting codes or other methods for multi-class classification). This is quite different from voting between multiple data representations.

[SFB+98] provide some insight into the power of voting by stating that voting between multiple representations can be seen as a form of smoothing over a hidden posterior distribution over the true labels in the test data. If we see voting as being a smoothing method, one way we can choose an appropriate representation to participate in voting is to check the goodness of the representation based on methods used to evaluate smoothing methods in language modelling. A representation can be added if the addition reduces perplexity on the training set.

There is another theoretical approach that can be used to analyze this result: the bias-variance tradeoff could be used to discover that the multiple representations are all increasing bias while reducing variance in the labelling task.

We plan to explore these alternate analysis methods as well in the near future.

5.6 Chapter Summary

This chapter has described the approach experiments and results. The following are some of the highlights of this chapter.

The **Dataset** is the part of the Wall Street Journal corpus (WSJ), section 15-18 as training data (211727 tokens) and section 20 as test data (47377 tokens), defined in the shared task of CoNLL-2000.

All the training and tagging tasks were conducted by using **TnT tagger** developed by [Bra00] without making any modification. TnT is an implementation of Viterbi algorithm for second order Markov models.

We achieved **94.01** in $F_{\beta=1}$ score on arbitrary phrase chunking and slightly higher than **93.91** obtained by [KM01], the second best result. Based on **McNemar Test**, we have to say this slight difference is not statistical significant. Additionally, they trained eight different SVM classifiers and took considerable longer time than our approach. The best result is achieved by [ZDJ02], they obtained **94.17** with a full parser. Since we do not have this knowledge, we consider their result is not comparable with ours. Without a full parser, they obtained **93.57**.

In addition, we obtained **95.23** in $F_{\beta=1}$ score on the Base NP chunking task and our score is higher than the current best score **94.22** obtained by [KM01]. By the paired McNemar test, our approach vs. [KM01], we showed this difference is significant.

To our knowledge, there has not been a study of voting purely between multiple data representations using a single machine learning method. Our results seem to indicate

that even *simple majority voting between multiple data representations does better than voting for system combination.*

CHAPTER SIX: CONCLUSION

The main contribution of this study is that a single learning method, a simple trigram HMM can use voting between multiple data representations to obtain results equal to the best on the CoNLL-2000 text chunking data set. Using no additional knowledge sources, we achieved 94.01 $F_{\beta=1}$ score on arbitrary phrase chunking compared to the previous best comparable score of 93.91. Based on the McNemar test, we find the difference between our approach with the comparable state-of-the-art approach is not statistical significant, which means we have the equal performance on the CoNLL-2000 dataset. Secondly, we achieved 95.23 $F_{\beta=1}$ on the Base NP chunking, which is better than the current comparable state-of-the-art score of 94.22 and we showed our approach is significantly different with the current comparable state-of-the-art approach on the Base NP chunking task by the McNemar test. In addition, our text chunker is considerably faster than comparably accurate methods in training as well as in decoding.

CHAPTER SEVEN: FUTURE WORK

[SFB+98] provided some insight into the power of voting by starting that voting between multiple data representations can be seen as a form of smoothing. Hence, we can try to choose an appropriate representation based on its perplexity on the training set.

Secondly, the bias-variance tradeoff could be used to discover that the multiple representations are all increasing bias while reducing variance in the labelling task.

Lastly, our research also shows voting between multiple data representations can improve performance and we can continue to improve it if we have more data representations. We may recursively create more data representations to capture more context information. In the following example, we create a new representation by considering the previous word information for each word tagged as *O*. However, we are not sure if the new created data representations are syntactically meaningful.

He is the man .	
B O B I O	(IOB2)
B O-B B I O-I	(A new representation)

Figure 5 Example of a new representation.



BIBLIOGRAPHY

- [Abn91] S. Abney. Parsing by Chunks. In Principle-Based Parsing. *Kluwer Academic Publishers, 1991.*
- [ADK98] S. Argamon, I. Dagan and Y. Krymolowsky. A Memory-Based Approach to Learning Shallow Natural Language Patterns. In *Proceedings of 36th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 67-73, Montreal, Canada, 1998.
- [ATH03] Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden Markov Support Vector Machines. In *Proceedings of the 20th International Conference on Machine Learning: ICML 2003, 2003*
- [BD01] S. Buchholz and W. Daelemans. Complex Answers: A Case Study using a WWW Question Answering System. In *Proceedings of Natural Language Engineering, 2001.*
- [BFK+95] A. Bies, M. Ferguson, K. Katz, R. Macintyre. *Bracketing Guidelines for Treebank II Style Penn Treebank Project. 1995.*
- [Bra99] T. Brants. Cascaded Markov Models. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL-99)*, Bergen, Norway, 1999.
- [Bra00] T. Brants. TnT – a statistical part-of-speech tagger. In *Proceedings of the 6th Applied Natural Language Processing Conference: ANLP-2000, Seattle, WA, 2000.*
- [BVD99] S. Buchholz, J. Veenstra and W. Daelemans. Cascaded Grammatical Relation assignment. In *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, 1999.*
- [Chu88] K. W. Church. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In *Second Conference on Applied Natural language Processing, pp 136-143. Austin, Texas, 1988.*
- [Col96] Machael John Collins. A new statistical parser based on bigram lexical dependencies. In *34th Annual Meeting of the Association for Computational Linguistics. University of California, Santa Cruz, California, USA, June, 1996.*
- [Col02] M. Collins. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of EMNLP and ACL, 2002.*
- [CoN03] CoNLL URL. <http://cnts.uia.ac.be/conll2000/chunking/> and <http://cnts.uia.ac.be/conll2003/ner/>, 2003

