NAIST-IS-DD0961004

Doctoral Dissertation

Development of Pairwise Comparison-based Japanese Dependency Parsers and Application to Corpus Annotation

Masakazu Iwatate

March 16, 2012

Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Masakazu Iwatate

Thesis Committee:

Professor Yuji Matsumoto	(Supervisor)
Professor Hiroyuki Seki	(Co-supervisor)
Associate Professor Masashi Shimbo	(Co-supervisor)
Project Associate Professor Masayuki Asahara	(Co-supervisor, National Institute for Japanese Language and Linguistics)

Development of Pairwise Comparison-based Japanese Dependency Parsers and Application to Corpus Annotation*

Masakazu Iwatate

Abstract

For machine-learning-based Japanese dependency parsing, it is essential to develop an accurate parsing model. It is also essential to construct a corpus of a new domain efficiently when parsing sentences of the domain.

For developing an accurate parsing model, we propose three approaches: developing a single accurate model, combining parsers, and combining a parser with a coordination analyzer.

Considering selectional preferences between a dependent bunsetsu and its candidate head bunsetsus plays an important role to construct an accurate parser. In Japanese dependency parsing, Kudo's relative preference-based model [27] outperforms both deterministic parsing models and probabilistic CFG-based parsing models. In the relative preference-based model, an ME model estimates selectional preferences for all candidate heads, which cannot be considered in the previous parsing models. We propose a parsing model in which the selectional preferences are directly modeled by one-on-one games in a step-ladder tournament. In the evaluation experiment with Kyoto Text Corpus Version 4.0, the proposed model outperforms the previous researches, including the relative preference-based model. We also investigates the accuracy of partial parsing of three SVM-based Japanese dependency parsing models: the shift-reduce model, the cascaded chunking model, and the tournament model. We show the coverage-accuracy curves based on the scores produced by SVMs. Our performance evaluation with the Kyoto Text Corpus shows that the partial parsing accuracy of the tournament model is the highest among the three dependency parsing models. The tournament model achieves 99% accuracy with 60% bunsetsu-based coverage.

*Doctoral Dissertation, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0961004, March 16, 2012.

We also conducted the stacking experiments with KNP, which is a Japanese dependency parser and is different in several aspects from the tournament model. The tournament model incorporates the output of KNP as guide features. The stacked model outperformed the individual parsers.

One of the difficult problems for dependency parsers is to identify spans of coordinated structures. A solution is to employ a coordination analyzer, which is specialized for capturing such structures. Employing such an analyzer raises the problem of how to integrate the output of the dependency parser and that of the coordination analyzer. We employ the dual decomposition algorithm, which is a decoding algorithm designed to integrate two analyzer's outputs. For the algorithm, we propose a parsing model which inherits the advantage of the tournament model.

For constructing a corpus efficiently, we propose two approaches. The first is active learning, which selects training sentences to be annotated that maximize the impact to parsing accuracy. Preliminary experiments show that our SVM score-based active learning method can be used to select effective sentences to be annotated. Second, given the coordinated structure annotation of a sentence, possible dependency structures are constrained by the coordinated structure. We propose to boost the efficiency of dependency structure annotation by construction of a precise dependency parser using the annotation. The dependency parser provides dependency structure annotators an automatic parse with fewer errors. We conducted experiments with the Balanced Corpus of Contemporary Written Japanese (BCCWJ) and show that our method improves automatic parsing accuracy and detects dependency annotation errors.

Keywords:

natural language processing, Japanese, dependency parsing, tournament model, corpus annotation

候補の直接比較に基づく日本語係り受け解析器の開発とこれを用いたコーパスアノテーション支援*

岩立 将和

内容梗概

機械学習に基づく高精度の日本語係り受け解析を実現するためには，高精度の係り受け解析モデルが必要なのはもちろんのこと，新しいドメインのコーパスを高精度で解析しようとする際には，解析器の訓練データとなるそのドメインのコーパス構築を効率的に行えることも重要である．

本稿では高性能な係り受け解析モデルの実現に関して，3 つのアプローチをとる．高精度の単体解析モデルの開発，係り受け解析器の組み合わせによる精度向上の実現，並列構造解析器との組み合わせによる精度向上の実現である．

高精度の単体解析モデルの開発に関して，日本語係り受け解析においては，工藤らの相対的な係りやすさを考慮したモデルが，決定的解析モデルや文脈自由文法を用いた構文解析アルゴリズムに基づく手法を上回る精度を示している．決定的解析手法が二文節間に係り受け関係が成立するか否かの判定に基づく解析であるのに対し，工藤らのモデルではすべての係り先候補文節間の相対的な選択選好の強さを学習し，解析時には各係り先候補について選択選好の強さのスコアを計算し，それらを相互に比較して係り先を選択する．これに対し本稿では，一度に比較対象とする係り先候補文節を二候補に限定し，選択選好を二つの候補同士の対戦からなるトーナメントで直接表現したモデル（トーナメントモデル）を提案する．京大コーパスを使用した実験において，提案手法は従来手法を上回る精度を示した．本稿ではさらに，トーナメントモデルを含めた，SVM に基づく 3 種類の日本語係り受け解析器の部分解析精度を評価する．SVM の出力する分離平面からの距離をスコアとみなし，このスコアと部分解析精度との関連を調査したところ，トーナメントモデルによる手法が最も高性能であり，SVM のスコア上位 60%の係り受け関係を選択すると 99%の係り受け正解率を実現できることがわかった．

*奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 博士論文, NAIST-IS-DD0961004, 2012 年 3 月 16 日.

また，トーナメントモデルとは大きく性質の異なる日本語係り受け解析器である KNP の解析結果をトーナメントモデルの素性として用いることで，両解析器を上回る精度を達成できることを示す．

係り受け解析器にとって解析が難しい問題の一つに並列構造の範囲同定がある．これに対処する一つの方法として，並列構造の性質をとらえるのに特化した専用の解析器を用いることが考えられる．本稿では dual decomposition アルゴリズムを用いて並列構造解析器の出力と係り受け解析器の出力を統合する．このアルゴリズムはスコアに基づいて最適解を選択する解析器を想定しているため，トーナメントモデルの長所を保持しつつ分離平面からの距離に基づいて係り受け関係にスコアを付与したモデルを用いた．dual decomposition アルゴリズムは，係り受け・並列構造解析精度をわずかに改善できることがわかった．

コーパス構築において人間による作業の労力を削減し効率的なアノテーション作業を実現するために，本稿では 2 つのアプローチをとる．

まず，SVM のスコアを用いて係り受け解析器の能動学習を行うことで，人手によるアノテーションコストを削減しながら，高性能な係り受け解析器を構築できることが確認された．

次に，係り受け構造と並列構造を別々にアノテーションするという状況において，並列構造が与えられると可能な係り受け構造が制約されることを利用して高精度の係り受け解析器を構築することで，係り受けアノテーションの効率をさらに向上させる．また本手法は，並列構造に関連する係り受けアノテーション誤りの検出に用いることができる．

キーワード

自然言語処理、日本語、係り受け解析, トーナメントモデル, コーパスアノテーション

Acknowledgments

I would like to express my gratitude to my supervisor, Professor Yuji Matsumoto. His insightful comments and discussions with him supported, and encouraged me and my research continuously. He also gave advices about what readable and easy-to-understand papers and presentations are. The opportunity to join ChaKi, a corpus concordance and annotation tool, project which he gave showed me another aspects of my research. The aspect of providing supporting tools for Japanese linguistics researchers benefited me from not only learning Japanese Linguistics but also reminding me of importance of development of tools with usability in mind.

I also would like to thank Professor Hiroyuki Seki for reading this dissertation carefully and gave me helpful comments.

I also would like to express my gratitude to Project Associate Professor Masayuki Asahara, currently at National Institute for Japanese Language and Linguistics. He introduced the interesting research field to me and gave me helpful suggestions. His careful and continual support encouraged me.

I would like to thank Associate Professor Masashi Shimbo for teaching me about a coordination analyzer and giving helpful comments.

I would like to thank Professor Kentaro Inui at Tohoku University for giving sharp comments from a different perspective of his research field. I would like to thank Assistant Professor Ryu Iida at Tokyo Institute of Technology for giving helpful comments. I would like to thank Assistant Professor Yotaro Watanabe at Tohoku University for fruitful discussion.

I would like to thank Mr. Asad Habib, my co-author of papers in the field of human-computer interaction. Discussions with him made a connection between NLP research and my interest in development of easy-to-understand and usable software.

I would like to thank all the laboratory members for discussion and giving me useful knowledge not only about my research.

Finally, I would like to thank my parents. They have been supporting and encouraging me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Analysis of Japanese text is performed by the pipeline[1] shown in Figure 1.1. The first stage of Japanese processing is morphological analysis. The analysis divides an input Japanese sentence into a sequence of words and assigns them to parts of speech (POSs) such as nouns or verbs. The next stage is *bunsetsu* chunking. A bunsetsu is similar concept to an English base phrase. It consists of at least one content words followed by zero or more function words. A bunsetsu chunker receives the output of a morphological analyzer and chunks it into a sequence of bunsetsus. A dependency parser receives the output of the chunker, and outputs a *dependency tree*. A dependency tree consists of dependency relations. In a dependency tree, a dependency relation is represented as a directed arrow from a *dependent* bunsetsu to its *head* bunsetsu. The output dependency trees are often passed to a semantic analyzer such as a predicate-argument structure analyzer[2], or directly used by natural language processing (NLP) applications.

Semantic analysis or NLP applications such as information extraction or machine translation usually use the output of dependency parser supposing it is correctly parsed. Therefore, accuracy of dependency parsing is a factor that determines performance and usefulness of the applications. Per-bunsetsu accuracy of state-of-the-art parsers is about 92% and sentence accuracy, which is the percentage of sentences in which all dependency relations are determined correctly, is about 56%. More accurate parsers are

[1] A processing model that performs each analysis sequentially is referred to as a *pipeline* model.

[2] Since there are semantic analyses employing semantic dependencies between words [59], dependency parsing is sometimes referred to as *syntactic* dependency parsing.

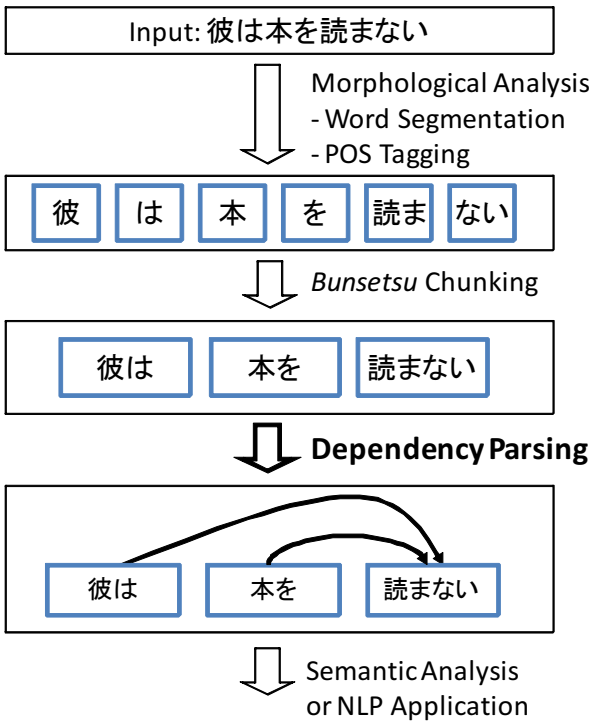


Figure 1.1: Position of dependency parsing in Japanese processing pipeline. POSs of words assigned by a morphological analyzer are not shown in this figure.

requested for the applications. Since morphological analysis and bunsetsu chunking are relatively accurate (over 99%), it is important to improve the accuracy of dependency parsing.

Statistical dependency parsing is also referred to as *data-driven* dependency parsing because rules for parsing sentences are not written by human but are learned from an *annotated corpus* by a machine learner. An annotated corpus is a text data in which several information is annotated by human. For example, Kyoto Text Corpus contains POS and dependency information. CaboCha[3]is one of the data-driven parsing models whose code has been released publicly. Another dependency parser named KNP[4], which is a manually-maintained rule-based parser, has also been released. KNP performs morphological through semantic analysis jointly rather than in pipeline. Because of manual maintenance, it is difficult for a rule-based parser to keep consistency and

[3]`http://code.google.com/p/cabocha/`
[4]`http://nlp.ist.i.kyoto-u.ac.jp/index.php?KNP`

wide-coverage of rules. On the other hand, data-driven dependency parsing requires human resources for corpus annotation[5]. Designing the tag set to be annotated, which is a prior step to annotation, requires professional expertise in NLP. However, annotators do not have to be professional. Corpus annotation usually employs preprocessing by an automatic analyzer for efficiency. Compared to completely manual annotation, which is very time consuming, it is more reasonable to let the annotators spot and correct errors in analyzer's output dependency trees.

1.2 Approaches for Accurate Dependency Parsing

We propose three approaches to parse a sentence more accurately. The first approach is to develop a more accurate single parsing model. The second is to combine two or more dependency parsers to exploit their advantages. The final approach is to combine a dependency parser with a coordination analyzer, which can be regarded as a partial parser.

1.2.1 Development of a More Accurate Single Parsing Model

Recent approaches to Japanese dependency parsing can be classified into two groups. The first group is *classification-based* approach. The classification-based model employs a binary classifier and tests *whether* a pair of two bunsetsus is in a dependency relation [25, 55]. The other group is *preference-based* approach. The preference-based model assigns a score to each of possible dependency relations. The score indicates *how likely* the pair of bunsetsus is in a dependency relation. Some of these models simply select the maximally-scored head of each bunsetsu, whereas some models define the score of a dependency tree as the product of score of each dependency relation and then search the dependency tree whose score is the highest. The latter models are preferable to as they allow us to introduce the constraint that prohibits crossing of dependency relations.

Especially in long sentences, whether a pair of bunsetsus is in a dependency relation sometimes depends on the context where an instance of the pair appears, such as

[5] *Annotation* or *tagging* is to assign information such as POSs or dependency relations to corresponding words or segments by human called *annotators*. The assigned information is also referred to as an *annotation* or a *tag*. On the evaluation of parsing accuracy, an annotation is referred to as the *gold-standard*. In other words, it is the correct answer if it is not an annotation error.

its surrounding words. The preference-based approach is one of the solutions. The approach makes a decision based on the frequency of instances which the pair is in a dependency relation in training data[6]. For classification-based approach, it is hard for a classifier to learn parsing rules from such inconsistent training examples. Instead, the approach intends to maximize its parsing accuracy by using heuristics such as that short dependency relations are more likely to be correct than long ones.

We propose another classification-based solution. In most cases, it can discriminate contexts where a pair of bunsetsus is in a dependency relation, from contexts where the pair is not in a dependency relation, by using wider context. Our model takes a dependent bunsetsu and its two candidate heads into consideration and selects the better candidate head out of those two. This step is repeated in a step-ladder tournament to get the best candidate head (hereafter we call this model as *tournament model*). The tournament model was first introduced by Iida et al. [14] for coreference resolution. We applied this model for selecting the most plausible candidate head for each bunsetsu except for the rightmost bunsetsu of a sentence. Considering two candidates allows the tournament model to improve its context discriminative performance. In other words, the binary classification problem in the tournament model tends to easier to solve than that of existing classification-based approaches. Since the former sees wider context, key information for a classification rarely be unseen on decision of the former.

We show that the tournament model is also accurate in partial parsing. Most research of dependency parsing intends to improve overall accuracy of parsing. However, not only overall accuracy but also partial parsing accuracy is important because NLP applications often do not request a full dependency tree but its subtrees. Suppose an information extraction system where a user specify a pattern to be extracted and the system searches it from a large amount of parsed Web text. The pattern may be converted to a subtree and pattern-matching based on the subtree is performed. To realize a high-precision information extraction system even sacrificing its coverage, the system should extract only the dependency relations whose confidence is sufficiently high. We propose an extraction method based on the distance from the hyperplane of support vector machines (SVMs) as the confidence score. SVMs are one of the binary classifiers where the tournament model uses. Coverage and accuracy of extracted relations are evaluated.

[6]A part of a corpus which is used for learning parsing rules is referred to as *training data*. In experiments, a corpus is split into training data and *test data*, which is used for evaluating parsing accuracy. They must be disjoint.

1.2.2 Combining Dependency Parsers by Stacking

It has been reported that incorporating co-occurrence information can improve parsing accuracy. Whether there is a dependency relation between a pair of words is strongly depends on their lexical entries, that is, the words themselves. For example, a "cat" may "run" but a "cake" never "run"s. Without any linguistic information, a data-driven parser would have to learn all the parsing rules from at most tens of thousands of training sentences. To learn such *selectional preference* information with wide-coverage, a very large annotated corpus is requested. However, since corpus annotation requires manual effort, construction of such corpus is impractical. Therefore, several research incorporated the information from unannotated corpora, such as using a word class ID which is common in training data and unannotated corpus, instead of the word itself [21]. Another solution is to extract reliable dependency relations or dependency subtrees from such an unannotated corpus [4, 3]. The IDs of equivalent classes based on their frequency are used instead of words themselves.

One of the easiest way for classification-based parser to incorporate such information is stacking with another parser which selects the most probable dependency tree based on such information. Stacking is a method for parser combination in which the output of the first parser is passed to the second parser as guide features [45, 35]. In multilingual word-based[7] dependency parsing, it has been reported that the more diverse the two parsers are, the more the final parsing accuracy improves. We employ KNP 3.01 as the first parser and the tournament model as the second parser. KNP employs case frames[8], which is a kind of dictionary of co-occurrence information, constructed from very large unannotated corpora.

1.2.3 Integration of a Parser and a Coordination Analyzer

One of the difficult problems for dependency parsers is to identify spans of coordinated structures. A coordinated structure is a structure such that two or more syntactically equivalent components are conjoined with coordinate conjunctions such as "と" (and). For example, "温暖化の抑制と経済の成長を両立する" (Pursue reduction of global warming and growth of economy simultaneously) contains a coordinated

[7] We use the phrase *word-based* in the sense that parsing models and experiments regard a word as a unit of parsing. Similarly, we refer to the regular Japanese dependency parsing setting as *bunsetsu-based* parsing.

[8] Only KNP 3.01 and newer versions incorporate such information.

structure “温暖化の抑制と経済の成長” (reduction of global warming and growth of economy). The coordinated structure consists of the two conjuncts: “温暖化の抑制” (reduction of global warming) and “経済の成長” (growth of economy). One of the reasons why coordination analysis is difficult is strong dependency to meaning and little dependency to syntax. A coordination analyzer may output another coordinated structure “抑制と経済”(reduction and economy). It is wrong because “growth of global warming’s reduction and economy” is semantically wrong even if it is syntactically correct. Another reason which is specific to Japanese is that the particle “と” does not always play a role of a coordinate conjunction. A solution is to employ a coordination analyzer, which is specialized for capturing such structures. Coordination analyzers based on similarity has been proposed [30, 58, 10, 48]. The two conjuncts of the coordinated structure are *similar* in the sense that:

- Both of the first bunsetsus of the conjuncts contain a particle “の” (of).
- In their first conjuncts, the words “温暖化” (global warming) and “経済” (economy) are semantically co-related in the sense that there were classified into a same class or near semantic classes in a thesaurus.
- Similarly, in their second conjuncts, “抑制” (reduction) and “成長” (growth) are also semantically co-related.
- Both conjuncts consist of two bunsetsus.

Employing such an analyzer raises the problem of how to integrate the output of the dependency parser and that of the coordination analyzer. We employ the dual decomposition algorithm [23], which is a decoding algorithm designed to integrate two analyzer’s outputs. For the algorithm, we propose a parsing model which inherits the advantage of the tournament model.

1.3 Approaches for Efficient Corpus Annotation

Generally, parsing accuracy significantly drops if domains of its training and test data are different. For example, the training data is a newspaper corpus and the test data is a blog corpus.

One of the ways to achieve an accurate parser for the blog data is to construct a sufficient amount of annotated blog corpus and to train a parser with the annotated corpus.

We propose two approaches for efficient corpus annotation: employing active learning, and exploiting coordinated structure annotation. The former reduces the number of sentences to be annotated required to achieve a certain objective parsing accuracy. The latter reduces the annotation cost for each sentence.

Once we constructed a certain amount of annotated blog corpus, we can apply the stacking as a domain adaptation method. Kyoto Text Corpus, a large newspaper corpus, is already available. We can exploit both of the corpora by stacking the first parser trained with the large newspaper corpus and the second parser trained with the small blog corpus. The first parser plays the role of parsing with general parsing rules and the second parser plays the role of adapting to domain-specific rules.

1.3.1 Active Learning

Active learning, or active sampling, boosts construction of an accurate parser by selecting training sentences to be annotated that maximize the impact to parsing accuracy. Active learning is an application of the scores defined as the distances from the hyperplane. We perform preliminary experiments of active learning for corpus construction and show correlation between the scores and partial parsing accuracy.

1.3.2 Exploiting Coordinated Structure Annotation

There are other possibilities to boost efficiency in the annotation process of the Balanced Corpus of Contemporary Written Japanese (BCCWJ) [34], which we have been constructing. We have been annotating dependency structures and coordinated structures separately. When constructing a corpus with dependency and coordinated structures, it is efficient to annotate coordinated structures first. This is because definition of coordinated structures is more understandable than that of dependency structures, and automatic coordination analysis is less accurate than dependency parsing. Dependency structure annotators work with the merged structure of dependency parser's output and coordinated structure annotation. Such sequential annotation keeps consistency between dependency structure annotation and coordinated structure annotation. This annotation order also enables annotators to grasp the structure of a long sentence instantly by the given coordinated structure annotation.

The coordinated structure annotation are not fully exploited. We constructed a more precise dependency parser using the constraints derived from the annotation. The constraints also detect dependency annotation errors.