- [CP98] C. Cardie and D. Pierce. Error-Driven Pruning of Treebank Grammars for Base Noun Phrase Identification. In *Proceedings of COLING/ACL*, pp 218-224, Montreal, Canada, 1998.
- [Dej00] H. Déjean, Learning Syntactic Structures with XML. In: *Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal, 2000*.
- [GC89] L.Gillick and S.J.Cox. Some Statistical Issues in The Comparison of Speech Recognition Algorithms. In *Proceedings of Acoustics, Speech, and Signal Processing, 1989. pp. 532 - 535 vol.1. ICASSP-89., 1989 International Conference on , 1989*.
- [Hal98] H. V. Halteren. "Improving data driven wordclass tagging by system combination," In: COLING-ACL'98. In *Proceedings of the Conference. Vol. 1: 491-497.1998*.
- [Hal00] H. Halteren, Chunking with WPDV Models. In: *Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal, 2000*.
- [HOA+02] J. Hammerton, M. Osborne, S. Armstrong, and W. Daelemans. Introduction to Special Issue on Machine Learning Approaches to Shallow Parsing. In *Journal of Machine Learning Research 2*, pp. 551-558, 2002.
- [Joh00] C. Johansson, A Context Sensitive Maximum Likelihood Approach to Chunking. In: *Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal, 2000*.
- [KLR99] J.D. Kim, S.Z. Lee, and H.C. Rim. HMM Specialization with Selective Lexicalization. In *Proceedings of the join SIGDAT Conference on empirical Methods in Natural Language Processing of Very Large Corpora (EMNLP-VLC-99), 1999*.
- [KM01] T. Kudo and Y. Matsumoto. Chunking with support vector machines. In *Proceedings of the 2nd Meeting of the North American Association for Computational Linguistics: NAACL, 2001*.
- [Koe00] R. Koeling, Chunking with Maximum Entropy Models. In: *Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal, 2000*.
- [LTR00] S. Lee, J. Tsujii, and H. Rim. Lexicalized Hidden Markov Models for Part-of-Speech Tagging. In *Proceedings of 18th International Conference on Computational Linguistics, Saarbrücken, Germany, August, 2000*.
- [Meg01a] B. Megyesi. Phrasal Parsing by Using Data-Driven PoS Taggers. In *Proceedings of Recent Advances in Natural Language Processing (EuroConference RANLP-2001), Tzigov Chark, Bulgaria, September, 2001*.
- [Meg02] B. Megyesi. Shallow Parsing with PoS Taggers and Linguistic Features. In *Journal of Machine Learning Research 2*, pp. 639-668, 2002.
- [Mer94] B. Merialdo. Tagging English Text with a Probabilistic Model. In *Proceedings of Computational Linguistics, 20(2):155-171, 1994*.