1.4 Organization of the Dissertation

This dissertation is organized as follows. Chapter 2 describes an introduction to Japanese dependency parsing, and Japanese corpora and classifiers used in experiments. Chapter 3 surveys Japanese dependency parsing models and discusses what characteristics are important for accurate dependency parsing. This chapter also includes related work in multilingual dependency parsing. Chapter 4 proposes the tournament model for Japanese dependency parsing and show the model outperforms previous models. We then conduct error analysis of the parser's output. Chapter 5 defines the confidence score of dependency relations and evaluates the partial parsing performance. The tournament model achieves higher partial accuracy than other models. This chapter also shows that active learning using the scores reduces annotation cost. Chapter 6 conducts experiments of stacking the tournament model on KNP. The tournament model naturally incorporates co-occurrence information by stacking. We show the stacked model outperforms its individual parsers even in a domain adaptation setting. Chapter 7 proposes to integrate dependency parser's and coordination analyzer's outputs using the dual decomposition algorithm. We show that such combination of a parser and a coordination analyzer can improve accuracies of the both analyzers. Chapter 8 proposes a method to boost efficiency of dependency structure annotation by using coordinated structure annotation. We show the method significantly reduces parsing errors and then annotation cost significantly reduced. The method also contributes to efficient and high-quality annotation by detecting annotation errors. Chapter 9 lists our future work and Chapter 10 concludes this dissertation. Details of features are shown in Appendix A. Examples of parsing errors are listed, and appropriateness of annotations are evaluated in Appendix B.

Chapter 2

Data-driven Japanese Dependency Parsing

2.1 Japanese Dependency Structure

There are several differences between Japanese dependency structures and those of other languages. The main difference is that Japanese is a strictly head-final language. In other words, the head of every bunsetsu in a Japanese sentence is one of the bunsetsus located on the right of the dependent bunsetsu in the sentence. Another difference is that the basic unit of Japanese dependency parsing is a bunsetsu[1], unlike other languages whose basic unit is a word.[2]

Figure 2.1 shows example sentences for comparison between Japanese and English dependency structures. In the Japanese sentence, each box represents a bunsetsu. A dependency relation is represented as a directed arc from a dependent bunsetsu to its head bunsetsu. Although some authors draw the arcs from heads to their dependents, we draw them from dependents to their heads since we regard a dependency relation as a syntactic modification. In Figure 2.1(a), the head of "*kare-wa*" is "*hito-da*". We also refer to the dependency as that the former *depends on* the latter, or the former *modifies* the latter. On the other hand, in multilingual dependency parsing, many authors draw an arrow from the head to a dependent, as shown in Figure 2.1(b). The artificial root

[1]There are exceptions. Sassano and Kurohashi [56] conducted experiments of joint analysis of bunsetsu chunking and dependency parsing. Mori [40] conducted parsing experiments with a word-based dependency corpus.

[2]The CoNLL-2000 Shared Task [53], a workshop on chunking using an English corpus, was held. The workshop considers chunking to be a useful preprocessing step for parsing. However, word-based parsing is still the standard setting as in the CoNLL-2006 and 2007 shared tasks [1, 44].

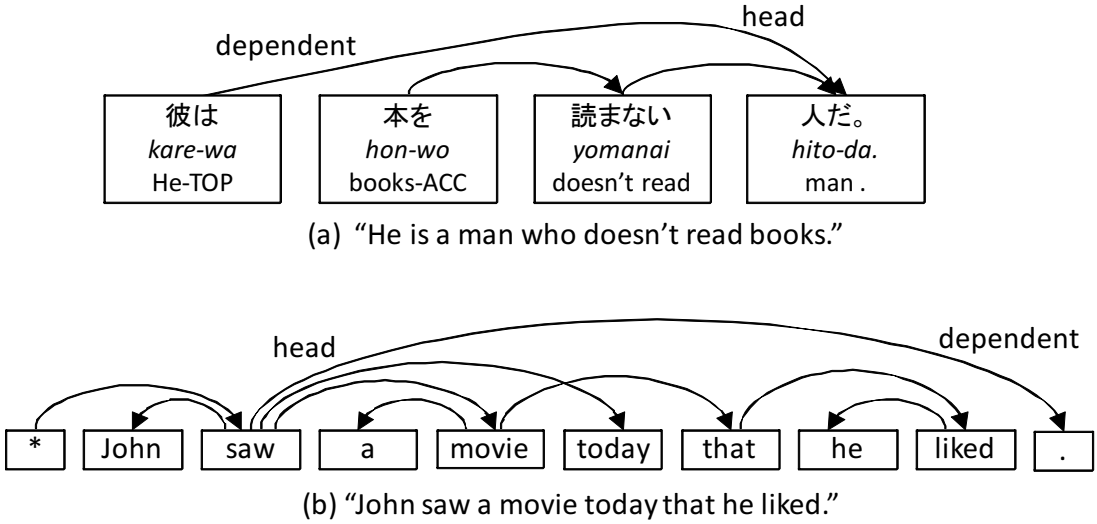


Figure 2.1: Examples of Japanese and English sentences.

symbol ("*") is always placed at the leftmost of the a sentence for convenience for parsing algorithms. Hereafter, we will draw an arrow from a dependent to its head even in an English dependency structure.

In dependency parsing research for English and other languages, a dependency structure of a sentence is formalized as a graph $G = (V, A)$ where nodes, or vertices, $V = \{1, 2, \ldots, n\}$ are words and arcs $A = \{(i_1, j_i), \ldots, (i_k, j_k)\}$ are dependency relations [42]. Since dependency structures are formalized as directed graphs, a dependent and its head are also referred to as a *child* and its *parent*, respectively. Similarly, in Figure 2.1(b), "movie", "today" and "." are *siblings* because they share the same parent "saw". Also, "liked" is a grandchild of "movie", and the latter is the grandparent of the former. They are also in an ancestor-descendant relation. A dependency parser should output *well-formed* dependency graphs. Well-formedness is defined as the set of constraints to be satisfied. A well-formed dependency graph is a connected and acyclic rooted tree.

In a similar manner, we define the constraints for Japanese dependency structures. Let $V = \{1, 2, \ldots, i, \ldots, n\}$ be a Japanese sentence where i corresponds to i^{th} bunsetsu. Similarly to the inverse direction of a dependency arrow, let $(j, i) \in A$ be a dependency relation where j is a dependent and i is its head. Well-formedness with respect to Japanese dependency structures is formalized as follows.

Definition (Well-formed Dependency Tree)

A dependency graph $G = (V,A)$ is a *well-formed* dependency tree if and only if:

1. The node n is the root and the graph is connected, that is, there is no node i such that $(n,i) \in A$, and for every node $j \in V$, there is a subset of arcs $\{(j,i_1),(i_1,i_2),\ldots,(i_{k-1},i_k)\}$ such that $i_k = n$. (SINGLEROOTED)

2. Every node has at most one head, that is, if $(j,i) \in A$ then there is no node i' such that $(j,i') \in A$ and $i \neq i'$. (SINGLEHEAD)

3. Every arc goes left to right, that is, if $(j,i) \in A$ then they must satisfy $j < i$. (HEADFINAL)

4. A dependency graph $G = (V,A)$ is *projective* if and only if, for every arc $(j,i) \in A$ and node $k \in V$, if $j < k < i$ then there is a subset of arcs $\{(i_m,i_{m-1}),\ldots(i_2,i_1),(i_1,i)\}$ such that $i_m = k$. (PROJECTIVE)

Parsers for many languages assume that correct dependency structures satisfy SINGLEROOTED, SINGLEHEAD and PROJECTIVE constraints. Since Japanese is a strictly head-final language, parsers for Japanese also assume that HEADFINAL constraint is satisfied. While there are Japanese sentences violating the constraints, Japanese dependency parsers usually assume that all input sentences to the parsers satisfy the constraints. There are few written Japanese sentences violating the constraints. Since the unit of dependency parsing is a sentence, it is implicitly assumed that there is exactly one correct dependency tree for a sentence. Moreover, it is also implicitly assumed that which dependency tree is the correct one is never affected by any other surrounding sentences.

Parsing algorithms whose output always satisfies PROJECTIVE constraint are referred to as *projective* algorithms. On the other hand, *non-projective* parsing algorithms can output *non-projective* dependency trees, which violate PROJECTIVE constraint.

Because of the Japanese constraints described above, we can simplify the dependency parsing as the task; for given bunsetsu sequence $x = \{b_1,b_2,\ldots,b_n\}$, identifying their corresponding head sequence $D = \{d_1,d_2,\ldots,d_n\}$. Since the rightmost bunsetsu does not have its head, we define $d_n = ROOT$ for convenience, as much previous work does. In this dissertation, we also adopt this formalization.

2.2 Japanese Corpora for Experiments

Kyoto Text Corpus

Most of our experiments were performed with Kyoto Text Corpus[3], where Mainichi newspaper articles and editorials written in 1995 were compiled. POSs and dependency relations are annotated on the corpus.[4]

Figure 2.2 shows a sentence in the corpus. Most of the lines correspond to words, such as the line beginning with "今年". The line consists of its lexical attributes: lexical form, ruby, lemma, coarse-grained POS, fine-grained POS, inflection type and inflection form. The character "*" indicates that the word does not have the attribute. The corpus employs JUMAN's[5] POS tag set. The corpus contains a few annotation errors. For example, the correct ruby of "四月" is not "よんがつ"(*yongatsu*) but "しがつ"(*shigatsu*). The corpus also contains dependency annotation errors. Some of the errors are violations of the annotation guideline [31][6].

A line beginning with "*" represents a bunsetsu boundary. The line includes the ID of the bunsetsu, its head and label of dependency relation. Words that follow the line belong to the bunsetsu. Although a bunsetsu ID is 0-origin, our formalization is 1-origin. The special bunsetsu ID "-1" at the head of the rightmost bunsetsu indicates the ROOT. The corpus defines four labels of dependency relations; label D is for normal dependency relations; label A is for appositions; label P is for parallel relations connecting conjuncts; label I is for certain relations inside of incomplete coordinated structures.

Label A is used for paraphrase or exemplification. In the following two examples, the first bunsetsu modifies the second bunsetsu with label A: "自動車専門誌『オートモービル』" (a car magazine *Automobile*) and "銀行など 金融機関" (financial institutions such as banks).

Label P indicates coordinated structures by connecting the rightmost bunsetsus of

[3]The corpus has various names such as Kyoto Corpus, Kyoto University Corpus or Kyoto University Text Corpus.

[4]Most of the annotations satisfy the constraints described above. The annotation guideline of Kyoto Text Corpus [31] allows violation of PROJECTIVE constraint only for appositions. Except for appositions, there are very small number of relations violating the constraint. However, such annotations seem to be appropriate.

[5]A morphological analyzer distributed at `http://nlp.ist.i.kyoto-u.ac.jp/index.php?JUMAN`.

[6]`http://nlp.ist.i.kyoto-u.ac.jp/nl-resource/corpus/KyotoCorpus4.0/doc/rel_guideline.pdf`

```
# S-ID:950101010-030 KNP:96/10/27 MOD:96/12/03
* 0 5D                                    // this year,
今年 ことし * 名詞 時相名詞 * *
は は * 助詞 副助詞 * *
* 1 2I                                    // in April
四 よん * 名詞 数詞 * *
月 がつ * 接尾辞 名詞性名詞助数辞 * *
に に * 助詞 格助詞 * *
* 2 4P                                    // nationwide local election
統一 とういつ * 名詞 サ変名詞 * *
地方 ちほう * 名詞 普通名詞 * *
選挙 せんきょ * 名詞 サ変名詞 * *
、 、 * 特殊 読点 * *
* 3 4I                                    // in July
七 なな * 名詞 数詞 * *
月 がつ * 接尾辞 名詞性名詞助数辞 * *
に に * 助詞 格助詞 * *
* 4 5D                                    // Upper house election
参院 さんいん * 名詞 普通名詞 * *
選挙 せんきょ * 名詞 サ変名詞 * *
が が * 助詞 格助詞 * *
* 5 -1D                                   // will be held .
あり あり ある 動詞 * 子音動詞ラ行 基本連用形
ます ます * 接尾辞 動詞性接尾辞 動詞性接尾辞ます型 基本形
。 。 * 特殊 句点 * *
EOS
```

Figure 2.2: An example sentence in Kyoto Text Corpus. "This year, a nationwide local election (will be held) in April and an Upper house election will be held in July." The first "will be held" is ellipted in the original Japanese sentence. The comment beginning with '//' is an English translation of the bunsetsu. We added the comments for description.

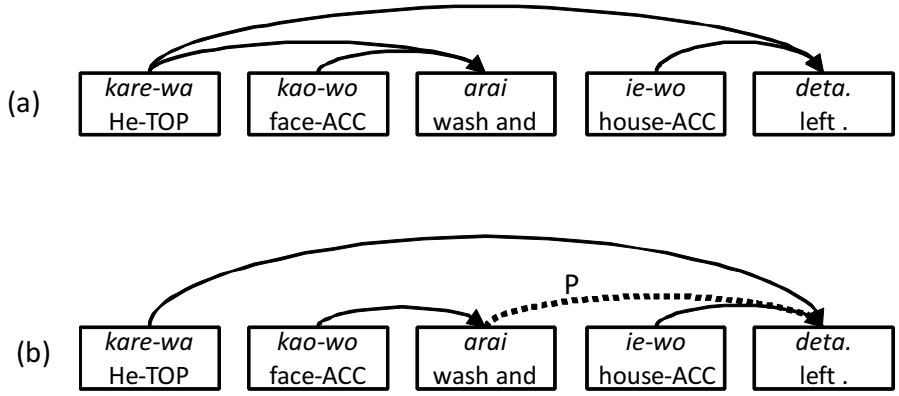


Figure 2.3: An example of label-P relation. (a) A straightforward dependency graph representing the coordinated structure. (b) A well-formed dependency tree with a label-P relation.

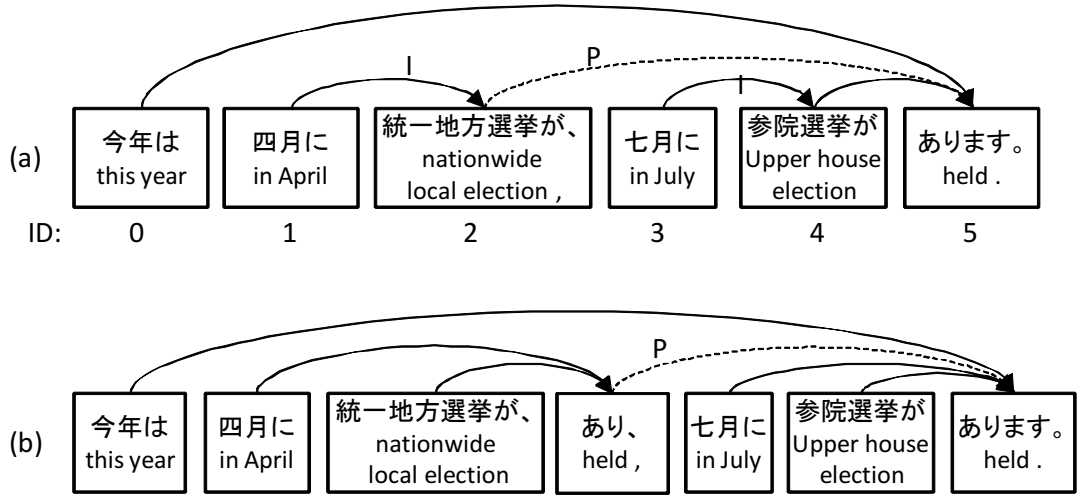


Figure 2.4: An example of (a) an incomplete coordinated structure and (b) its completed representation by adding the ellipted constituent.

conjuncts. Figure 2.3 shows an example sentence "*kare-wa kao-wo arai ie-wo deta.*" (He washed his face and left the house.) If we do not consider the constraints described above, it seems to be natural that expressing the sentence as the dependency graph (a).[7] However, to satisfy the constraints, the annotator of the corpus set the relations as shown in Figure 2.3(b). That is,"*kare-wa*" modifies only "*deta.*" and "*arai*" modifies "*deta.*". The meaning implied by the dependency graph is that he did both washing and leaving.

The coordinated structure shown in Figure 2.3 is a coordination of verb phrases. Such a coordination is referred to as a *predicate coordination*, against a noun phrase coordination. Annotations to such a label-P relation tend to ambiguous and inconsistent in certain senses such as whether the two predicates are parallel (label P) or not (label D).

Label I is employed when some of syntactic constituents are ellipted, as shown in Figure 2.4. If the sentence was "今年は四月に統一地方選挙があり、七月に参院選挙があります。", a normal dependency relation would be annotated between "四月に" and "あり、" naturally, as shown in Figure 2.4(b). However, since "あり、" is ellipted in the original sentence, label I was employed. For example, "四月に" modifies "統一地方選挙が、", which is the rightmost bunsetsu of the conjunct "四月に統一地方選挙が、".

We do not use the labels for most of experiments. The task is referred to as *unlabeled* dependency parsing, which identifies only heads of bunsetsus. The task that identifies both heads and labels is referred to as *labeled* dependency parsing.

Several versions of Kyoto Text Corpus have been released. Version 2.0 is smaller than later versions: newspaper articles of January 1st to 10th (about 10,000 sentences). Version 3.0 is larger: articles of January 1st to 17th and editorials of a year (about 40,000 sentences). Version 4.0 refined Version 3.0 and added semantic annotations. Since we do not have Version 2.0, we use corresponding part of Version 4.0 in some of experiments.

[7] Semantic analyses such as predicate-argument structure analysis employ such a representation. That is, every predicate governs its arguments individually.

EDR Corpus

EDR Japanese corpus[8] is used in several experiments conducted in previous work summarized in Section 3.6. The corpus is compiled from texts such as magazines, newspapers and an encyclopedia. Syntactic structure of a sentence is represented as the form of an S-expression. Unlike Kyoto Text Corpus, the unit of an S-expression is a word. However, a sentence in the corpus is usually converted to a bunsetsu-based dependency structure.

Kyoto University-NTT Blog (KNB) Corpus

KNB corpus[9] is used in the stacking experiments in Chapter 6. The corpus consists of 249 blog articles of 4 themes: Kyoto sightseeing, mobile phones, sports and gourmet. POSs, dependency relations and other several information are annotated. The annotation criteria is same as Kyoto Text Corpus.

Balanced Corpus of Contemporary Written Japanese (BCCWJ)

BCCWJ is the balanced corpus which consists of wide range of sub-corpora such as white papers (OW, for short), *Yahoo! Chiebukuro*[10] Q&As (OC), newspapers (PN) and books (PB).

We have been participating the project to construct BCCWJ. Our mission is to annotate dependency structures and coordinated structures on the corpus. Unlike Kyoto Text Corpus, a coordinated structure is annotated not as a set of the special dependency relations but as a set of spans of its conjuncts. In Chapter 8, we describe the annotation criteria for BCCWJ and then we develop a method which exploits coordinated structure annotation for efficient dependency structure annotation by reducing parsing errors and detecting annotation errors.

[8] `http://www2.nict.go.jp/r/r312/EDR/J_index.html`

[9] `http://nlp.ist.i.kyoto-u.ac.jp/kuntt/index.php?FrontPage`

[10] It is a web community where people ask questions and other people give answers. `http://chiebukuro.yahoo.co.jp/`.

2.3 Supervised Classifiers

In data-driven parsing, a parser learns parsing rules from a training corpus by using supervised classifiers. A *classifier* is a machine learner to classify a given example into one of several classes. A supervised classifier is trained using training data. A binary classifier is a classifier whose classes are exactly two. The two classes of a binary classifier are represented by "+1" and "-1". A training example belonging to the class "+1" is referred to as a *positive example* and a training example belonging to the class "-1" is referred to as a *negative example*.

Most of features used in NLP tasks are *binary features* such as "*whether* the POS of the leftmost word of the dependent bunsetsu is verb." The corresponding element is set to 1 in a feature vector if the question is yes and set to 0, otherwise. Typically, most of values of a feature vector are 0. Therefore, a feature vector is often written by enumerating activated features such as "`dependent-rightmost-pos:verb, candidate-rightmost-pos:noun`". Their feature values are not written explicitly since all the values are 1.

Support Vector Machines (SVMs)

SVMs are binary classifiers based on margin maximization. A training example is a pair of training feature vector $\mathbf{x_i} \in R^n$ and its class label $y_i \in \{+1, -1\}$. When a set of training examples $(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)$ are given, an SVM learns a hyperplane that maximizes the margin between positive and negative examples. The hyperplane is a subset of feature vectors of training examples, with their learned weights. Such feature vectors are referred to as *support vectors*. The hyperplane is the decision function $f(\mathbf{x})$ for a new example feature vector $\mathbf{x}$:

$$f(\mathbf{x}) = sign(\sum_{\mathbf{z_i} \in SV} a_i y_i K(\mathbf{x}, \mathbf{z_i}) + b)$$

where SV is the set of support vectors, a_i is the weight corresponding to a support vector $\mathbf{z_i}$, and y_i is its class label. The function

$$sign(x) = \begin{cases} 1 & (x = 0) \\ \frac{x}{|x|} & (otherwise) \end{cases}$$

returns either $+1$ or -1. A kernel function $K(\mathbf{x}, \mathbf{z_i})$ maps a feature vector into higher dimensional space. Our experiments use the third degree polynomial kernel. The d^{th}

degree polynomial kernel

$$K(\mathbf{x}, \mathbf{z_i}) = (\mathbf{x} \cdot \mathbf{z_i} + 1)^d$$

allows a parser to incorporate combinations of up to d features implicitly. The signed distance from the hyperplane

$$\sum_{\mathbf{z_i} \in SV} a_i y_i K(\mathbf{x}, \mathbf{z_i}) + b$$

is also useful for applications which consider the confidence of a classification.

We used one of the SVM implementations, TinySVM[11].

Maximum Entropy (ME) models

An ME model learns the weights of each feature w_i subject to maximize the entropy of the model satisfying the empirical probabilities of features. The probability that a new example x belongs to a class y is

$$P(y|x) = \frac{\exp(\mathbf{w} \cdot \Phi(x))}{\sum_{y \in \{+1,-1\}} \exp(\mathbf{w} \cdot \Phi(x))}$$

where $\Phi(x)$ is the feature vector representation of x. Unlike an SVM with a polynomial kernel, the ME model does not automatically incorporate all possible feature combinations. We need manual effort to incorporate effective feature combination.

Passive Aggressive (PA) Algorithms with Polynomial Kernels

PA algorithms are on-line learning algorithms similar to the perceptron algorithm. Same as perceptrons, the class of a new example x is define as

$$f(\mathbf{x}) = sign(\mathbf{w} \cdot \Phi(x)).$$

Unlike perceptrons, PAs dynamically control the update rate in each iteration based on a corresponding feature vector. PA-I and PA-II are variations with different updating equations. Unlike SVMs, PAs do not implicitly combine features by using polynomial kernels.

[11] `http://chasen.org/~taku/software/TinySVM/`

Yoshinaga et al. [71] proposed to incorporate kernels to PAs. Since input and output formats of an SVM and a PA with polynomial kernels can be identical, we can replace SVMs with the PAs easily. The PAs in Yoshinaga's implementation *opal*[12] are much faster than SVMs.

[12] *Opal* is distributed at `http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/opal/`. We used the version released on October 28th, 2010.

He also released a Japanese dependency parser software *jdepp* at `http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/jdepp/`. The implemented parsing algorithms include the tournament model.

Chapter 3

Previous Work

In this chapter, we survey Japanese dependency parsing models and discuss what characteristics are important for accurate dependency parsing.

Dependency parsing is a structure learning task which is solved by decomposing it into subproblems. An input to Japanese dependency parsing is a sequence of bunsetsus and its output is a structure, that is, a dependency tree. Unlike classification tasks such as document classification, it is unrealistic for a machine learner that a whole sentence is passed by a parser as a single example, and then returns a dependency tree. Therefore, dependency parsing of a sentence should be divided into subproblems.

The architecture of a data-driven dependency parser can be divided into the following four elements:

- Factorization: how to decompose the task into subproblems, in other words, what a unit of a classification is.
- Integration: the strategy to obtain the most appropriate dependency tree for a sentence with the results of subproblems.
- Classifiers: choice of classifiers to solve each subproblem.
- Feature templates: what information should be extracted from a bunsetsu or other constituents corresponding to a subproblem, which has a great impact to a parsing accuracy.

These elements are not independent each other. Designing a classifier-based parsing algorithm means designing its factorization and integration methods. However, it sometimes depends on a machine learner whether a parsing model can be implemented

Head selection procedure	Considering relative preference in training? No	Yes
Preference-based	absolute preference models (Section 3.1)	indirect comparison models (Section 3.3)
Classification-based	transition-based models (Section 3.2)	direct comparison model (Section 3.4)

Table 3.1: Classification of Japanese dependency parsing models.

naturally. It can improve the accuracy to adopt a high-performance machine learner or appropriate feature templates which capture the property of the task.

We classify previous research into the four types of parsing models: absolute preference models, transition-based models, indirect comparison models and a direct comparison model. As shown in Table 3.1, these types of models are illustrated with two axes: head selection procedure in parsing, and whether considering relative preference between candidate heads in training. The preference-based head selection procedure selects the most likely candidate head of a dependent, that is, it select the candidate with the highest preference score. In contrast, the classification-based procedure selects the head of a dependent by classifications. Models considering *relative preference* learn that the correct candidate head of a dependent is more likely to be its head *compared to the other candidates*.

3.1 Absolute Preference Models

An absolute preference model is a sort of preference-based models. Preference-based models factorize the problem of dependency parsing of a sentence into subproblems which are based on the probabilities of how likely a pair of bunsetsu is in a dependency relation. The models make an independence assumption; given an input bunsetsu sequence x, the probability of a dependency tree y corresponding to x is defined as the product of probabilities of each of dependency relations in y:

$$P(y|x) = \prod_{j=1}^{n-1} P(b_j \rightarrow b_i|b_j)$$

where $P(b_j \rightarrow b_i|b_j)$ is the probability that a dependent bunsetsu b_j modifies one of its candidate head bunsetsu $b_i \in C_j$.[1] The set C_j is candidate heads of a dependent b_j. In a naive definition, C_j is the set of all the bunsetsus located on the right of b_j. Preference-based models choose dependency relations subject to maximize the probability of a dependency tree.

Absolute Preference Models and Relative Preference Models

How can we construct a model to estimate $P(b_j \rightarrow b_i|b_j)$ when a pair of bunsetsus b_j and b_i is given? A simple solution is to generate a positive training example if the pair is in a dependency relation in a training data. Similarly, a negative example is generated if the pair is not in a dependency relation. These training examples is for a binary classifier and is generated for each instance of the bunsetsu pair in the training data. Training examples are generated for all the bunsetsu pairs in a sentence in a same manner.

Kudo and Matsumoto refer to the parsing model as the *absolute preference model*, in contrast to their *relative preference model* [27], which we will describe in Section 3.3. We use the word "absolute" in the sense that the absolute preference model estimates how b_j likely to modify b_i from only the two bunsetsus. The model does not consider the plausibilities of the other candidate heads. The absolute preference model is a *single-candidate model* or a *2-tuple model* in the sense that the model considers 2-tuple of a dependent and a candidate at each classification.

The absolute preference model generates the training examples shown in Table 3.2 when the two training sentences shown in Figure 3.1 are given. Both the two sentences generates the training examples whose dependents are *kare-wa* and candidates are *yomanai*. However, their class labels are different. To avoid generating such confusing training examples, a feature is necessary that expresses the difference of their contexts. In this example, a period can be a clue. It is contained in the dependent of the example 8 but is not contained in the example 2.

In contrast to the absolute preference model, relative preference model can parse these sentences correctly. The model learns that *hito-da* is more preferable to *yomanai* as the head of *kare-wa*.

[1] We use j for the index of a dependent bunsetsu and i for the index of a candidate head bunsetsu, as Sassano [55] did.

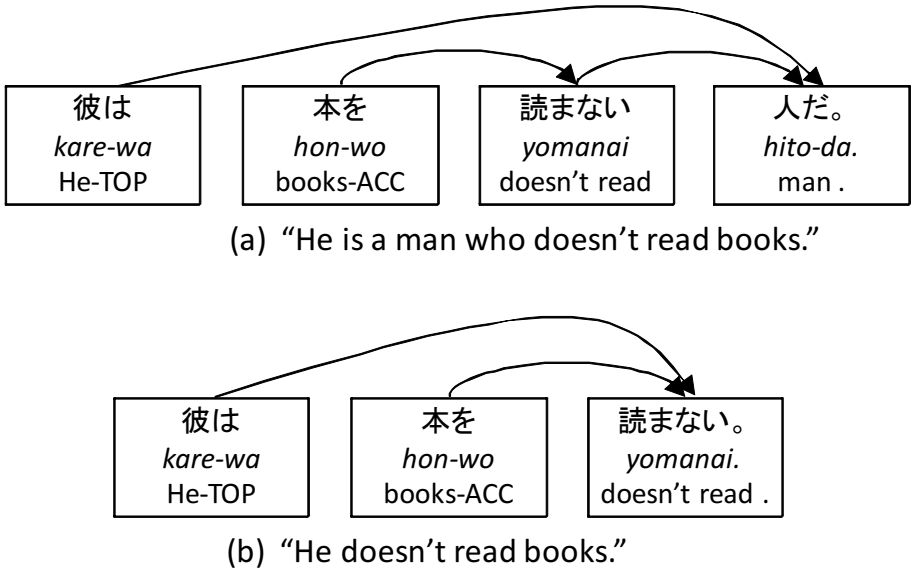


Figure 3.1: Examples of Japanese sentences.

	Sentence	Dependent	Candidate head	Class label
1		*kare-wa*	*hon-wo*	No
2		*kare-wa*	*yomanai*	**No**
3	(a)	*kare-wa*	*hito-da.*	Yes
4		*hon-wo*	*yomanai*	Yes
5		*hon-wo*	*hito-da.*	No
6		*yomanai*	*hito-da.*	Yes
7		*kare-wa*	*hon-wo*	No
8	(b)	*kare-wa*	*yomanai.*	**Yes**
9		*hon-wo*	*yomanai.*	Yes

Table 3.2: Generated training examples from sentences (a) "*kare-wa hon-wo yomanai hito-da.*" and (b) "*kare-wa hon-wo yomanai.*" by the absolute preference model. The class label "Yes" is the positive class and "No" is the negative class.

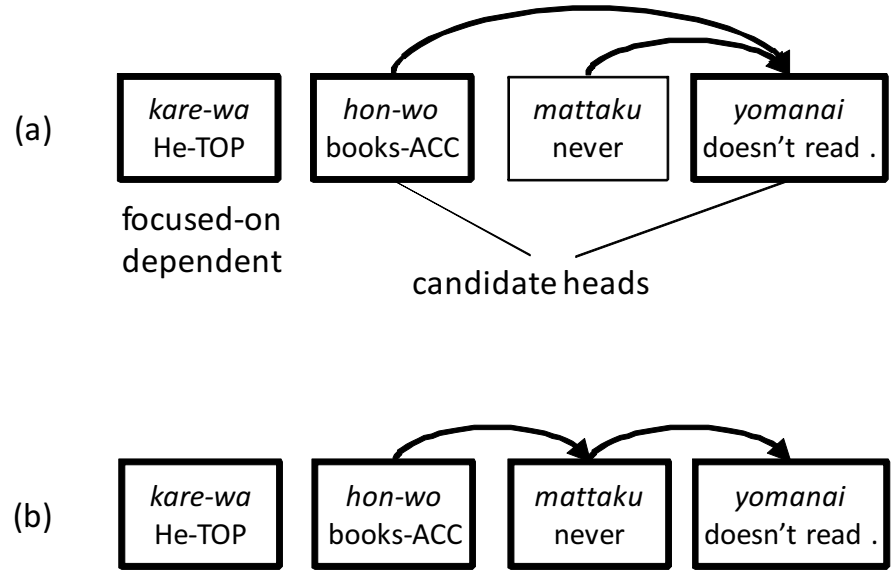


Figure 3.2: Parsing the sentence "*kare-wa hon-wo mattaku yomanai.*" ("He never read books.") from right to left. The beam holds possible subtrees (a) and (b).

Projective Parsing with Beam Search by Absolute Preference Models

Sekine et al. [57] and Uchimoto et al. [65] proposed the absolute preference models which perform beam search. Beam search is an approximate search algorithm that incrementally constructs a dependency tree with limiting the maximum number of possible dependency subtrees which the algorithm preserves. Their models perform a beam search from the end of sentence to the beginning of sentence[2] with considering PROJECTIVE constraint.

The approximate search algorithm reduces the number of estimations and comparisons of preferences required to parse a sentence. That is, possible candidate heads of a dependent are limited to ones which satisfy the PROJECTIVE constraint. An example is shown in Figure 3.2(a). Their models perform a beam search from right to left. Since the rightmost bunsetsu has no head, the beam search begins from the second rightmost bunsetsu "*mattaku*". Since the bunsetsu has only one candidate head "*yomanai*", a dependency relation is added between the pair. The third rightmost bunsetsu "*hon-wo*" has two candidate heads, "*mattaku*" and "*yomanai*". The beam search

[2]We refer to this parsing order as *right-to-left* parsing.

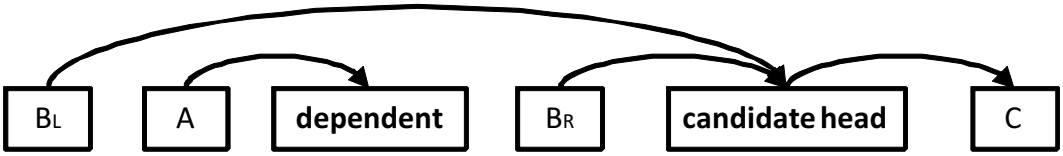


Figure 3.3: Types of dynamic features.

algorithm preserves the two possible subtrees shown in Figure 3.2. In other words, the *beam* of the algorithm contains the two subtrees. The algorithm proceeds with each of the two possibilities. In the final step, the focused-on dependent is the leftmost bunsetsu "*kare-wa*". The dependent has only two candidate heads if "*hon-wo*" modified "*yomanai*" because adding the dependency relation from "*kare-wa*"to "*mattaku*" violates the PROJECTIVE constraint. The absolute preference model does not perform the classification corresponding the pair. Compared to exact search algorithms, the beam search algorithm reduces computation costs, especially for long sentences. The time complexity of the beam search algorithm is $O(bn^2)$ for the number of bunsetsus in a sentence n and a beam size b. A beam size is the maximum number of possible dependency subtrees which the algorithm preserves.

Sekine et al. [57] reported that the parsing accuracy with the model hardly depends on a beam size. In addition, its parsing accuracy decreased with beam sizes > 10. This result shows that narrowing down the possibilities on each step of beam search can improve the accuracy rather than the dependency tree with highest probability.

Deterministic Parsing by Absolute Preference Models

Parsing with the setting $b = 1$ is referred to as *deterministic* parsing. The setting $b = 1$ is widely used because the time complexity is lowest and the parsing accuracy is comparable to the accuracy with best beam size. The time complexity becomes $O(n^2)$. In deterministic parsing, a parser selects the most probable candidate head for each focused-on dependent:

$$d_j = argmax_{c \in C_j} P(b_j \rightarrow c | \langle b_j, c \rangle).$$

To implement the probabilities, Sekine et al. [57] and Uchimoto et al. [65] used an ME. Kudo and Matsumoto [24] used an SVM. The latter outperformed the former in their parsing accuracy. While an SVM is not a probabilistic classifier, they regularized a

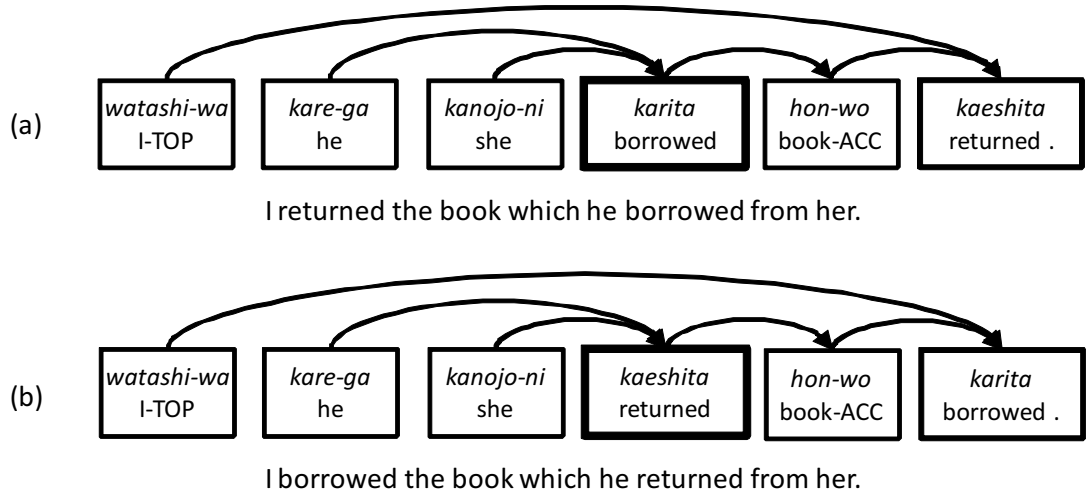


Figure 3.4: Two similar sentences.

distance from the hyperplane into the range of probability by using a sigmoid function $\sigma(x) = tanh(x)$; $0 < \sigma(distance) < 1$.

Although deterministic parsing with PROJECTIVE constraint reduces computation costs, the problem of error propagation may arise. That is, an incorrect decision can make following decisions incorrect. In right-to-left parsing, if a parser selected a candidate head located on the right-hand-side of the correct head, correct heads of some dependents may be covered by the incorrect dependency relation. Such dependents cannot modify their correct heads in later steps.

Kudo and Matsumoto [24, 25] proposed to incorporate *dynamic features*, which are features corresponding to already-determined dependency relations. As shown in Figure 3.3, they proposed the three types of dynamic features: A, B and C. The dynamic feature A corresponds to dependents of the focused-on dependent. The dynamic feature B corresponds to dependents of a candidate head. The dynamic feature C corresponds to the head of a candidate head. We separate the dynamic feature B into B_L and B_R based on whether the dependent is located on the left or right of the focused-on dependent. Available dynamic features depend on a parsing order. For example, the dynamic features B_R and C are available for a right-to-left parsing. The other dynamic features are not available because their heads have not been identified yet. An incorrect decision may also be harmful for parsers with dynamic features in the sense that parsers will see *wrong* contexts.

Disadvantages of Absolute Preference Models

An absolute preference model have two disadvantages. First, its procedures of training and parsing are not consistent. The model learns a *binary classification* problem whether a pair of bunsetsus is in a dependency relation. In contrast, in parsing, the model *compares absolute preferences* of candidate heads and chooses the most preferable one. Since the training procedure does not designed for comparing preferences, the two procedures are consistent if and only if only one candidate is classified as the positive class and the others are classified as the negative class. However, it may occur that no candidate, or two or more candidates are classified as the positive class. The second problem is its low context discriminative performance. The model looks at only a dependent and a candidate head when it estimates the strength of their preference. However, the preference for a candidate may depend on the context in it. For example, the two sentences shown in Figure 3.4 are exactly the same except for the positions of the two verbs: "*karita*" and "*kaeshita*". The absolute preference model cannot discriminate the two sentences.[3]

Tamura et al. [60] proposed a solution for the former problem by using the technique of error correcting codes. An SVM classifies whether each pair of bunsetsus is in an ancestor-descendant relation. Then, the results are integrated with error corrections based on hamming distances or other distance functions.[4]

Farther-Modify-Between Model

Uchimoto et al. [64] proposed a model based on the classification with three classes. The three classes are defined with respect to the relative position of the correct head, given a dependent and a candidate head: the correct head is farther than the candidate (*farther*), the candidate is the correct head (*modify*) , and the correct head is between the dependent and the candidate (*between*). The probability of a dependency relation

[3]Our discussion supposes that we use only the features corresponding to a dependent and a candidate head, and ignores dynamic features or contextual features such as the information of bunsetsus between a dependent and a candidate head. One reason is to simplify the discussion. Another reason is that providing such features are sometimes harmful. Providing a large amount of features often confuses a machine learner and then the parsing accuracy often falls.

[4]They also conducted an experiment with signed distances which an SVM output. They employed the cosine distance as the distance function. The accuracy outperforms that with hamming distances.

is estimated as follows:

$$P(b_j \rightarrow b_i | \langle b_j, b_i \rangle) = P(modify | \langle b_j, b_i \rangle) \times \prod_{k=j+1}^{i-1} P(farther | \langle b_j, b_k \rangle) \times \prod_{k=i+1}^{n} P(between | \langle b_j, b_k \rangle).$$

The model can discriminate the two sentences shown in Figure 3.4. Suppose the focused-on dependent is "*kare-ga*" and the focused-on candidate is "*karita*". The two candidate heads "*hon-wo*" and "*kaeshita*" belong to the class *between* in the sentence (a). In contrast, the candidates belong to the class *farther* in the sentence (b). Therefore, the probabilities $P(kare\_ga \rightarrow karita | \langle kare\_ga, karita \rangle)$ will be different in the two sentences.

3.2 Transition-based Models

Kudo and Matsumoto [25] proposed a deterministic parsing model which uses an SVM as not a preference estimator but just a binary classifier. Unlike preference-based models, the model performs a binary classification *whether* two bunsetsus are in dependency relation, in both its training and parsing. Such models are referred to as *transition-based* models or *deterministic incremental* models [42]. The models are more deterministic than absolute preference models. The transition-based models add a dependency relation when the SVM decided that the two bunsetsus are in a dependency relation, without considering the other candidates. The models perform the transition action corresponding to a decision of the SVM.

Kudo and Matsumoto's model [25] is referred to as *cascaded chunking model (CC model)*. The model iteratively chunks each neighboring bunsetsu pair if the SVM judged that the pair is in a dependency relation. A parsing example is shown in Figure 3.5. Parsing begins from the beginning of a sentence. The first decision is whether the two neighboring bunsetsus "*kare-wa*" and "*hon-wo*" are in a dependency relation. The SVM estimates there is no relation. Then the SVM examines the pair "*hon-wo*" and "*mattaku*" and there is also no relation. The next pair "*mattaku*" and "*yomanai*" is decided that there are in a dependency relation without any SVM classification because "*yomanai*" is the rightmost bunsetsu. When parsing reaches the end of a sentence, the model removes the bunsetsus which satisfy a certain condition, from the workspace.

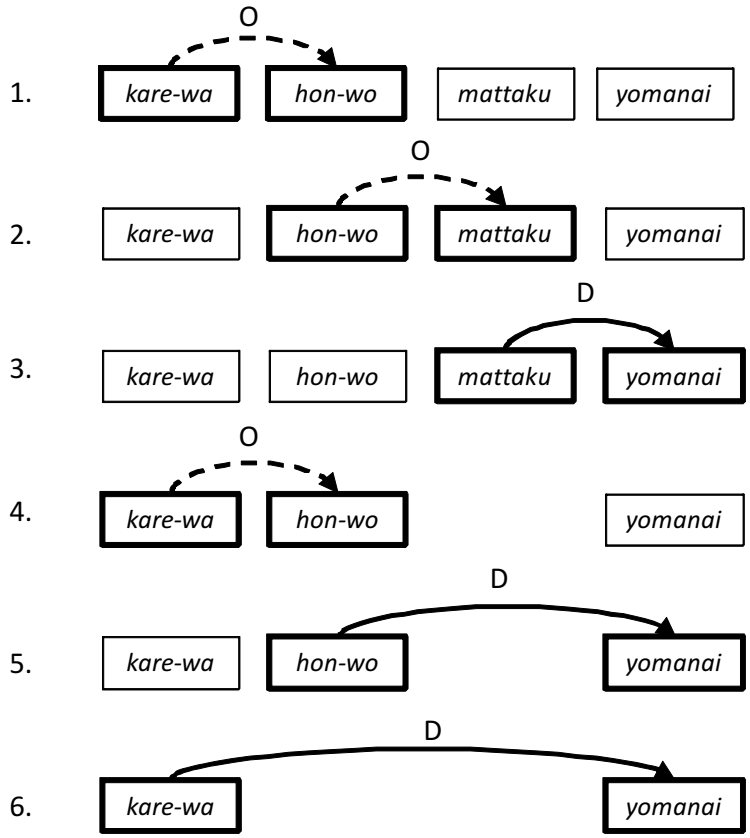


Figure 3.5: Parsing example of the sentence "He never read books." by the CC model. "D" represents that the pair of two bunsetsus is in a dependency relation. "O" represents that it is not. The model parses the sentence in the six steps.