- [MP02] A. Molina and F. Pla. Shallow Parsing using Specialized HMMs. *In Journal of Machine Learning Research, volume 2, pp. 595-613, Match, 2002.*
- [MPR+99] M. Muñoz, V. Punyakanok, D. Roth and D. Zimak, A Learning Approach to Shallow Parsing, In: "Proceedings of EMNLP/WVLC-99", University of Maryland, MD, USA, 1999.
- [MS99] C. D. Manning and H. Schutze. Foundations of Statistical Natural Language Processing by 680 pages 1 edition, *M.I.T. Press/Triliteral, ISBN: 0262133601, 1999.*
- [NP02] NP Chunking URL. <http://lcg-www.uia.ac.be/~erikt/research/np-chunking.html>, 2002.
- [Osb00] Miles Osborne. Shallow Parsing as Part-of-Speech Tagging. In *Proceedings of CoNLL-2000 and LLL-2000, pp. 145-147, Lisbon, Portugal, 2000.*
- [PM01] F. Pla and A. Molina. Part-of-Speech Tagging with Lexicalized HMM. In *Proceedings of International Conference on recent Advances in Natural Language Processing (RANLP2001), Tzigov Chark, Bulgaria, September 2001.*
- [PMP00] F. Pla, A. Molina and N. Prieto, Improving Chunking by Means of Lexical-Contextual Information in Statistical Language Models. In: *Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal, 2000.*
- [Rij79] C.J. van Rijsbergen, "Information Retrieval". *Buttersworth, 1979.*
- [RM95] L. Ramshaw and M. Marcus. Text Chunking Using Transformation-Based Learning. In *Proceedings of the 3ird ACL Workshop on Very Large Corpora: WVLC-1995, Cambridge, USA, 1995*
- [San90] B. Santorini. Part-of-Speech Tagging Guidelines for the Penn Treebank Project(3rd Revision, 2nd printing), 1990.
- [Sar03] Anoop Sarkar. CMPT-825, Natural Language Processing, Course Notes, *Simon Fraser University, Fall 2003.*
- [SB98] W. Skut and T. Brants. Chunk Tagger Statistical Recognition of Noun Phrases. In *ESSLLI-98 Workshop on Automated Acquisition of Syntax and Parsing (ESSLLI-98), Saarbrückhen, Germany, 1998.*
- [SFB+98] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A New Explanation for the Effectiveness of Voting Methods. *The Annals of Statistics, 26(5):1651-1686, 1998.*
- [SL99] R. Srihari and W. Li. Information extraction supported question answering. In *Proceedings of TREC 8, 1999.*
- [SP03] F. Sha and F. Pereira. Shallow Parsing with Conditional Random Fields. *In Proceedings of Technical Report CIS TR MS-CIS-02-35, University of Pennsylvania, 2003.*

- [SPT98] T. Sekimizu, H. Park, and J. Tsujii. Identifying the interaction between genes and gene products based on frequently seen verbs in medline abstracts. In *Genome Informatics*, pp. 62-71. Universal Academy Press, Inc, 1998.
- [SS00] E. F. Tjong Kim Sang and Sabine Buchholz. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proceedings of Conference on Computational Natural Language Learning: CoNLL-2000*, pages 127-132, Lisbon, Portugal, 2000
- [SV99] E. F. Tjong Kim Sang and Jorn Veenstra. 1999. Representing Texting Chunks. In *Proceedings of the 7th Conference of the European Association for Computational Linguistics: EACL-1999*, pp. 173-179, Bergen, Norway, 1999.
- [TB00] Erik F. Tjong Kim Sang and Sabine Buchholz, Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal, 2000*.
- [TDD+00] E. F. Tjong Kim Sang, W. Daelemans, H. Déjean, R. Koeling, Y. Krymolowski, V. Punyakanok and D. Roth, Applying System Combination to Base Noun Phrase Identification. In *Proceedings of COLING 2000, Saarbrücken, Germany, 2000*.
- [TKS00] E. F. Tjong Kim Sang. Text Chunking by System Combination. In *Proceedings of Conference on Computational Natural Language Learning: CoNLL-2000*, pp. 151-153. Lisbon, Portugal, 2000.
- [TKS02] E. F. Tjong Kim Sang. Memory-Based Shallow Parsing. In *Journal of Machine Learning Research*, volume 2, pp. 559-594, 2002.
- [TV99] E. F. Tjong Kim Sang and J. Veenstra, Representing Text Chunks. In *Proceedings of EACL'99, Bergen, Norway, 1999*.
- [VB00] J. Veenstra and A. Bosch, Single-Classifer Memory-Based Phrase Chunking. In: *Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal, 2000*.
- [VD00] M. Vilain and D. Day, Phrase Parsing with Rule Sequence Processors: an Application to the Shared CoNLL Task. In *Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal, 2000*.
- [Vee99] Jorn Veenstra. Memory-Based Text Chunking. In *Workshop on Machine Learning in Human Language Technology, ACAI-99, Crete, Greece, 1999*.
- [Wah00] Wolfgang Wahlster, editor. *Verbmobil: Foundations of Speech-to-Speech Translation*. Springer, 2000.
- [XS03] N. Xue and L. Shen. Chinese Word Segmentation as LMR Tagging. In *Proceedings of the 2nd SIGHAN Workshop on Chinese Language Processing, in conjunction with ACL'03. Sapporo, Japan, 2003*.

- [XTAG98] The XTAG Research Group, "A Lexicalized Tree Adjoining Grammar for English". In *Proceedings of IRCS Tech Report 98-18, University of Pennsylvania, PA, USA, 1998*.
- [ZDJ02] T. Zhang, F. Damerau, and D. Johnson. Text chunking based on a generalization of winnow. In *Journal of Machine Learning Research, Volume 2, pp. 615-637, March, 2002*.
- [ZST00] G. Zhou, J. Su and T. Tey, Hybrid Text Chunking. In: *Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal, 2000*.