Suppose b'_j is the j^{th} bunsetsu[5] in the workspace and d'_j is its head, and $d'_j = -1$ denotes that its head have not been determined. Additionally, suppose $d'_0 = -1$. The model removes all the bunsetsus b'_j such that $d'_j \neq -1$ and $d'_{j-1} = -1$. The only bunsetsu "*mattaku*" satisfies the condition. The condition is derived from PROJECTIVE constraint. The removed bunsetsu "*mattaku*" never be the head of any bunsetsus in later parsing steps because its immediate left bunsetsu "*hon-wo*" will modify farther bunsetsu than "*mattaku*". The model iterates such steps and results a dependency tree. Note that a pair such as "*kare-wa*" and "*hon-wo*" can be estimated twice or more. However, it may not be redundant if dynamic features are incorporated because the

[5]The index of the leftmost bunsetsu is 1.

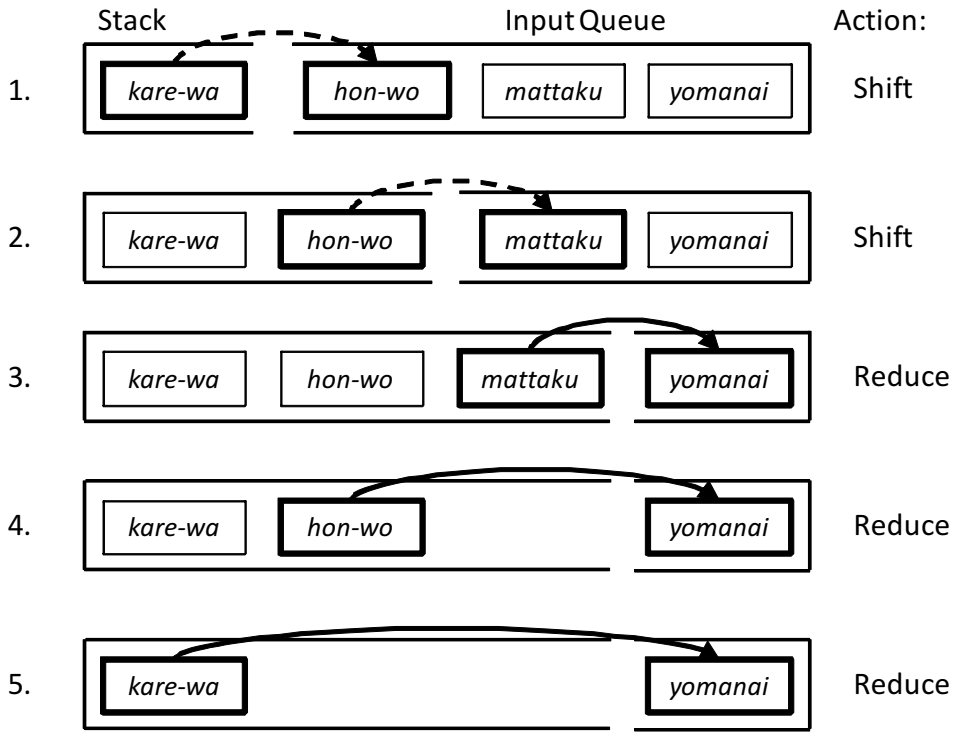


Figure 3.6: Parsing example by the SR model. The model parses the sentence in five steps.

decisions may be changed.

Sassano [55] proposed a shift-reduce-like transition-based model (*SR model*). A parsing example is shown in Figure 3.6. The model employs a stack and an input queue. At the beginning of parsing, only the leftmost bunsetsu "*kare-wa*" is in the stack and the others are in the input queue. An SVM decides that whether the stack-top bunsetsu "*kare-wa*" and the queue-top bunsetsu "*hon-wo*" is in a dependency relation. Because the decision is "no", the model performs the *Shift* action, which pushes the queue-top bunsetsu into the stack. For the next pair, "*hon-wo*" and "*mattaku*", the *Shift* action is also performed. Since the pair "*mattaku*" and "*yomanai*" is in a dependency relation, the *Reduce* action is performed. The *Reduce* action adds a dependency relation between them and removes the stack-top bunsetsu "*mattaku*". A sequence of such decisions construct a dependency tree.

Deterministic parsing reduces the time complexity to parse. The SR model parses a sentence in exactly $2n-3$ steps for the number of bunsetsus n: $n-2$ *Shift* actions and

$n-1$ *Reduce* actions. The time complexity of the model is $O(n)$. The time complexity of the CC model is $O(n^2)$. However, it is empirically close to $O(n)$.

In the transition-based parsers, their training example generation algorithms are exactly the same as their parsing algorithms. That is, training examples are generated also by performing a sequence of actions. The only difference is that the parsers refer the gold-standard data in training. The models generate smaller number of training examples than absolute preference models. The former is $O(n)$ and the latter is $O(n^2)$. The former is a subset of the latter.

Transition-based parsers overcome one of the two disadvantages of absolute preference models. Unlike absolute preference models, the transition-based models use an SVM as just a classifier, and the methods of training and parsing are consistent. However, the problem of error propagation still remains.

There is another interpretation of the binary classification which the transition-based models perform. The models examine a nearer candidate head of a dependent earlier than a farther candidate. Therefore, the two classes of a classification correspond to the *modify* and *farther* classes of Uchimoto's model [64]. This parsing order, which satisfies the PROJECTIVE constraint, is based on the heuristic that a dependency relation tend to be short.[6]

3.3 Indirect Comparison Models

All models presented above perform the binary classification: whether a pair is in a dependency relation.[7] However, such models may confuse a classifier when two training examples are *conflicted*. That is, feature vectors of two training examples are exactly the same but their class labels are different. That is, one is $+1$ and the other is -1. In most cases, a reason of a conflict is that an SVM makes a decision without considering the other candidates.

Note that the heuristic of the transition-based models can avoid conflicts. The transition-based models do not generate inconsistent training examples from the sentences shown in Figure 3.4. The models learn the rule that "*kare-ga*" can modify both "*karita*" and "*kaeshita*". However, the opposite rules are not learnt because "*kare-ga*"

[6] About 60% of bunsetsus modify their immediate right bunsetsus in gold-standard data.

[7] Uchimoto's model [64] is slightly different. However, the following discussion is also applicable to the model.

is already removed when the model generates an example from a pair of a bunsetsu and the rightmost bunsetsu.

Kudo and Matsumoto [27] proposed a *relative preference model* which estimates the preference for a candidate based on the set of all the candidates of a dependent. The model employs an ME model as its machine learner. Absolute preference models learn the preference between a dependent and a candidate independently of the other candidates. In contrast, in training, the relative preference model maximizes the probability of the correct candidate, given the set of candidates. Its parsing algorithm is as same as absolute preference models. Since there are no class labels in the relative preference model, a label conflict never occurs.

A conflict in the relative preference model occurs with a certain 3-tuple $(d, \{c_1, c_2\})$ where d is a dependent bunsetsu and c_1 and c_2 are its candidate heads. The 3-tuple $(kare\_ga, \{karita, kaeshita\})$ in the two sentences of Figure 3.4 is an example of such a case. In the two sentences, the set of candidates of the dependent "*kare-ga*" is exactly the same. Therefore, the relative preference model cannot learn that which of c_1 and c_2 is more preferable. However, there are few such symmetric sentences. The number of conflicts in the relative preference model is expected to be quite small compared to the binary classification-based models because they cause a conflict by a certain 2-tuple (d, c).

Yamamoto and Masuyama [70] pointed out that the traditional distance feature does not discriminate precise distance between a focused-on dependent and a candidate head. The traditional distance feature is defined as a bucket feature, that is, a distance is categorized into either of the three buckets: 1 (they are neighboring), from 2 to 5 bunsetsu distant (2-5), or 6 or more bunsetsu distant (6+). In Figure 3.4, the distances between "*kare-ga*" , and both "*karita*" and "*kaeshita*" belong to the bucket 2-5. Although their actual distances are different in (a) and (b), the feature failed to discriminate them. A naive solution is to introduce a fine-grained distance feature: a feature for the distance 1, a feature for the distance 2, a feature for the distance 3, and so on. However, it raises data sparseness problem. It is also the reason that the traditional distance feature is defined as a bucket feature.

They focused on a preference rule of rule-based dependency parsers that a dependent modifies its nearest possible candidate head. According to the rule, the absolute distance between a dependent and a candidate is not essential. The relative distance between them is important to incorporate the rule.

They extended the relative preference model so as to take into consideration the

relative distance between candidates. Recognition of relative distances is realized as comparison of two candidates with a position feature. In comparison, the value of a position feature is "L" if the candidate is located on the left-hand side of the other candidate. The value is "R", otherwise. The feature weights $\mathbf{w}$ of the relative distance model is trained for all j and $c \in C_j \setminus \{d_j\}$ subject to, if d_j is located on the left-hand-side of c,

$$\mathbf{w} \cdot F(\langle b_j, d_j \rangle, L) > \mathbf{w} \cdot F(\langle b_j, c \rangle, R)$$

otherwise,

$$\mathbf{w} \cdot F(\langle b_j, d_j \rangle, L) < \mathbf{w} \cdot F(\langle b_j, c \rangle, R).$$

In parsing, the most likely candidate of a dependent is selected by a series of comparisons. The comparisons form a step-ladder tournament. The model is similar to our model which we will present in Section 4.

3.4 Direct Comparison Model

All models presented above are *single-candidate models*. Given a dependent and a candidate head, the models assign a score or select a transition action.

In contrast, Kanayama et al. [16] proposed a model which restricts the number of candidate heads at most three, and then selects the best candidate by a single classification. Unlike the relative preference model, which compares candidate heads through their scores, the model compares candidates directly. The model first reduces the number of candidates by a head-driven phrase structure grammar (HPSG). If the number of candidates of a dependent is reduced to two, its head will be chosen by the 3-tuple model:

$$P(b_j \rightarrow c_n) = P(n|\Phi(b_j, c_1, c_2))$$

where b_j is the dependent, and c_1 and c_2 are the candidates reduced by the HPSG. The candidate c_1 is located on the left-hand-side of c_2. Similarly, if the number of candidates of the dependent is reduced to three, its head will be chosen by the 4-tuple model:

$$P(b_j \rightarrow c_n) = P(n|\Phi(b_j, c_1, c_2, c_3))$$

where b_j is the dependent, and c_1, c_2 and c_3 are the candidates reduced by the HPSG. The candidate c_1 is the leftmost and c_3 is the rightmost candidate among the three. If reduction by the HPSG results four or more candidates, a heuristic performs a further reduction. That is, the 4-tuple model will be applied to the three candidates: the leftmost two candidates and the rightmost candidate. Their experiments showed that the heuristic reduction limits the upper bound of parsing accuracy to 98.6%. That is, 1.4% of correct candidate heads cannot be selected due to the heuristic. However, the loss of accuracy is probably not really a matter because the accuracy of state-of-the-art parsers, which is around 90%, is relatively lower than 98.6%. In addition, the dependency relations which are not selected by the heuristic tend to be difficult to parse correctly and parsers probably fail to identify their correct heads.

The reduction plays the essential role of avoiding the data sparseness problem. We could construct a naive direct comparison model which selects the most probable candidate among all candidates in a single classification without any reduction of candidates. That is, the model trains a 3-tuple model, a 4-tuple model, a 5-tuple model, and so on. However, it is an unrealistic idea because this implementation will cause a serious data sparseness problem. For example, a training example of the 9-tuple model is generated from a dependent which has eight candidates. However, there are few such dependents, and it is hard to learn which features are useful for classifications because the number of features becomes quite large. Therefore, a method such as an HPSG is necessary in practice which reduces the number of candidates to a constant number.

The direct comparison model has two inseparable advantages. First, the model considers relative positions of candidates. In the above equations, the features $\Phi(\cdot)$ are extracted with preserving relative positions of the candidates, that is, which candidate is the leftmost and which candidate is the rightmost. Relative positions can discriminate confusing contexts. In the example sentences shown in Figure 3.4, "*kare-wa*" modifies the *nearer* verbs in both sentences. The model can parse such sentences correctly by learning the rule that "in such a case, the head of the dependent is the nearer verb."

The second advantage is high discriminative performance of contexts. Note that while the following discussion is applicable to both the 3-tuple and the 4-tuple models, we will discuss the 3-tuple model as a representative. The number of conflicts in the 3-tuple model is smaller than that in 2-tuple models, such as absolute preference models or the transition-based models. In the 3-tuple model, a conflict of training examples occurs only when a certain 3-tuple (d, c_1, c_2) is given. Unlike the relative

preference model, such examples conflict each other only if their class labels are different, and their feature vectors are the same. The 3-tuples $(kare\_wa, karita, kaeshita)$ and $(kare\_wa, kaeshita, karita)$ never conflicts in the direct comparison model while the relative preference model causes a conflict in the sentences.

Requiring an HPSG can be a disadvantage of the model. Manual grammar construction cancels the advantage of data-driven parsing of easy domain adaptation.

Single Candidate Model with Dynamic Feature C vs. Direct Comparison Model

Single-candidate models with the dynamic feature C can be regarded as a 3-tuple model. Since the dynamic feature corresponds to the head of a focused-on candidate head, the head can be regarded as the second candidate head of the focused-on dependent. However, compared to the 3-tuple model in the direct comparison model, the dynamic feature may provide no useful information for a classification of whether the pair of bunsetsus are in a dependency relation. On the other hand, the second candidate c_2 of the 3-tuple model (d, c_1, c_2) cannot be useless because the model performs the classification of which of c_1 and c_2 is the more likely candidate. If we regard dependency parsing as the task of selecting the best candidate head of a dependent among only *strong* candidates[8], the direct comparison model is a direct solution. In contrast, it is an indirect solution that shows the second candidate as the dynamic feature. The second candidate shown as the dynamic feature is not likely to be a strong candidate.

Providing useless features to a classifier can cause the overfitting problem. As described above, a conflict is caused with the training examples whose feature vectors are exactly the same but class labels differ. However, in practice, it is rare that two feature vectors are equivalent because of contextual features such as features corresponding to bunsetsus between a dependent and a candidate. Therefore, a conflict can be caused with training examples whose features are *similar* but class labels differ. If their unique features express the reason why their class labels differ, there is no conflict. A unique feature is a feature which is not commonly included in the two training examples. Even if their unique features do not express the reason, a classifier will assume that one of the unique features is the reason of the difference. It is impossible to make a correct de-

[8]Most of candidates of a dependent are never selected as its head in any context for some reasons such as their POSs. On the other hand, we refer to the candidates which can be selected in some contexts as *strong* candidates.

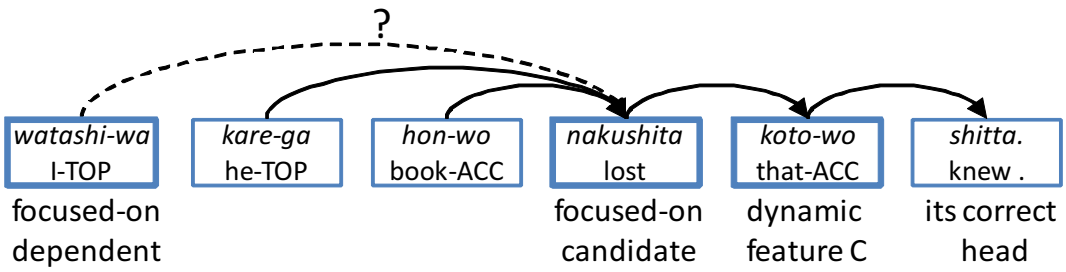


Figure 3.7: A typical case that the dynamic feature C works well. "I knew that he lost a book."

cision without showing the bunsetsus which are necessary to recognize the difference of the two contexts, even for human. While lack of information causes conflicts, it is an inappropriate solution for a classifier to provide plenty of contextual features. The classifier cannot estimate which feature is useful for classifications and then parsing accuracy will fall seriously.

According to the discussion, the direct comparison model is one of the best solutions that avoid conflicts with minimum number of features. In most cases, strong candidates are covered by an HPSG.

A typical case that the dynamic feature works well is for embedded sentences. An example is shown in Figure 3.7. The sentence "*kare-ga hon-wo nakushita*"(he lost a book) is embedded. When the focused-on dependent is "*watashi-wa*" and the focused-on candidate is "*nakushita*", the dynamic feature enables to see "*koto-wo*". Then, a single candidate model recognizes that "*nakushita*" is not the end of sentence but the end of an embedded sentence. With the clue, the model will decide that the pair of bunsetsus are not in a dependency relation and the correct head is probably farther away.

When both the focused-on candidate and its head are verbs, the dynamic feature can also work well as the second candidate. A parser recognizes there are at least two verbs and make a decision by comparing plausibilities of the candidates.

3.5 Related Work in Multilingual Dependency Parsing Literature

The shared tasks on multilingual dependency parsing took place at CoNLL-2006 [1] and CoNLL-2007 [44]. Many language-independent parsing models were proposed there. The shared task on joint learning of syntactic and semantic dependencies for English is also held at CoNLL-2008 [59].

There are two major types of dependency parsing models: transition-based and graph-based models. Transition-based models are similar to the classification-based models presented in Section 3.2. Yamada and Matsumoto [67] proposed a shift-reduce-like $O(n^2)$ transition-based model. Nivre et al. [41, 47] also proposed a shift-reduce-like $O(n)$ transition-based model. These transition-based models are for projective languages.

Nivre's model is enhanced for non-projective languages. Nivre and Nilsson [46] transforms a non-projective dependency tree into a projective tree and encodes non-projective information as dependency labels in preprocessing. The projectivized tree is input into a projective dependency parser, and its output is inverse-transformed. Nivre [43] proposed another solution: introducing the *Swap* action, which swaps two words inside of the stack.

Goldberg and Elhadad [9] proposed an easy-first transition-based parser. Unlike Nivre's parser, the parser is not restricted to left-to-right parsing. The model is slightly similar to the CC model; it iteratively chunks neighboring words in its workspace. Scores corresponding to dependency arcs are assigned to all neighboring word pairs, for each of left arc and right arc. The parser assigns the dependency arc whose score is the highest. The time complexity of the algorithm is $O(\log n)$ because of $O(n)$ heap insertions. However, the time complexity is virtually $O(n)$ because score calculation from a feature vector, which is the most time-consuming $O(1)$ processing, are performed $O(n)$ times.

Huang and Sagae [11] proposed a non-deterministic transition-based model. They showed dynamic programming is applicable to a certain sort of transition-based dependency parsers. The key of efficiency of their model is merging "equivalent" states and performing beam search. Equivalence between states is judged by small number of *kernel features*.

A *graph-based* model was proposed by McDonald et al. [37] which is based on searching of *maximum spanning trees (MSTs)* by the Chu-Liu-Edmonds algorithm [5,

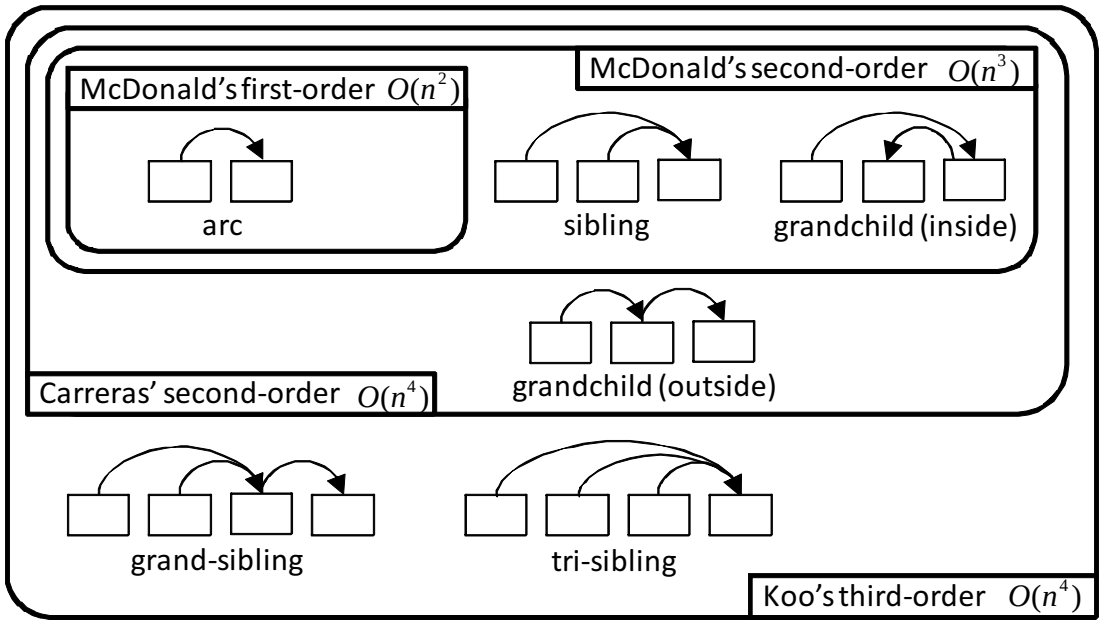


Figure 3.8: Summary of factorizations in and their time complexities of MST parsers for multilingual dependency parsing. A box is a word.

8]. The model factors the score of a dependency tree into the sum of scores of each dependency relations. The model first assigns scores for every pair of a dependent and each of its candidate heads, then obtains the best dependency tree, which maximizes the sum of scores of dependency relations. While the model assigns a score to each candidate dependency relation considering only a dependent and its candidate head, McDonald and Pereira [36] proposed to assign a score which considers a dependent, its candidate head and a sibling of the dependent. Carreras [2] proposed to consider also its grandchild. The original MST-based parser is referred to as *first-order* MST and the other two improved versions are referred to as *second-order* MST. Koo and Collins [22] proposed a third-order MST model which incorporates grand-sibling and tri-sibling interactions. Figure 3.8 summarizes the time complexities to decode the highest-scored dependency tree, and the subtrees which each of the MST parsers can assign features to.

Several methods were proposed that intend to improve parsing accuracy by a combination of several parsers. Sagae and Lavie [51] proposed a method to combine parsers' outputs by searching the MST of the graph such that the score of each arc corresponds

Type	Model	Time	Classifier	Acc2	Acc3
Absolute preference	S&U [57, 65] ([25, 27])	$O(n^2)$	ME	87.14	90.93
	Kudo [24] ([25])	$O(n^2)$	SVM	89.09	-
	Uchimoto [64] ([27])	$O(n^2)$	ME	87.93	91.09
Transition -based	Kudo [25] ([27])	$O(n^2)$	SVM	89.29	91.23
	Sassano [55]	$O(n)$	SVM	89.56	–
Indirect	Kudo [27]	$O(n^2)$	ME	–	91.37
	Yamamoto [70]	$O(n^2)$	PA	–	91.60
Direct	Kanayama [16]	*	ME	(88.33)*	–

Table 3.3: Summary of Japanese dependency parsing models. The columns represent the type and author of the parsing model, its time complexity, the classifier, and the parsing accuracies with Kyoto Text Corpora Version 2.0 and 3.0, respectively. Citations in parenthesis are the papers which reported the accuracy of the model with the corpora. S&U means Sekine and Uchimoto's model.

to the number of the dependency parsers which assigned a dependency relation between the pair of words. The method guarantees to obtain a well-formed dependency tree which maximizes the number of votes. Nivre and McDonald [45] proposed a simple combination method of two parsers. The second parser incorporates the output of the first parser as features such as the feature that whether there is a dependency relation between the two words in the first parser's output. The stacking method significantly improves parsing accuracy, especially when the two parsing models are very different. Martin et al. [35] performed further experiment of the method. A transition-based parser and a graph-based parser are usually employed as the first and the second parsers of the method.

3.6 Summary

Table 3.3 summarizes the Japanese dependency parsing models presented in this section. Most models were evaluated with Kyoto Text Corpus Version 2.0. The corpus was separated into three parts: 7,958 training sentences, 1,246 test sentences and the rest development sentences. Kudo and Matsumoto [27] evaluated four models with Kyoto Text Corpus Version 3.0: 24,263 training sentences, 9,278 test sentences and the

rest development sentences. Since Kyoto Text Corpus is a corpus compiled from newspaper articles and editorials, articles and editorials are blended in all parts. Features depend on each of experiments; not only parsing models but also feature templates have been refined.

The direct comparison model [16] was trained with approximately 190,000 sentences of EDR corpus. Parsing accuracy on Kyoto Text Corpus Version 2.0 is 87.08% and that on 3,372 sentences of EDR corpus is 88.33%. Since Kyoto Text Corpus is out-of-domain data and EDR is in-domain data for the trained model, the accuracy on EDR is probably fairer to compare with other models. The time complexity of the model depends on that of the HPSG parsing algorithm. The time complexity without the HPSG parsing is $O(n)$ because the number of candidates is constant.

Chapter 4

Japanese Dependency Parsing Using a Tournament Model

In this chapter, we present a tournament model for Japanese dependency parsing. The model is a sort of the direct comparison-based models described in Section 3.4; it considers relative preference in training, and selects the head of a dependent by classifications.

The tournament model was first introduced by Iida et al. [14] for coreference resolution. The model chooses the most likely candidate in a step-ladder tournament, which is a sequence of one-on-one games between candidate referents for a given anaphoric expression. In each game, the winner is chosen by a binary classifier such as an SVM.

Yamamoto and Masuyama [70] employed the tournament for the Japanese dependency parsing model which compares preferences of two candidate heads with considering their relative positions.

We also applied the tournament model to Japanese dependency parsing. Figure 4.1 illustrates a tournament. The focused-on dependent bunsetsu is "*kare-wa*", and the candidate heads are the three bunsetsus on its right-hand side: "*hon-wo*", "*yomanai*" and "*hito-da*". The first game is "*hon-wo*" vs. "*yomanai*". Then the next game is the winner of the first game vs. "*hito-da*". The winner of the second game (i.e., "*hito-da*") is chosen as the most likely candidate of the dependent, "*kare-wa*". In the tournament model, the most likely head of a given bunsetsu is determined by a series of one-on-one games in a tournament.

Below, we present training and parsing algorithms of the tournament model in Sections 4.1 and 4.2. Then, we discuss the advantages of the model in Section 4.3. Experiments conducted in Section 4.4 compare the tournament model to the previous

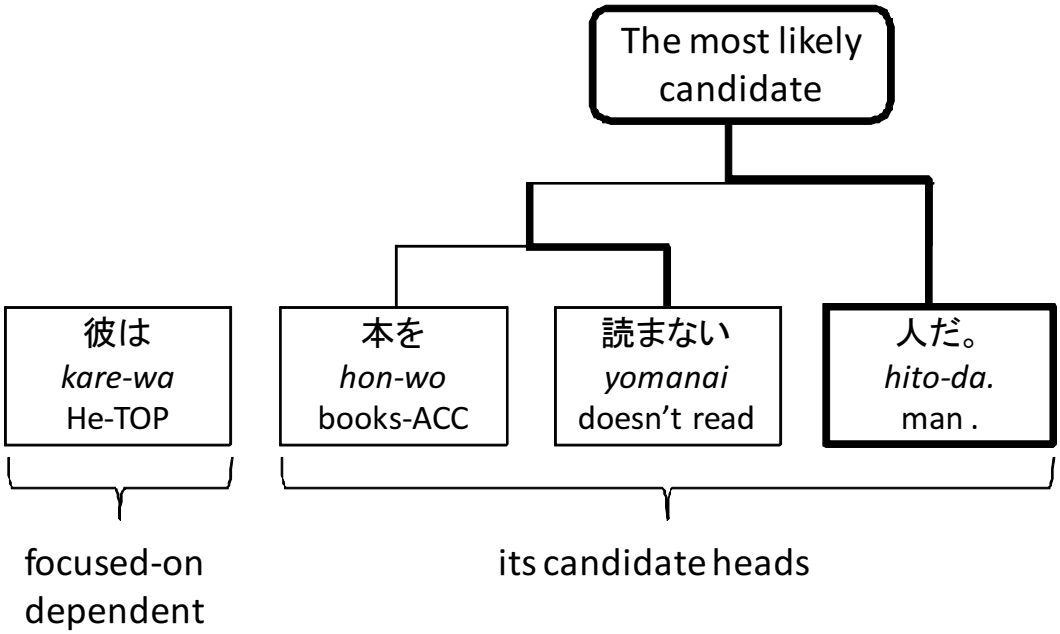


Figure 4.1: An example of a tournament.

models. We conduct quantitative and qualitative error analyses in Section 4.5. Finally, we present applications of the tournament model to multilingual dependency parsing in Section 4.6.

4.1 Training Example Generation Procedure

As shown in Algorithm 1, for each dependent, the model generates training examples for all pairs of its correct head and all other candidate heads. In the example generation, the procedure does not take into account PROJECTIVE constraint; all bunsetsus located on the right of the focused-on dependent are candidate heads.

Table 4.1 shows all the examples generated from the two sentences shown in Figure 3.1. As we discussed in Section 3.1, since 2-tuple models generate training examples in the form of (dependent, candidate), the examples generated from the pair (*kare-wa*, *hito-da*) are conflicted. On the other hand, the examples generated by the tournament model do not contain such inconsistency.

Algorithm 1 Training example generation procedure of the tournament model.

```
{N: The number of bunsetsus in input sentence}
{gold_head[j]: bunsetsu j's head in training data}
{gen(j,i1,i2,LEFT): generate an example where bunsetsu j is a dependent of i1
and is not a dependent of i2}
{gen(j,i1,i2,RIGHT): generate an example where bunsetsu j is a dependent of i2
and is not a dependent of i1}
for j = 1 to N − 1 do
    h = gold_head[j]
    for i = j + 1 to h − 1 do
        gen(j,i,h,RIGHT)
    end for
    for i = h + 1 to N do
        gen(j,h,i,LEFT)
    end for
end for
```

Sentence	Dependent	Left candidate	Right candidate	Class label
(a)	*kare-wa*	*hon-wo*	*hito-da.*	RIGHT
(a)	*kare-wa*	*yomanai*	*hito-da.*	RIGHT
(a)	*hon-wo*	*yomanai*	*hito-da.*	LEFT
(b)	*kare-wa*	*hon-wo*	*yomanai.*	RIGHT

Table 4.1: Generated training examples from the two sentences in Figure 3.1. The column "Dependent" is the focused-on dependent. Left candidate is the nearer candidate head. Right candidate is the farther candidate head.

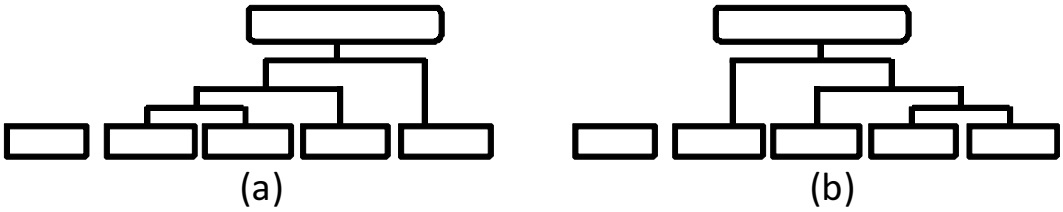


Figure 4.2: Tournament pairing schemes: (a) games are performed from left to right, or (b) from right to left.

4.2 Parsing Algorithm

The tournament model has quite wide freeness in its parsing steps. We introduce one of the tournament algorithms in which dependents are picked from right to left, and the games in a tournament are performed from left to right. The two tournament pairing schemes shown in Figure 4.2 correspond to the latter options. Also, this parsing algorithm takes into account PROJECTIVE constraint.

PROJECTIVE constraint is easily met. When the model enumerates the candidate heads for a focused-on dependent, the model only considers the candidates which do not introduce any crossing of dependency relations.

This algorithm is shown in Algorithm 2. The overall parsing process moves from right to left. On selecting the head of a dependent, the heads of all of the bunsetsus located on the right of the dependent have already been decided. In Algorithm 2, the array "`head`" stores the parsed results and ensures that only non-crossing candidate heads are taken into consideration.

Note that the structure of the tournament has little effect on the parsing accuracy (< 0.1) in our preliminary experiments which we will present in Section 4.4.4. We tried the 2×2 options: whether the dependents are picked from right to left or from left to right, and whether the games in the tournament are performed from right to left or from left to right. We choose the most natural combination for Japanese dependency parsing, which is easiest to implement.

Variations of Parsing Algorithm

As described above, the tournament model has quite wide freeness in its parsing steps. Focusing on the order of picking up dependents, as shown in Figure 4.3, left-to-

Algorithm 2 Parsing algorithm of the tournament model.

```
{N: The number of bunsetsus in input sentence}
{head[j]: (analyzed-) head of bunsetsu j}
{classify(j,i1,i2): ask SVM which candidate (i1 or i2) is more likely to be head of
j. Return LEFT if i1 wins. Return RIGHT if i2 wins.}
head = {2,3,...,N-1,N,EOS}
for j = N − 1 downto 1 do
    h = j+1
    i = head[h]
    while i ≠ EOS do
        if classify(j,h,i) == RIGHT then
            h = i
        end if
        i = head[i]
    end while
    head[j] = h
end for
return head
```

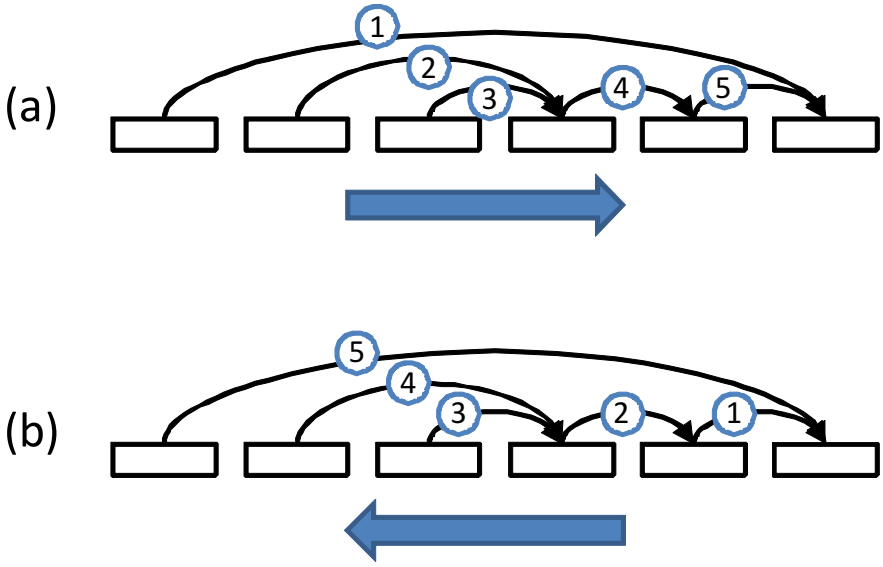


Figure 4.3: The order of picking up dependents: (a) left-to-right, or (b) right-to-left. Their identification order of dependency relations are drawn by the digits in circles.

right parsing is relatively top-down[1] and right-to-left parsing is relatively bottom-up[2] because the former identifies outermost relations first, and the latter identifies innermost relations first. If a model uses already-determined dependency relations as dynamic features or performs projective parsing, right-to-left parsing seems to be better. The parsing order first identifies short relations, whose accuracy is relatively high. However, their available dynamic features can also determine which parsing order is better. As described in Section 3.1, available dynamic features depend on the parsing order.

If the tournament model considers PROJECTIVE constraint, time complexity of the left-to-right parsing is $O(n^3)$ and that of the right-to-left parsing is $O(n^2)$. As Algorithm 2 does, the right-to-left parsing enumerates all candidate heads by traversing the outermost dependency relations. In contrast, since the left-to-right parsing cannot perform such an enumeration, it maintains the matrix that each element (j, i) represents whether b_j can modify b_i, instead. Therefore, time complexity of left-to-right parsing is $O(n^3)$.[3]

4.3 Advantages

The tournament model has two advantages compared to previous models. The model reduces the number of candidates in a classification without any grammar. Moreover, the classification problem is easier to solve.

Although characteristics of the tournament model is quite similar to the direct comparison model, a difference lies in the method to reduce the number of candidates. The direct comparison model uses an HPSG, which limits the upper bound of parsing accuracy. In contrast, the tournament model restricts the number of candidates of a game to exactly two by factorizing the direct comparison of all candidates into a series of one-on-one games. A tournament plays a role of strong candidate extractor instead of the HPSG in the direct comparison model. If there are at least two strong candidates, what the tournament model performs is almost the tournament among the strong candidates. Both the 3-tuple model in the direct comparison model and the tournament model are

[1] Constructing a dependency tree from the root to leaves.

[2] Constructing a dependency tree from leaves to the root.

[3] However, most of the parsing time is dominated by calculations in machine learners. The time for maintenance of the matrix is little. The main reason why the left-to-right parsing often spends more time than the right-to-left parsing is the number of classifications, rather than its time complexity, $O(n^3)$. The right-to-left parsing with PROJECTIVE constraint effectively reduces the number of candidates.

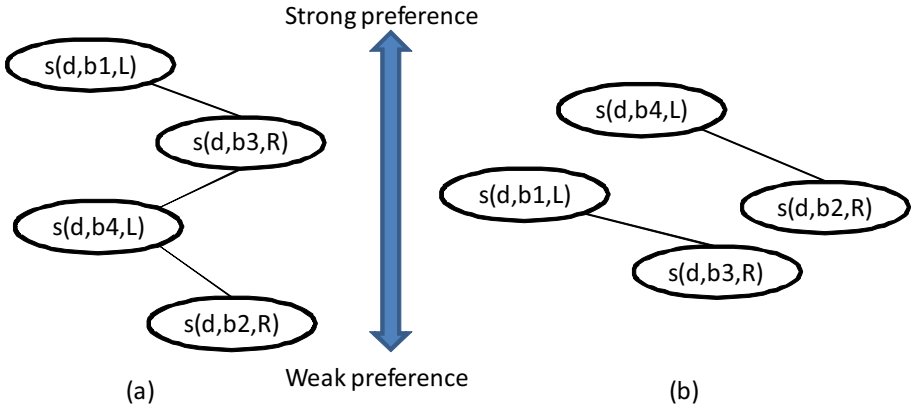


Figure 4.4: Ordering of the scores in the relative distance model. (a) The inequation $s(d,b_1,L) > s(d,b_2,R)$ is derived based on transitivity. (b) No partial ordering is defined between $s(d,b_1,L)$ and $s(d,b_2,R)$.

twin-candidate models. Therefore, exactly the same games will be performed by the two models if there are two strong candidates. If there are three strong candidates, which model is better: the 4-tuple model and the tournament model? It is the trade-off between data sparseness and context discriminative performance. Discriminative performance of the 4-tuple model is higher than the tournament model because of the third candidate. However, the model requires more training examples for learning sufficient parsing rules.

The binary classification which the tournament model performs is an easier problem than that performed by certain 2-tuple models such as absolute preference models and the transition-based models. As described in the discussion about the dynamic feature C, a binary classification of the direct comparison model is an easier problem than the binary classification whether a pair is in a dependency relation. The information of the second candidate cannot be useless for a classification because the classification is "which candidate is better?" The tournament model factors the problem of selecting the best candidate head into a series of easy binary classification problems.

Comparison to Relative Distance Model

Our model is similar to Yamamoto and Masuyama [70]'s model. Both the models can recognize relative positions of candidates and choose the most likely candidate by

a series of comparisons of two candidates. Our model is a classification-based twin-candidate model which can be interpreted as factorization of the direct comparison model. Their model is a preference-based single-candidate model which was derived as an extension of the relative preference model.

A difference of the two models in characteristics is appeared when they make a classification for an *unknown* candidate pair, which did not appear in their training data. In such a case, the tournament model makes a classification relying on generalization power of SVMs. In contrast, the relative distance model makes a classification based on transitivity of ranking constraints of scores. We refer to the inequations which the relative distance model learns as *ranking constraints* in the sense that a set of the inequations represents partial ordering among strengths of preferences for candidates. However, since a score is a scalar, the partial ordering is mapped into a total ordering.

Suppose each of two candidate head bunsetsus b_1 and b_2 of the dependent d is known but the pair of candidates is unknown. That is, in training, the tournament model generated examples for the 3-tuples (d, b_1, b_3) and (d, b_4, b_2) but there were no examples such that (d, b_1, b_2). Note that we denote $\mathbf{w} \cdot F(\langle d, b_1 \rangle, L)$ as $s(d, b_1, L)$. Similarly, for example, the relative distance model generated the ranking constraints of scores such that $s(d, b_1, L) > s(d, b_3, R)$ and $s(d, b_4, L) > s(d, b_2, R)$ but there were no direct ranking constraints between $s(d, b_1, L)$ and $s(d, b_2, R)$.

If the dependent d and the candidates b_1 and b_2 are appeared in parsing, the relative distance model estimates which candidate is better based on transitivity of their scores. For example, as shown in Figure 4.4, the model can derive the inequation $s(d, b_1, L) > s(d, b_2, R)$ if the model learned the ranking constraints such that $s(d, b_1, L) > s(d, b_3, R)$, $s(d, b_3, R) > s(d, b_4, L)$ and $s(d, b_4, L) > s(d, b_2, R)$. Such inference can be performed only if certain intermediate candidate pairs such as b_3 and b_4 were appeared in the training data. If there were no such candidate pairs, the scores $s(d, b_1, L)$ and $s(d, b_2, R)$ are caluculated with no ranking constraints. However, the model makes a classification based on the scores even if no partial ordering is defined between the scores.

The relative accuracy of the two models depends on the appropriateness of the transitivity assumption and the mapping scheme from partial ordering into total ordering.

Yamamoto and Masuyama [70] conducted experiments for comparing their relative distance model and our model with the features which CaboCha, a Japanese dependency parser software, employs. The two models are comparable in their parsing accuracies. With the polynomial kernel expended representation (PKE) [26], their model is more compact than our model in the sense that the number of features tend to be

smaller and its parsing speed tends to be faster. PKE is an approximate method for fast combinational feature computation by ignoring combinational features whose weight or the number of occurrences is sufficiently low. Accuracy of our model tend to decrease by ignoring a combinational feature even if its weight is quite small.

Since our model directly considers two candidates, our model can be more accurate than their model. In their report, our model is more accurate for dependency relations whose distances between their dependents and heads are 1. Especially, our model tends to successfully parse dependency relations corresponding to idioms; when the focused-on dependent and the left candidate head in our model captures two bunsetsus which may constitute an idiom, the model can decide whether it is an idiom or not by considering the right candidate head, and then make an appropriate decision. For example, an idiomatic translation of "(noun)*to shite*" is "as" but its non-idiomatic translation is "do with (noun)". Another example is "*sakunen*(last year) *juunigatsu*(December)"; "*sakunen*" modifies "*juunigatsu*". If the sentence did not include "*juunigatsu*", the bunsetsu "*sakunen*" would modify a verb.

4.4 Experiments

4.4.1 Settings

Corpus

We implemented the tournament model, the SR model [55] and the CC model [25] with SVM classifiers. We evaluated dependency accuracy and sentence accuracy on Kyoto Text Corpus Version 4.0. Dependency accuracy is the percentage of correct dependencies out of all dependency relations. Sentence accuracy is the percentage of sentences in which all dependencies are determined correctly. Dependency accuracy is calculated excluding the rightmost bunsetsu of each sentence.[4] Sentences that consist of one bunsetsu were not used in our experiments. Sentences that consist of two bunsetsus were used but no classifications or score comparisons will be performed.

We used January 1st to 8th (7,587 sentences) for the training data. We used January 9th (1,213 sentences), 10th (1,479 sentences) and 15th (1,179 sentences) for the test data. These numbers of sentences exclude the sentences consisting of one bunsetsu.

[4]Most research such as Kudo's [27] used this criteria.

Support Vector Machines

We use TinySVM as the binary classifier. We employed the third degree polynomial kernel for its kernel function. The cost of constraint violation was set to 1.0. These SVM settings are the same as previous research [25, 55]. All experiments were conducted on Dual Core Xeon 3GHz x 2 Linux machines.

Features

Here we describe the features used in our experiments. Note that for the tournament model, the features corresponding to candidates are created for each of the nearer and the farther candidates. We define the *information* of a word as the following features: lexical form, coarse-grained POS tag, full POS tag and inflection form. We also define the *information* of a bunsetsu as the word *information* of each of *syuji* and *gokei* of the bunsetsu. *Syuji* is the head content word of a bunsetsu, defined as the rightmost content word[5]. *Gokei* is the representative function word of a bunsetsu, defined as the rightmost function word[6].

The *information* of a bunsetsu also includes the existence of punctuations or brackets, the indicator functions of whether the bunsetsu is the leftmost bunsetsu in the sentence, and whether it is the rightmost bunsetsu in the sentence.

We classified our features into the three groups: standard, additional and case particle features. The standard feature has been used in much previous research. The additional feature was introduced by Sassano [55]. The case particle feature was introduced by us [15]. The features corresponding to an already-determined dependency relation are referred to as *dynamic* features, and the other features are referred to as *static* features. The standard and the additional features are static features, and the case particle features are dynamic features. Whether a dynamic feature is available for a parsing algorithm depends on its parsing order. Details of features are described in Appendix A.

The *standard* features are the following: *information* of the dependent and the candidate heads, the distance between the dependent and the candidate heads (1, 2-5 or 6+ bunsetsus), all punctuations, brackets and all particles between the dependent and the candidate heads.

[5] A content word is a word whose POS is neither *symbol*, *particle* nor *suffix*.

[6] A function word is a word whose POS is not *symbol*.

Model	Features	Jan. 9th	Jan.10th	Jan. 15th
Tournament	Standard	89.89/49.63	89.63/48.34	89.40/49.70
	All	**90.09/49.71**	**90.11/49.02**	**90.35/52.59**
SR [55]	Standard	88.18/45.92	88.80/44.76	88.03/47.24
	All	89.22/47.90	89.79/47.87	89.55/49.79
CC [25]	Standard	88.17/45.92	88.80/44.76	88.00/47.24
	All	89.22/47.90	89.80/47.94	89.53/49.79

Table 4.2: Dependency accuracy and sentence accuracy [%] using 7,587 sentences as training data.

The *additional* features are the following: all case particles in the dependent and the candidate heads, *information* of the leftmost word in the candidate heads, and the lexical form of every candidate head's neighboring bunsetsu on the right.

The *case particle* features are the following: all case particles appearing in the candidates' dependent. These features are intended to capture the correlation between the case particles in the dependent. The dependent and a candidate's dependent become siblings if a dependency relation is established between the dependent and the candidate. When the candidate head is a verb, the features have a similar effect to learning case frame information.

4.4.2 Parsing Accuracy with Small Data Set

Table 4.2 summarizes the parsing accuracies of our model and previously proposed models. By McNemar test ($p < 0.01$) on the dependency accuracies, the tournament model significantly outperforms the other models except for the SR model on January 10th data with all features ($p = 0.083$) and the CC model on January 10th data with all features ($p = 0.099$). The difference between the tournament models with all features and with the standard feature only is significant except for on January 9th data ($p = 0.25$).

Our model outperforms the SR model, which achieved the highest dependency accuracy for January 9th of Kyoto Text Corpus Version 2.0. Sassano [55] reported the accuracy of the SR model as 89.56%.[7] Since we do not have the outputs of Sassano's

[7]This accuracy in Sassano [55] is not on Kyoto Text Corpus Version 4.0 but Version 2.0. It cor-

Feature	Model	# Step	Time[s]	# Example	# Feature	# SV
Standard	Tournament	26,438	300	374,579	56,169	42,079
	SR [55]	15,582	63	94,669	37,179	29,830
	CC [25]	18,879	80	112,759	37,179	30,728
All	Tournament	26,396	406	374,579	157,057	46,043
	SR [55]	15,641	87	94,669	90,972	34,443
	CC [25]	18,922	107	112,759	90,972	35,251

Table 4.3: Parsing times and the sizes of the training examples. Each column represents the feature set, the parsing model, the number of SVM classification steps, parsing time in seconds, the number of training examples, the number of features, and the number of support vectors, respectively.

experiments, we cannot apply a McNemar test between the tournament model and Sassano's results. Our model outperforms Sassano's results in dependency accuracies, but the difference between these two is not significant by prop test ($p = 0.097$).

When we add the additional and the case particle features, the improvement of the tournament model is less than that of other models. This is interpreted as that our model can consider richer contextual information within the model itself than other models.

This result also shows that the accuracies of the SR model and the CC model are comparable when their features are same. However, this does not mean that their substantial powers are comparable because their parsing orders limit their available dynamic features.

4.4.3 Parsing Speed

Table 4.3 shows the times to parse all sentences of January 9th and the sizes of the training examples. The column "# Step" represents the number of SVM classification steps in parsing all sentences in the test data. The time complexities of the tournament model and the CC model are $O(n^2)$ and that of the SR model is $O(n)$. The tournament model requires 1.7 times more SVM classification steps and is 4-5 times slower than the SR model. The reason for this difference in steps (x1.7) and time (x4-5) is the

responds to Acc2 in Table 3.3. The feature set of Sassano's experiment is also different from our experiment.

number of support vectors and features in the SVM. Since the number of training examples is quite large, the SVM model in the tournament model is more complex and the number of its support vectors is larger than SVMs in other models.

4.4.4 Variants of Parsing Algorithm

As noted in Section 4.2, the tournament model has quite wide freeness in its parsing steps: the order to pick up dependents, the pairing scheme of tournament games, and whether PROJECTIVE constraint is assumed. We varied these settings and compared their accuracies. We also varied whether to incorporate the case particle features, which are the only dynamic features. The result is shown in Table 4.4. Note that if we incorporate neither the case particle features nor PROJECTIVE constraint, each dependency relation is independent each other. Thus, the choice of the order to pick up dependents has no effect on the parsing result.

Differences in the accuracies are small: most of them are around 0.1%. The best setting in parsing accuracy is assuming PROJECTIVE constraint, picking up dependent from right to left, pairing tournament from right to left and incorporating the case particle features across all the test data. Using the case particle features improves the accuracies in most settings but the improvement is small. Note that we can share a single SVM among settings if we do not use the case particle features or we do not change the order to pick up dependents. This is because the availability of the dynamic features B_L or B_R (defined in Section 3.1), corresponding to the case particle features, depends on the order to pick up dependents: the dynamic feature B_L is available only if the order is left to right and B_R is available only if the order is right to left.

4.4.5 Comparison to Relative Preference and Relative Distance Models

We conducted another experiment under the same settings as Kudo's [27] to compare the tournament model and the relative preference model. The corpus is *Kyoto Text Corpus Version 3.0* since Kudo and Matsumoto [27] used this corpus. The training data is the articles from January 1st to 11th and the editorials from January to August (24,263 sentences). The test data is the articles from January 14th to 17th and the editorials from October to December (9,287 sentences). We did not perform parameter engineering with the development data, although Kudo and Matsumoto [27] performed

Feature	Order	Pair	Proj.	Jan. 9th	Jan. 10th	Jan. 15th
All	$\rightarrow$	$\rightarrow$	Yes	89.96/49.55	90.06/48.95	90.11/50.98
			No	89.90/49.38	90.06/48.68	90.12/50.89
		$\leftarrow$	Yes	89.97/49.05	90.18/49.09	90.18/51.15
			No	89.94/48.89	90.17/48.88	90.19/50.98
	$\leftarrow$	$\rightarrow$	Yes	90.09/49.71	90.11/49.02	90.35/52.59
			No	90.10/**49.79**	90.08/48.95	90.29/52.16
		$\leftarrow$	Yes	**90.11**/49.63	**90.19/49.43**	**90.42/52.67**
			No	**90.11**/49.46	90.18/49.29	90.39/52.33
Std&Add	$\rightarrow$	$\rightarrow$	Yes	90.02/49.79	90.06/49.22	90.09/50.89
		$\leftarrow$		90.04/49.38	90.13/49.09	90.14/50.89
	$\leftarrow$	$\rightarrow$		90.01/49.79	90.06/48.95	90.12/50.89
		$\leftarrow$		90.04/49.55	90.14/49.02	90.21/51.06
	-	$\rightarrow$	No	90.01/49.71	90.06/48.95	90.12/50.81
		$\leftarrow$		90.03/49.30	90.11/48.88	90.17/50.72

Table 4.4: Accuracies when we varied several options. Each column represents the feature set (all, or standard and additional), the order to pick up dependents ($\rightarrow$: left to right, $\leftarrow$: right to left), the pairing scheme of tournament games, whether the PROJECTIVE constraint is assumed, and accuracies for the three test data, respectively.

Model	Features	Dep. Acc.	Sentence Acc.
Tournament	All	**91.96**	**57.44**
Relative distance [70]	All	91.80	56.88
SR [55]	All	91.48	55.67
CC [25]	All	91.47	55.65
Relative distance [70] ([68])	Kudo's [27]	91.60	56.33
Combination [27]	Kudo's [27]	91.66	56.30
Relative preference [27]	Kudo's [27]	91.37	56.00
CC [25]	Kudo's [27]	91.23	55.59

Table 4.5: Dependency and sentence accuracies [%] with the training data of 24,263 sentences with all features. The model "Combination" is the combination model of the CC and the relative preference models.

it. The criteria for dependency accuracy are the same as the experiments above. However, in this subsection, a sentence accuracy is evaluated with all sentences regardless of their sentence lengths, as Kudo and Matsumoto [27] did.

The results are shown in Table 4.5. Note that Kudo and Matsumoto's [27] and our feature sets are different.[8] Only the CC model is evaluated with both the feature sets. Our feature set looks better than Kudo's. By McNemar test ($p < 0.01$) on their dependency accuracies, the tournament model outperforms both the SR and the CC models significantly. Since we do not have the outputs of the relative preference models, we cannot apply a McNemar test between the tournament model and the relative preference models. By prop test ($p < 0.01$) on their dependency accuracies, our model significantly outperforms the relative preference model. Though the tournament model outperforms the "combination" model which Kudo and Matsumoto [27] proposed in their dependency accuracies, the difference between these two is not significant by prop test ($p = 0.014$).[9]

We also implemented the relative distance model with Ranking SVM[10]. By McNemar test ($p < 0.01$) on the dependency accuracy, the tournament model significantly outperforms the relative distance model. The tournament model have the advantage that the model directly considers two candidate heads.

Note that Kudo's experiment employed an ME model as a machine learner. It has shorter training time than an SVM. However, an ME model requires feature combination engineering, while an SVM automatically considers combinations of features by polynomial kernels.

[8] The best feature set depends on a parsing model. Incorporating a feature sometimes decreases its parsing accuracy. For the tournament model, Kudo's [27] dynamic feature C decreases its accuracy. In contrast, Sassano's [55] dynamic feature A is also beneficial for the tournament model with left-to-right parsing; the dynamic feature corresponds to the immediate left bunsetsu of the focused-on dependent, and the feature is incorporated only if the bunsetsu modifies the focused-on dependent. Note that we did not use these dynamic features in experiments.

Yamamoto and Masuyama evaluated their relative distance model not only with Kudo's [27] features [68] but also with the features which CaboCha's feature extraction script outputs [70].

[9] The "combination" model is the combination of the CC model and the relative preference model. In Kudo's experiment, whereas the relative preference model outperformed the CC model for long-distance dependency relations, it is reversed for short-distance relations. They determined the optimal combination (the threshold was set to bunsetsu length 3) with the development set. In our experiment, the tournament model outperforms the CC and the SR models for relations of all lengths. Therefore, the tournament model does not need such an ad hoc combination.

[10] We used SVM^{light}, an implementation of SVMs.

4.5 Parsing Error Analysis

We conduct both quantitative and qualitative parsing error analyses of the output of the tournament model.

4.5.1 Frequency-based Error Analysis

In this subsection, we analyze parsers' outputs and illustrate what kind of dependency relations tend to cause a parsing error. The target data is the output of the tournament and the CC models generated in the experiment in Section 4.4.2. The target data is all sentences of January 9th, 10th and 15th. We ignore the rightmost two bunsetsus of each sentence because there are always correct.

Dependency Labels

First, we analyze the target data focusing on their dependency labels: D, P, A and I. In all the experiments in Section 4.4, we never use the labels for both training and testing. However, there will be a bias. For example, it is said that a coordinated structure is hard to parse correctly.

Table 4.6 shows the numbers of bunsetsus and their accuracies for each label. The distribution of labels in the training data is similar to that in the test data. The distribution of labels is strongly biased: about 90% of dependency relations have the label D. Characteristics of a pair which is in a dependency relation depend on its label. If there are a small number of bunsetsus which have a certain label, it is hard for a parser to learn the characteristics of the label. The tournament model outperforms the CC model in labels D, P and I. In particular, in label P, the difference between them is about 2%.

The reason of this great improvement can be that the tournament model can see both sides of the left candidate. That is, the tournament model implicitly estimates whether the two candidates are in a parallel (label P) relation. The tournament model can recognize that the current context is likely to be a coordinated structure. In contrast, the CC model cannot recognize it implicitly because the model is a single-candidate model.

The CC model is more accurate for label A. The reason may be that the tournament model needs more training examples than the CC model because the former is a twin-candidate model and the latter is a single-candidate model.

Label	#(Train)	#(Test)	Tournament	CC
D	53,363	28,282	**90.34**	89.76
P	5,462	2,683	**77.23**	75.29
A	242	345	75.36	**75.94**
I	190	60	**41.67**	26.67
Total	59,257	31,370	88.96	88.25

Table 4.6: Accuracy and distribution classified by dependency labels. D: normal dependency, P: parallel (coordination), A: apposition, I: incomplete coordination. "#(Train)" is the number of instances of corresponding label appeared in the training data. "#(Test)" is that in the test data.

For label I, the tournament model outperforms the CC model. The reason is probably similar to that of label P. Dependency relations with label I are unnatural dependency relations, as we will discuss in Section 8.1.2. Thus, a pair of bunsetsus with label I can be in a dependency relation only in an incomplete coordinated structure. Moreover, since there is quite few dependency relations with the label I, the training examples corresponding to such relations are rarely applicable to parsing a new dependency relations. Since the CC model decides whether a pair of bunsetsus is in a dependency relation, the model will decide that a pair of bunsetsus with gold-standard label I is not in a dependency relation. As a result of such decisions, the model attaches the dependent of label-I relation to the rightmost bunsetsu or a farther candidate which is plausible as the head of a label-D relation, because the model examines closer candidate heads first. In contrast, the tournament model may recognize existence of dependency relation whose label is I by recognizing that current context is in an incomplete coordinated structure.

Distance between Dependent and Candidate Head

Table 4.7 shows the analysis focusing on the distance between a dependent and its correct head. Distance 1 means they are neighboring. About 60% of Japanese bunsetsus modify their neighbors on the right. Such bunsetsus are relatively easy to parse and their accuracy is high. The longer a distance, the lower its accuracy. However, the accuracy of very long relations such as "20+" is relatively high because the rightmost bunsetsu of a sentence is one of the popular heads in Japanese dependency structures.

Distance	#(Train)	#(Test)	Tournament	CC
1	34,632	18,666	**96.49**	95.67
20+	207	116	**84.48**	83.62
2-3	13,861	7,056	**80.16**	79.71
15-19	463	289	**77.16**	**77.16**
4-5	4,841	2,478	**75.87**	75.42
8-9	1,330	773	74.90	**75.29**
10-14	1,497	732	**74.73**	72.13
6-7	2,426	1,260	**72.38**	71.83

Table 4.7: Accuracy and distribution classified by dependency distance between a dependent and its head. The distance bin "20+" means "20 bunsetsues or more."

Parts of Speech

Table 4.8 shows the analysis focusing on the syuji[11] POS of a dependent. The tournament model is more accurate for adjectives, nouns and verbs. The CC model is more accurate for demonstratives, adnominals, conjunctions, adverbs and no-syuji bunsetsus[12]. The POSs for which the tournament model is better are relatively frequent ones. The reason may be that a twin-candidate model requires more training examples than single-candidate models to exploit its performance.

Unknown Words

We focus on to what extent the accuracy depends on whether syuji words of a dependent and/or candidate bunsetsus are unknown. If the words are known, that is, the words appeared in the training data, a classification can be performed using their lexical information. Otherwise, a classification should be performed by using coarser features such as POSs. For every sentence in the training data, we extract all pairs of syujis (s_j, s_i) as known pairs where s_j is the syuji of j^{th} bunsetsu and $j < i$. Note that while the tournament model considers all such pairs when it generates training examples, the CC model may not consider some of the pairs because the model deterministically generates training examples.

[11] In this analysis, POSs that cannot be syuji are not only symbols, particles and suffixes but also copulas and auxiliaries.

[12] There are bunsetsus containing no word satisfying the condition to be syuji.

Dependent POS	#(Train)	#(Test)	Tournament	CC
形容詞 adjective	2,772	1,571	**91.73**	91.02
指示詞 demonstrative	1,288	698	90.97	**91.12**
連体詞 adnominal	168	121	90.08	**92.56**
名詞 noun	41,496	21,515	**89.93**	89.09
接続詞 conjunction	517	265	87.92	**88.30**
動詞 verb	11,221	6,142	**85.66**	84.96
副詞 adverb	1,686	986	83.37	**84.08**
No syuji	109	72	79.17	**81.94**

Table 4.8: Accuracy and distribution classified by dependent POS.

Known	#(Train)	#(Test)	Tournament	CC
DH	59,255	8,396	**90.26**	89.80
Dh,dH	2	13,447	**89.14**	88.23
dH	0	4,110	**88.69**	88.37
dh	0	1,240	**87.18**	85.65
Dh	0	4,177	**86.55**	85.88

Table 4.9: Accuracy and distribution classified by unknown words.

Table 4.9 shows the analysis focusing on whether the syujis of a dependent and/or a candidate are unknown. Hereafter, we use the term "a bunsetsu is unknown" as "the syuji of a bunsetsu is unknown". When a syuji is unknown, there are two cases: the POS of the syuji is known, or its POS is also unknown. The column "Known" in the table represents whether their syujis are unknown or not. If both the focused-on dependent and its correct head are known, it is denoted as *DH*. If the focused-on dependent is known, but its correct head is unknown, and the POS of the dependent's correct head is known, it is denoted as *Dh*. The two are opposite each other: *Dh* and *dH*. If there are training sentences satisfying *Dh* and *dH* but there were no sentence satisfying *DH*, it is denoted as *Dh,dH*. If both syujis are unknown but both POSs are known, it is denoted as *dh*.[13] There were no bunsetsus which their POS are unknown.

The result matches our intuition that unknown word degrades the accuracy of a clas-

[13]There are two *Dh,dH* sentences in the training data because of annotation mistake. Their heads are annotated to be themselves.

sification. The accuracy of *DH* is more than 1% higher than the others. However, 1% is not quite a large difference. The reasons may be that not only syuji but also gokei play important roles for a classification, and there are difficult examples that cannot be classified without contextual features other than the information of dependent and head bunsetsus.

A surprising point is that *dh* wins *Dh* with the tournament model. That is, if the head of a dependent is unknown, the dependent should be also unknown. Notice that since the term "head" here is the correct head, the other candidate heads may be known. When the syuji are unknown, a classification will be performed by using coarser features such as POSs. It is important for high-accuracy dependency parsing to compare strong candidates and then select the best one. Therefore, if a candidate head is unknown and there are several unknown strong candidates whose POSs are identical to that of the correct head, it is hard for the tournament model to discriminate and select the best candidate. In another case, if the correct head is unknown and the other strong candidates are known, the tournament model may select an incorrect strong candidate.

Table 4.10 shows the analysis focusing on unknown syujis and labels. The criteria whether a syuji is unknown is the same as Table 4.9, that is, whether a syuji is unknown does not depend on its label. The tendency of the label D is similar to Table 4.9 but that of the rest labels are dissimilar. For the label P, syujis should be unknown especially for head bunsetsus. Since our experiments never consider labels and D is the most frequent label, a resulting SVM model is specialized for parsing label-D relations. Another interpretation is that since there are not sufficient training examples corresponding to label P, a resulting SVM model classifies an example based on not lexical forms but other features such as POSs or commas. All labels have the tendency that if the head of a dependent is unknown, the dependent should be also unknown. For label A, while difference among *DH*, *Dh* and *dh* are small, the accuracies of *Dh,dH* and *dH* are quite low. For label I, it is hard to interpret the tendency because there are few examples.

4.5.2 Typical Errors

In this subsection, we present several examples of typical parsing errors. We will present not full dependency trees of example sentences picked up from Kyoto Text Corpus but their subtrees around the focused-on dependency relations.

Label	Known	#(Train)	#(Test)	Tournament	CC
D	DH	53,361	7,706	**91.41**	91.15
D	dH	0	3,695	**90.61**	90.34
D	Dh,dH	2	12,194	**90.50**	89.73
D	dh	0	948	**89.66**	88.50
D	Dh	0	3,739	**87.51**	86.76
P	dh	0	265	**79.62**	77.74
P	Dh	0	362	**78.45**	**78.45**
P	Dh,dH	0	1,084	**77.68**	74.91
P	DH	5,462	610	**76.72**	74.10
P	dH	0	362	**73.76**	73.48
A	DH	242	72	**84.72**	**84.72**
A	Dh	0	64	84.38	**85.94**
A	dh	0	19	**84.21**	**84.21**
A	Dh,dH	0	144	69.44	**70.83**
A	dH	0	46	**63.04**	60.87
I	DH	190	8	**62.50**	37.50
I	dh	0	8	**50.00**	12.50
I	Dh	0	12	**41.67**	33.33
I	Dh,dH	0	25	**40.00**	32.00
I	dH	0	7	**14.29**	0.00

Table 4.10: Accuracy and distribution classified by labels and unknown words.

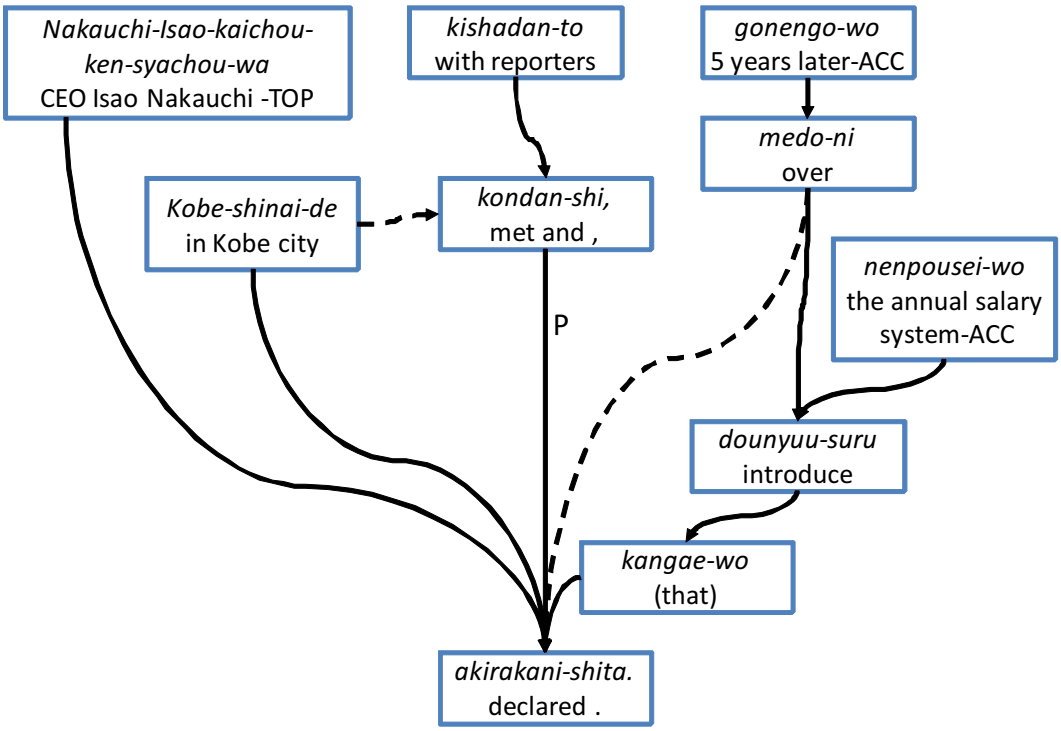


Figure 4.5: An example sentence which contains parsing errors corresponding to *renyoh chuushi-ho*. Each box is a bunsetsu. A solid arrow is a correct dependency relation and a dotted arrow is an incorrect dependency relation which was output by parsers. The arrow with "P" is a parallel dependency (label P). The others are normal dependencies (label D).

VP modifies VP

Kurohashi and Nagao [30] pointed out that long sentences often contain coordinated structures. Especially, they often contain a predicate coordinated structure, which is constructed as a sequence of conjunctive predicate clauses by using the form of *renyoh chuushi-ho*. Figure 4.5 is the dependency tree of the sentence:
Japanese: 中内功会長兼社長は　神戸市内で　記者団と　懇談し、五年後を　めどに　年俸制を　導入する　考えを　明らかにした。
Roman: *Nakauchi-Isao-kaichou-ken-shachou-wa Kobe-shinai-de kishadan-to kondan-shi, gonengo-wo medo-ni nenpousei-wo dounyuu-suru kangae-wo akirakani-shita.*
English: CEO Isao Nakauchi met with reporters and declared that the annual salary

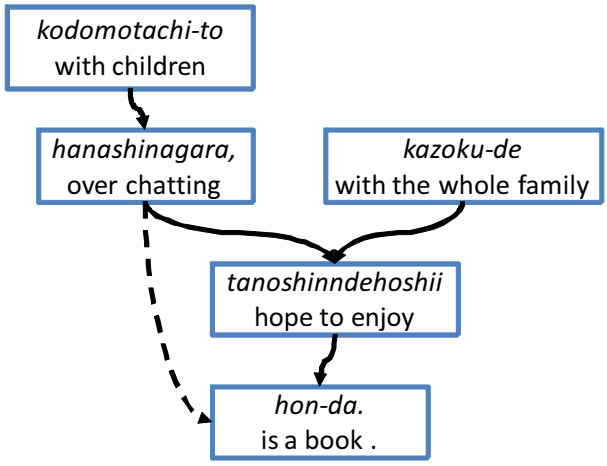


Figure 4.6: An example sentence which contains a parsing error corresponding to an embedded sentence.

system will be introduced over 5 years, in Kobe city.
The spaces in the Japanese sentence and the Roman transliteration are bunsetsu boundaries. The bunsetsu "*kondan shi*," is an example of *renyoh chuushi ho*; the predicate ends with *renyoh* form. While the correct head of "*Kobe-shinai-de*" is "*akirakani-shita*", parsers decided that its head is "*kondan-shi*". The difference in their meanings is whether Mr. Nakauchi did only meeting (*kondan*) or did both meeting and declaration (*akirakani-shita*), in Kobe city. It is an attachment problem that which activities were done in Kobe city. This example is easy to parse by human because of the temporal argument "over 5 years". However, there are more difficult cases. Such cases are either truly ambiguous cases or cases that need wider context for decisions.

Commas (、) tell a reader that the bunsetsu with comma modifies not the nearest possible candidate head but a farther bunsetsu. Therefore, commas are significant clues in Japanese dependency parsing. However, when parsing a long sentence, a comma is not sufficient to narrow down candidate heads because there are several strong candidates which are far from its dependent. In addition, to what extent a comma makes a dependency relation longer may depend on the author of a text.

Embedded sentence

If a sentence contains an embedded sentence, the head of a bunsetsu often ambiguous: whether the dependent modifies the rightmost predicate of the embedded sentence, or one of its subsequent bunsetsus. Figure 4.6 is the dependency tree of the sentence:
Japanese: 子供たちと　話しながら、　家族で　楽しんで欲しい　本だ。
Roman: *kodomotachi-to hanashinagara, kazoku-de tanoshindehoshii hon-da*.
English: It is a book which I hope you to enjoy with the whole family over chatting with children.
The embedded sentence is "*kodomotachi-to hanashinagara, kazoku-de tanoshindehoshii*" (I hope you to enjoy with the whole family over chatting with children). While the correct head of "*hanashinagara,*" (over chatting) is "*tanoshindehoshii*" (hope to enjoy), the parsers decided that its head is "*hon-da*" (is a book). While the correct head is the nearer one in this example, one of subsequent bunsetsus is the correct head in many cases. Another reason of this error is the comma in this sentence. It is misleading since a comma means a long relation, as noted above. Note that both syuji and gokei of the bunsetsu "*tanoshinode-hoshii*" is "*hoshii*". The words "*tanoshinde*" were not used in a classification. The word "*hoshii*" means "want (something)" if it appears alone. Therefore the dependency relation "*-nagara hoshii*" (want something with doing) may be decided inappropriate.

Richer Context Required

The sentence shown in Figure 4.7 needs more context information to parse correctly.
Japanese: 今月　公表した　女性の　人権侵害全般に　ついての　予備報告書の　中で、...。
Roman: *kongetsu kouhyoushita josei-no jinkenshingai-zenpan-ni tsuiteno yobihoukokusyo-no naka-de, ...*
English: In the preliminary statement on overall human rights violation against women which (someone) published this month...
While the correct head of "*kouhyou-shita*" (published) is "*yobihoukokusyo-no*" (preliminary statement), the tournament model decided that its head is "*josei-no*" (woman's) and the CC and the SR model decided that its head is "*jinkenshingai-zenpan-ni*" (overall human rights violation). While the head that the CC and the SR models selected is

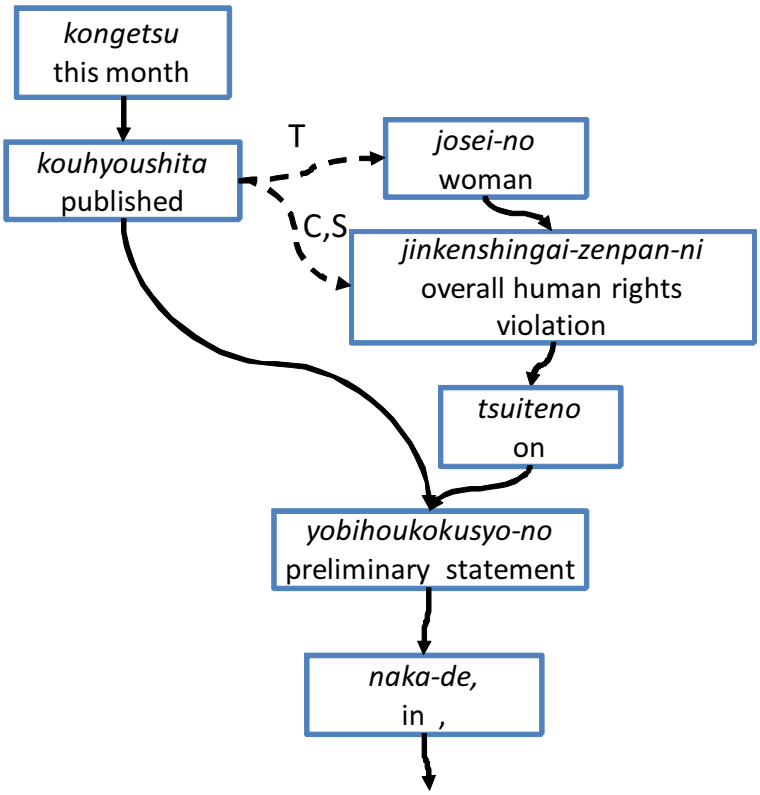


Figure 4.7: An example sentence which contains a parsing error which requires a richer context. The dotted arrow with "T" is a mistake by the tournament model. The dotted arrow with "C,S" is a mistake by the CC and the SR models.

something strange, the dependency relation output by the tournament model is plausible. If a human sees only the relation itself, the human will judge it to be plausible. The relation is locally plausible but globally wrong.

A possible clue to determine whether the relation is globally correct is the set of case elements modifying the dependent. If a *wo*-case element modifies the dependent, the correct head is probably "*josei-no*" because "*-wo kouhyoushita josei*" (the woman *who* published something) is plausible but "*-wo kouhyoushita yobihoukokusyo*" (the preliminary statement *who* published something) is not plausible. In contrast, if a *ga*-case element modifies the dependent, the correct head is probably "*yobihoukokusyo-no*" because "*-ga kouhyoushita yobihoukokusyo*" (the preliminary statement *which* someone published) is plausible. This sort of contextual information can be considered as the

dynamic feature A if a parser performs left-to-right parsing. However, no case elements modify the dependent in this example sentence. We can say that the correct head is "*yobihoukokusyo-no*". This may be because of the set of all arguments. That is, the dependent does not have any case elements but has a temporal argument.

As described above, one of the most difficult relations which requires a wide context is predicate coordinated structures. The structures may not be able to parse by using only simple contextual features.

Comparison between Models

Our qualitative analysis between parsing models resulted no distinct difference. The tournament model correctly parsed 95 more bunsetsus than the CC model for the test data of January 9th in the experiment performed in Section 4.4.2. However, it is not true that the tournament model is a better choice than the CC model for most of sentences. The number of bunsetsus that both models are correct is 9,588 and that both models are incorrect is 875. The number of bunsetsus that only the tournament model is correct is 309 and that only the CC model is correct is 214. A slight change in a feature template design or a parsing algorithm may flip a certain amount of error bunsetsus to be correct and vise versa.

4.6 Related Work: Tournament Models for Multilingual Dependency Parsing

The tournament model can be extended for parsing other languages. As described above, the tournament model is designed for parsing Japanese, which is a strictly head-final language. However, most languages do not have HEADFINAL constraint. A different parsing algorithm should be employed for other less-constrained languages so as to relax this constraint. A simple solution to adapt the tournament model to such languages is not to restrict the candidate heads to ones located on the right of a dependent. It does not matter for the tournament model that whether a language is projective or non-projective. The tournament model can be extended to relax PROJECTIVE constraint.

We conducted experiments with the dependency structure version of Penn Treebank, an English corpus, for the CoNLL-2008 Shared Task [66]. Our modification to

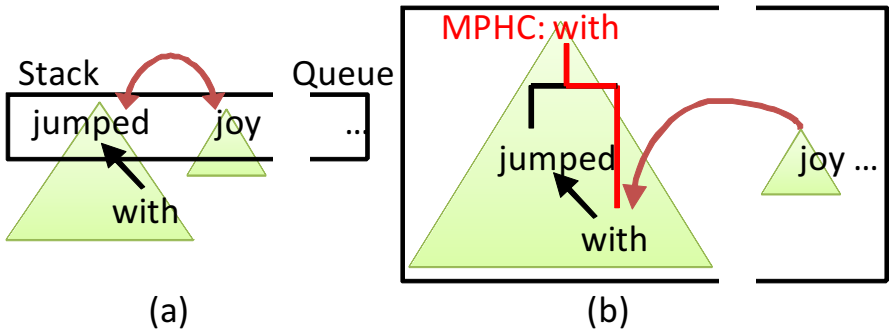


Figure 4.8: (a) Nivre's (word-based) transition-based model and (b) Kitagawa's tree-based model.

the tournament model is not only replacing bunsetsu-based features with word-based one. We discriminate that a candidate head is located on whether left or right of a dependent, as feature names. Our parser achieved labeled (syntactic) dependency accuracy of 88.06% on WSJ data, the in-domain test part of the corpus. We ranked third out of 16 participants of the closed challenge of the shard task in dependency parsing accuracy. The unlabeled dependency accuracy of 90.73% for WSJ data shows that the model is also effective in other languages, which may not strictly head-final or projective. However, since output of the English parser usually violates SINGLEROOTED constraint, this simple solution is not practical.

Kitagawa and Tanaka-Ishii [19] proposed a transition-based dependency parsing model in which naturally integrates tournaments into multilingual dependency parsing. They introduced a tournament into the *Left-Arc* and the *Right-Arc* actions of Nivre's transition-based model [41].

There is only one action sequence for the shift-reduce-like transition-based parsers to build a certain dependency tree. That is, the parsers restrict their parsing order strictly. For example, Nivre's arc-eager parser requests strictly top-down parsing for left-half, and strictly bottom-up parsing for right-half of the subtree rooted by each word. Therefore, even if a dependency relation is included in the gold-standard tree, selecting the action which adds the relation may be a mistaken decision. For example, in Figure 4.8(a), the word "joy" cannot modify "with" because of the preceding mistaken decision that "with" modifies "jumped". There are no actions which assign an arc between a word and another word inside a subtree. The relation should be set after constructing its children.

In contrast, Kitagawa's model may prevent such error propagations. As shown in Figure 4.8(b), the *Right-Arc* action of the model holds a tournament.[14] If "with" was selected as the most probable head candidate (MPHC), the *Right-Arc* action will add an arc between "with" and "joy". Action selection in the model has two steps. First, the model selects two MPHCs for the *Right-Arc* and the *Left-Arc* actions, respectively. Then, it selects the action to perform among *Right-Arc*, *Left-Arc* and *Shift*.[15] They also applied this method to Yamada's [67] and Goldberg's [9] transition-based parsers [20].

4.7 Summary

We proposed the tournament model, which is a classification-based model considering relative preference of two candidate heads in a direct comparison. The model is a data-driven extension of Kanayama's [16] direct comparison model; our model does not require any grammar. The tournament model is a twin-candidate model in the sense that the binary classification in it considers two candidate heads and then selects the better candidate. Its binary classification problem is easily solved in more contexts than that of previous single-candidate models.

The relative distance model proposed by Yamamoto and Masuyama [70] can be regarded as a score-based tournament model, which employs a score comparison instead of a direct comparison. Therefore, their model learns a total ordering of preferences. In contrast, our model learns a partial ordering. With an approximate combinational feature computation method, their model is more compact than our model in the sense that the number of features tend to be smaller and its parsing speed tends to be faster. Since our model directly considers two candidates, our model can be more accurate than their model.

We conducted experiments with Kyoto Text Corpus. The tournament model outperforms the previous models, and the dynamic features which we were developed improve parsing accuracy. Since time complexity of the model is at least $O(n^2)$, its parsing speed is relatively slow. Variations of its parsing algorithm do not have a significant impact on its parsing accuracy, such as the option of left-to-right parsing or right-to-left parsing.

[14] The *Right-Arc* action adds a left-arc in this figure because the action is named for the regular English dependency tree notation. The direction of an arc in Japanese and English dependency trees is inverse.

[15] Their model removed the *Reduce* action, which pops a word off the stack, because the action is no longer necessary.

Parsing error analysis suggests several tendency of the model. It is difficult to parse long dependency relations, and apposition and parallel dependency relations. The analysis focusing on dependents' POSs implies that the tournament model requires more training examples than single-candidate models to learn parsing rules for a dependent with a particular POS. We presented several typical patterns of parsing errors which seem to require a certain sort of rich information to parse correctly.

The tournament is also applicable to multilingual dependency parsing. Since most of languages do not satisfy HEADFINAL constraint, our straightforward application of the tournament model to English dependency parsing resulted in a parser whose output is usually an ill-formed dependency graph. Kitagawa and Tanaka-Ishii [19] incorporated a tournament into transition-based dependency parsing models naturally.

Chapter 5

Active Learning for Japanese Dependency Parsers

An SVM classifier returns a signed distance from its hyperplane corresponding to an input feature vector. In this chapter, we exploit the distance as a confidence measure. In other words, a feature vector close to the hyperplane is likely to correspond to a misclassification. On the other hand, a feature vector which is far from the hyperplane is likely to be correct. The distances enable a dependency parser with an SVM to assign confidence scores to dependency relations.

The scores which we define in Section 5.1 enable to extract only accurate dependency relations as we show in Sections 5.2 and 5.3. NLP applications such as an information extraction system usually uses not a full dependency tree but its subtree or its subset of dependency relations. The scores enable to construct a high-precision system by extracting the dependency relations which is likely to be correctly parsed.

The scores are also applicable to boosting efficiency of corpus annotation. Since a low score suggests that its corresponding dependency relation tend to be incorrect, the scores play the role of an error detector. Annotators efficiently improve the quality of a corpus by correcting each of bunsetsus corresponding to parsing errors and annotation errors which the error detector pointed out. While the error detector can reduce annotation cost significantly, as shown in Section 5.4, it is not a realistic scenario because the typical annotation setting is not a such bunsetsu-wise annotation but a sentence-wise annotation. Therefore, we attempt to develop an efficient annotation method by selecting sentences to be annotated by using active learning in Section 5.5. We explore an effective active sampling method based on the scores. The best sampling method chooses the most valuable sentences for training a parser, and the method results in the

most accurate parser with the lowest manual annotation cost.

5.1 Definition of Score for Active Learning

This section defines the score of a bunsetsu for each of three parsing models which uses SVMs: the SR, the CC and the tournament models. We regard the score of a dependency relation as the score of its dependent bunsetsu. Defining scores for these models is not trivial because an SVM is just a binary classifier, unlike probabilistic parsing models in which a probability itself is a confidence score. Note that we define the scores of rightmost two bunsetsus of a sentence as infinity because their heads are not ambiguous.

SR Model

The score definition for the SR model is basically the distance when the *Reduce* action is selected. The SR model, introduced in Section 3.2, parses a sentence as a sequence of action selections; the possible actions are *Shift* and *Reduce*. The model asks to its SVM whether a focused-on dependent b and a focused-on candidate head c is in a dependency relation or not. A score is assigned to the focused-on dependent when the SVM answered "Yes.", i.e., the *Reduce* action is selected. The score s_b is the distance from the hyperplane $d_{b,c}$. Since "Yes" is the positive class $(+1)$ in our implementation of the model, $d_{b,c} > 0$. However, the only exception is when the focused-on candidate head is the rightmost bunsetsu of a sentence. In such a configuration, the model performs the *Reduce* action without performing an SVM classification because there is no more bunsetsus to be *Shift*ed. To assign a score to such a dependency relation, we modified the model to perform an SVM classification even in such a case. Even the SVM answers "No", we adopt the distance as the score of the focused-on dependent. Therefore, there may be bunsetsus such that $d_{b,c} < 0$.

CC Model

The score for the CC model is defined as same as that for the SR model. The only difference between the SR and the CC models is in their parsing order. The CC model also does not perform an SVM classification and establishes a dependency relation for

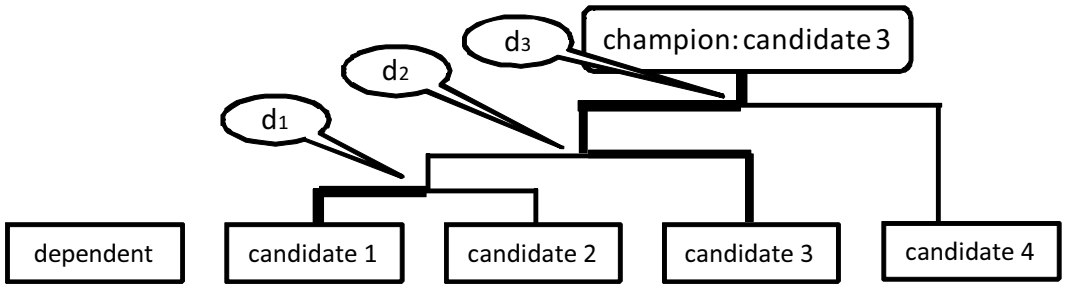


Figure 5.1: A tournament and the distances corresponding to each of games.

the pair of the rightmost two bunsetsus in the workspace of the CC model, in each iteration.

Tournament Model

We define scores for the tournament model in a different manner from the SR and the CC models. Unlike the SR and the CC models, the tournament model can play two or more games for selecting the head of a focused-on dependent. We define the score of a focused-on dependent as the minimum unsigned distance from the hyperplane among games. The minimum distance corresponds to the match of the two strongest candidates. A score of the focused-on dependent is defined as $s_b = \min_{d \in D} |d|$ where D is the set of signed distances from the hyperplane corresponding to each of games that the champion of the tournament participated. The sign of a distance is ignored because the two classes of the tournament model are not *Shift* and *Reduce* but *Left* and *Right*.

An example of a tournament is shown in Figure 5.1. The focused-on dependent has four candidate heads and there are three games. Suppose the distances from the hyperplane of each game as d_1, d_2 and d_3. The champion is the candidate 3 and the distances of the games that the champion participated are d_2 and d_3. Therefore, $D = \{d_2, d_3\}$ and then $s_b = \min(|d_2|, |d_3|)$.

5.2 Extracting Accurate Dependency Subtrees

In this section, we attempt to extract a subset of parser's output whose accuracy is much higher than that of the full output. We investigate the correlation between

its accuracy and coverage. Since high-scored relations are likely to be correct, this method may improve the performance of certain NLP applications such as information extraction by using only high-scored relations.

Experimental settings are same as Section 4.4.1. Dependency accuracy is calculated excluding the rightmost bunsetsu of each sentence. We use all features: the standard, additional and case particle features. The training data is January 1st to 8th (7,587 sentences) and the test data, i.e., the target data of this analysis, is January 9th (1,213 sentences).

Figure 5.2 shows partial accuracies when we extract only the bunsetsus whose score, defined in Section 5.1, is greater than or equal to a certain score threshold. Suppose we employed the tournament model as the parser and we extract only the bunsetsus whose scores are greater than or equal to 1.0. The accuracy calculated among the extracted bunsetsus is about 99%. In other words, 99% of the extracted bunsetsus are ones that their heads were identified correctly. A score of the SR and the CC models may be negative and that of the tournament model is always positive. Therefore, the curve of the tournament model is flat in the range that the score threshold is negative. Note that the curves of the CC and the SR models are almost the same in the figures in this section. The reason is probably that the only difference between these models is in their parsing order. Although their available features are different, we use few dynamic features. Therefore, the outputs of both the models are almost the same.

Figure 5.3 shows the coverage of bunsetsus when a certain score threshold is given. For example, if we extract only the bunsetsus whose scores are greater than or equal to 1.0, the coverage is about 65 %. Coverage is defined as the number of extracted bunsetsus divided by the total number of bunsetsus. In this figure, the coverages with these models are almost the same. Note that the second rightmost bunsetsu of each sentence are always extracted regardless of score threshold because their scores are infinity. Therefore, the lower bound of the coverage is not 0%.

From Figures 5.2 and 5.3, we get the coverage-accuracy curve shown in Figure 5.4. With the tournament model, if we extract 60% of the highest-scored bunsetsus, the accuracy among them is about 99%. On the other hand, those of the CC and the SR models are about 97%. To extract bunsetsus with the accuracy of 99% by the CC or the SR model, we should limit the coverage to 20%. The tournament model is also more accurate for partial parsing; in overall accuracy, the tournament model is more accurate than the CC and the SR models by 0.9%. However, the difference become greater in partial parsing accuracy. A reason is that while the CC and the SR models

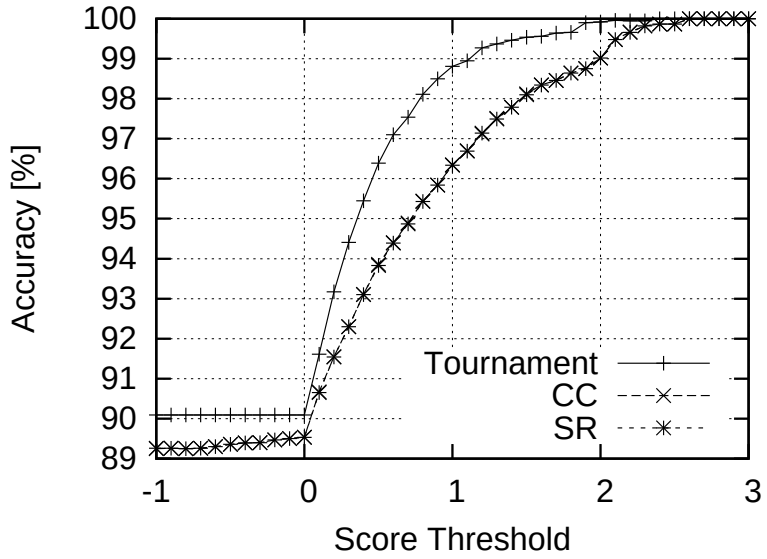


Figure 5.2: Score-accuracy curve.

calculate a score based on absolute preferences between a dependent and a candidate head, a score of the tournament model is based on relative preferences.

5.3 Quantitative Distinction between High-scored and Low-scored Group

We also investigate the correlation among the scores, lengths of dependency relations and their accuracies. The extracted bunsetsus with our method are probably biased. Generally, short-distance dependency relations tend to be more accurate than long-distance ones, as we investigated in Table 4.7. A concern arises that the reason of our high partial accuracy can be that our method extracts only the dependency relations whose distances between their dependents and heads are short.

Table 5.1 shows the distribution of bunsetsus and their accuracy divided by their scores with the tournament model and distance ranges. While the bunsetsus whose distances are 1 are mostly distributed in the score range 1.0-1.5, the bunsetsus whose distance is longer than 1 are mostly distributed in the score range 0.0-0.5. The score ranges 2.0-2.5, 2.5-3.0 and $\geq$ 3.0 are dominated by the bunsetsus whose distances are

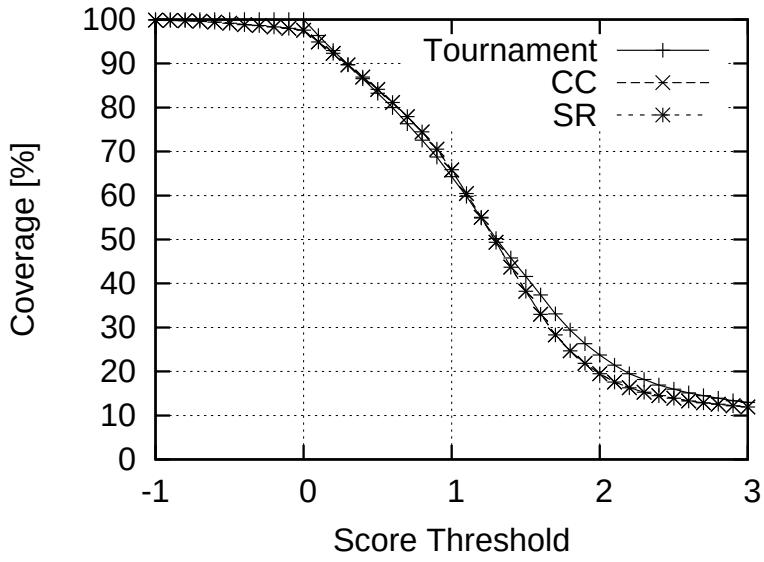


Figure 5.3: Score-coverage curve.

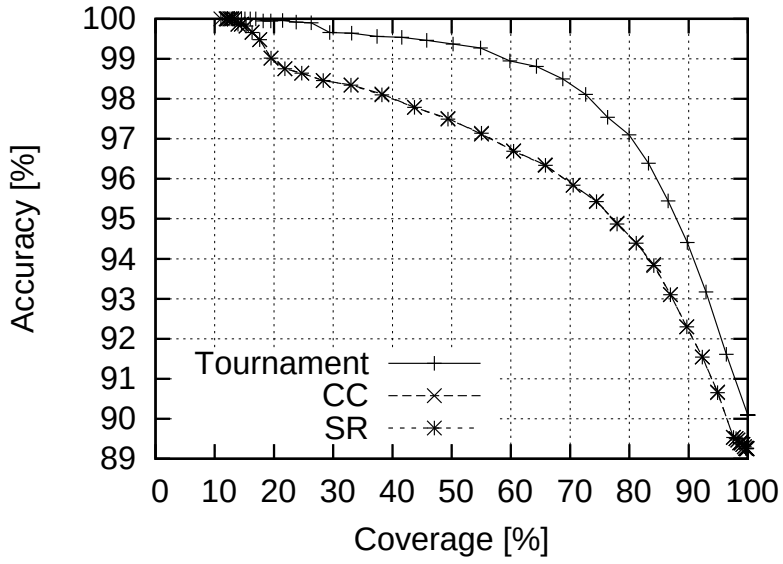


Figure 5.4: Coverage-accuracy curve.

Score/Dist	1	2-3	4-5	6-7	8-9	10-14	15-19	20+
0.0-0.5	464	715	275	175	93	79	34	9
	69.61	56.92	57.82	52.00	50.54	50.63	38.24	55.56
0.5-1.0	888	673	234	115	77	59	24	7
	96.17	83.95	79.91	80.00	80.52	72.88	87.50	**100.00**
1.0-1.5	1601	533	161	72	57	49	18	6
	99.44	95.12	91.93	86.11	94.74	95.92	**100.00**	**100.00**
1.5-2.0	1572	193	86	38	28	24	11	7
	100.00	95.34	94.19	94.74	**100.00**	91.67	**100.00**	85.71
2.0-2.5	769	24	15	9	10	12	8	5
	100.00	91.67	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
2.5-3.0	318	5	1	3	2	2	1	-
	100.00	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	-
≥ 3.0	210	-	-	-	1	-	1	-
	100.00	-	-	-	**100.00**	-	**100.00**	-

Table 5.1: Frequency distribution and accuracies of bunsetsus classified by their scores and distances. The parsing model is the tournament model.

Score/Dist	1	2-3	4-5	6-7	8-9	10-14	15-19	20+
< 0	7	46	34	39	49	49	32	13
	0.00	54.35	50.00	76.92	91.84	91.84	96.88	**100.00**
0.0-0.5	414	483	184	161	101	78	41	18
	85.02	52.17	51.63	55.28	61.39	51.28	60.98	61.11
0.5-1.0	960	572	231	103	66	58	19	3
	94.79	77.97	74.46	69.90	80.30	68.97	68.42	**100.00**
1.0-1.5	2157	521	202	69	37	37	2	-
	96.71	88.10	87.13	79.71	86.49	81.08	**100.00**	-
1.5-2.0	1575	338	93	34	13	3	3	-
	98.54	92.90	92.47	94.12	92.31	66.67	**100.00**	-
2.0-2.5	451	124	28	6	1	-	-	-
	97.34	94.35	**100.00**	**100.00**	**100.00**	-	-	-
2.5-3.0	168	50	-	-	1	-	-	-
	99.40	98.00	-	-	**100.00**	-	-	-
≥ 3.0	90	9	-	-	-	-	-	-
	100.00	**100.00**	-	-	-	-	-	-

Table 5.2: Frequency distribution and accuracies of bunsetsus classified by their scores and distances. The parsing model is the CC model.

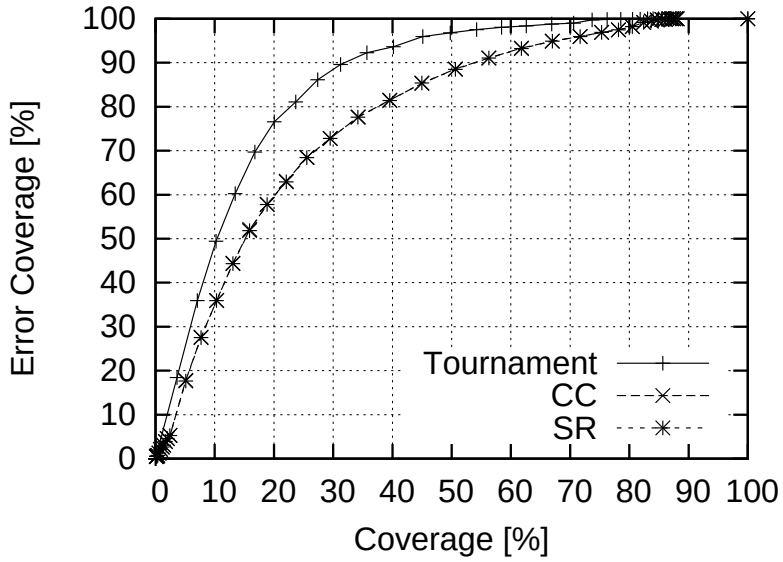


Figure 5.5: Coverage-error coverage curve.

1. In contrast, the bunsetsus whose scores $\geq$ 1.0, which cover about 60% with the accuracy of 99%, includes a certain number of long-distance bunsetsus. The table shows that the bias in their distances is not so serious, that is, the concern proved to be false.

Some of the cells in the table correspond to certain patterns. In the score ranges 2.5-3.0 and $\geq$ 3.0, the distances of most of the bunsetsus are 1. Their head bunsetsus usually include close brackets (」). Most of the bunsetsus whose score is lower than 0.5 seems to be difficult to parse correctly. They usually correspond to coordination or apposition relations.

Table 5.2 is the distribution table of the CC model. The tournament model has higher extraction performance than the CC model. That is, the accuracy among low-scored bunsetsus is lower and that among high-scored bunsetsus is higher.

When using the tournament model, the accuracy of the bunsetsus whose distances are 1 is 100% if we extract bunsetsus whose scores are greater than or equals to 1.5. There are 2869 bunsetsus which satisfy this condition.

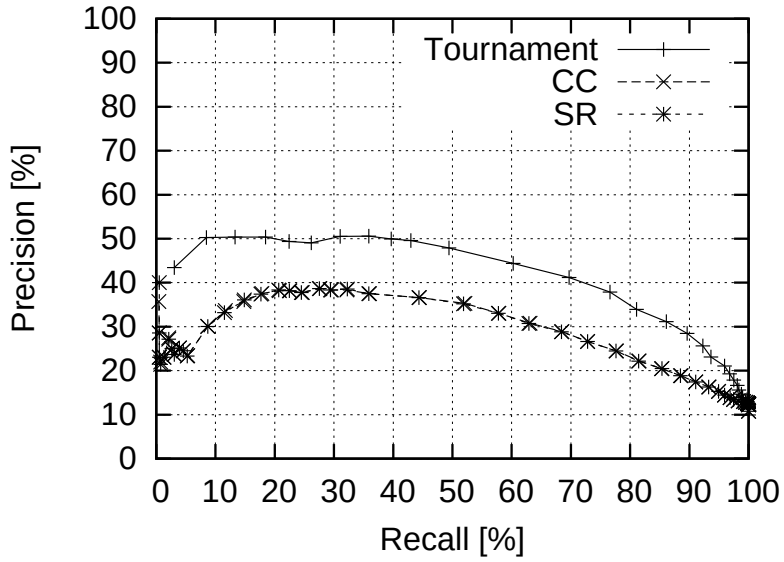


Figure 5.6: Error detection precision-recall curve.

5.4 Detecting Parsing Errors

This score-based bunsetsu extraction can be applied to supporting corpus annotation because low-scored bunsetsus are likely to be incorrect. Since dependency structure annotation usually means manual correction of a parser's output, the scores enable a parser to tell annotators which bunsetsu is likely to be incorrect. In this case, the scores play the role of a parsing error detector.

Figure 5.5 shows how many error bunsetsus are covered when we select a certain number of lowest-scored bunsetsus. For example, if we extract 30% of the lowest-scored bunsetsus, about 90% of errors are covered.

Figure 5.6 is the precision-recall curve corresponding to our error detection performance. In other words, it shows the performance of our method as an error detector. The performance is not sufficient as a stand-alone error detector. However, it is sufficient to support the annotators. Note that the result is obtained with an in-domain test data. However, in our scenario of corpus annotation, an error detection will be performed for a new corpus, which is often an out-of-domain data. In such a case, the parsing accuracy and then the error detection performance are expected to decrease.

5.5 Application to Active Learning

Since low-scored bunsetsus tend to be incorrect, our score-based method enables an efficient corpus construction by annotating such bunsetsus first. "Efficient" means constructing a more accurate parser with a lower annotation cost. Suppose our error detector extracts 30% of lowest-scored bunsetsus from a corpus to be annotated and annotators correct only them. By the correction, as shown in Figure 5.5, the number of errors are expected to be reduced to one-tenth. However, an annotator typically annotates a *sentence* as a unit of annotation. Therefore, it is not a realistic scenario. Because annotating a bunsetsu requires to see its surrounding context, the per-bunsetsu annotation cost in our scenario is higher than the regular sentence-wise annotation scheme. Therefore, we attempt to develop an efficient annotation method by selecting *sentences* which include lowest-scored bunsetsus. In the rest of this section, "the number of bunsetsus" is counted excluding the rightmost *two* bunsetsus of each sentence. Additionally, we never use the sentences whose lengths are less than 3 because their dependency structures are not ambiguous and then annotators have nothing to do.

Definition of Sentence Scores

We define the sentence score as the lowest bunsetsu score in the sentence. Let B be the set of all the bunsetsus in a sentence, its sentence score $s = \min_{b \in B} s_b$. We regard lowest-scored sentences as the mostly informative ones for the SVM which assigned the bunsetsu scores and then sentence scores.

The Algorithm

We use the pool-based uncertainty sampling algorithm [32]. The set of candidate sentences to be annotated is referred to as the *pool*. The algorithm is shown in Algorithm 3. Note that in our experiments, the step 3 just adds the selected annotated sentences from Kyoto Text Corpus, instead of annotating them by human.

We employs several sentence selection strategies in the step 1. The *Score* strategy selects the lowest-scored m sentences. There are several simple selection strategies: selecting sentences randomly (*Random* strategy), selecting from the first sentence of training sentence file (*Sequential* strategy), and selecting the longest m sentences, i.e., the number of bunsetsus is largest (*Long* strategy). The *Long* strategy may be a

Algorithm 3 Pool-based active sampling algorithm.

1: Select m sentences from the pool.
2: Annotate the m sentences by human.
3: Train an SVM with all the annotated sentences.
4: Assign scores to all sentences in the pool by the trained SVM.
5: Evaluate parsing accuracy of the SVM with the test data.
6: Return to step 1 and repeat.

good strategy because long sentences tend to be complex and often contain coordinated structures, which are difficult to parse correctly. Another reason is that if we evaluate these strategies with a same number of sentences, the *Long* strategy has an advantage over the other strategies because the *Long* strategy maximizes the number of bunsetsus to be selected. Random sampling is referred to as *passive sampling* and a strategy that selects samples based on certain scores is referred to as *active sampling*.

At the beginning of our experiments, the pool contains all the sentences of January 1st (1069 sentences[1]) and we set $m = 50$. For a set of sentences whose lengths are the same each other, the *Long* strategy selects sentences randomly among these sentences. We cannot apply the *Score* strategy at the first sampling because there are no annotated sentences and therefore no scores are attached to the sentences in the pool. In this case, we apply the *Long* strategy, instead.

Comparison between Models

Figure 5.7 shows the accuracy growing curve of the three parsing models with the *Long* strategy. Note that the *Score* strategy is the only strategy such that selected sentences depend on the parsing model. The parsing accuracy reaches 85% with 50-100 training sentences. The CC and the SR models are more accurate when the number of training sentences is less than 150. On the other hand, the tournament model is more accurate with more than 150 training sentences. Because the tournament model is a twin-candidate model, the model seems to require more training sentences than the others to learn basic parsing rules. Hereafter, we use the tournament model for comparison among strategies.

[1]The number is calculated excluding the sentences whose lengths are less than 3.

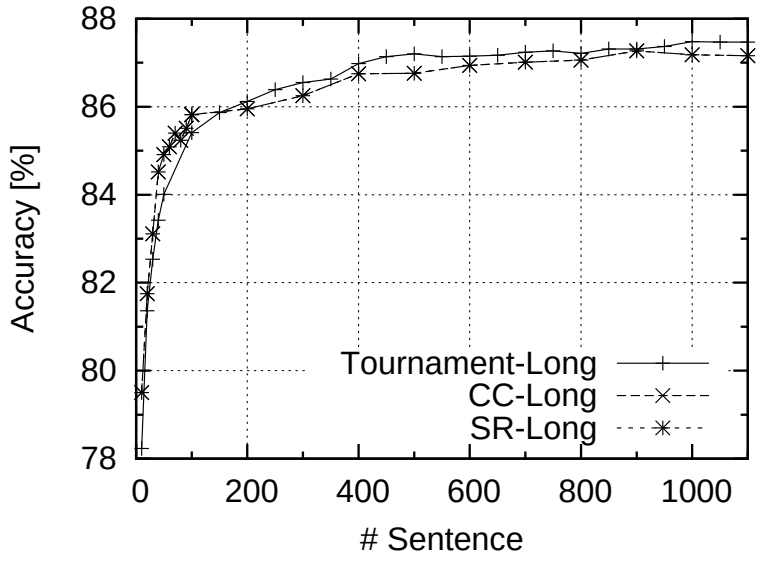


Figure 5.7: Accuracy comparison among the models with *Long* strategy.

Comparison between Strategies

Figure 5.8 shows the comparison in accuracy growing among the strategies: *Score*, *Long*, *Random* and *Sequential*. Since this comparison is in the number of sentences, the *Long* strategy has an advantage. However, the comparison is not fair because annotation cost increases according to the length of a sentence.

If we assume an annotation cost is proportional to the number of bunsetsus, the figure should be replaced with Figure 5.9, which is the comparison in sentence lengths. The differences between those strategies are reduced. While the *Long* strategy is defeated by the *Sequential* strategy when the number of bunsetsus is more than 4,000, the *Score* strategy keeps the top performance.

Figure 5.10 shows the tendency of the number of bunsetsus selected by each strategy. The gradients of the *Random* and the *Sequential* strategies are almost constant. The *Long* strategy selects long sentences. The strategy selects short sentences when the number of sentences in the pool decreased. However, since active sampling methods suppose to select a small amount of sentences from a large number of sentences, the second half of the graph is not so important. The curve of the *Score* strategy is in the middle of the other strategies and selects relatively long sentences. Since long sentences usually contain parsing errors, their scores tend to be low, and therefore

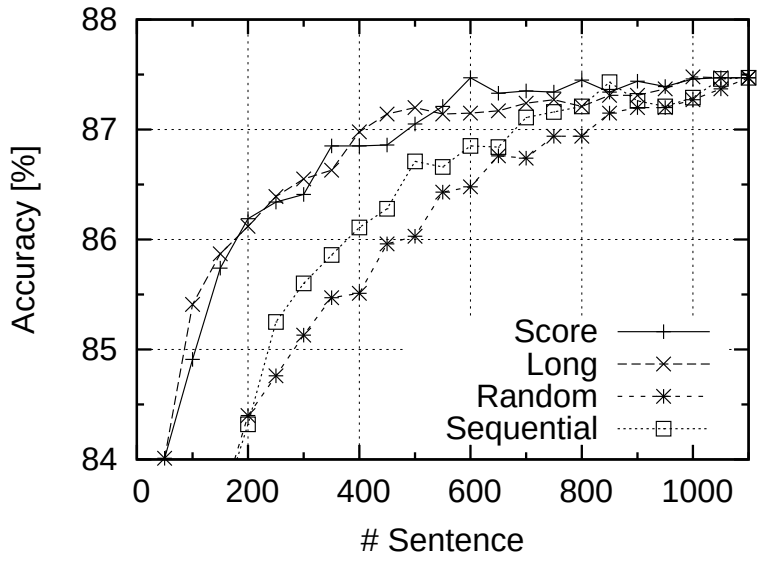


Figure 5.8: Accuracy comparison among the strategies in the number of sentences.

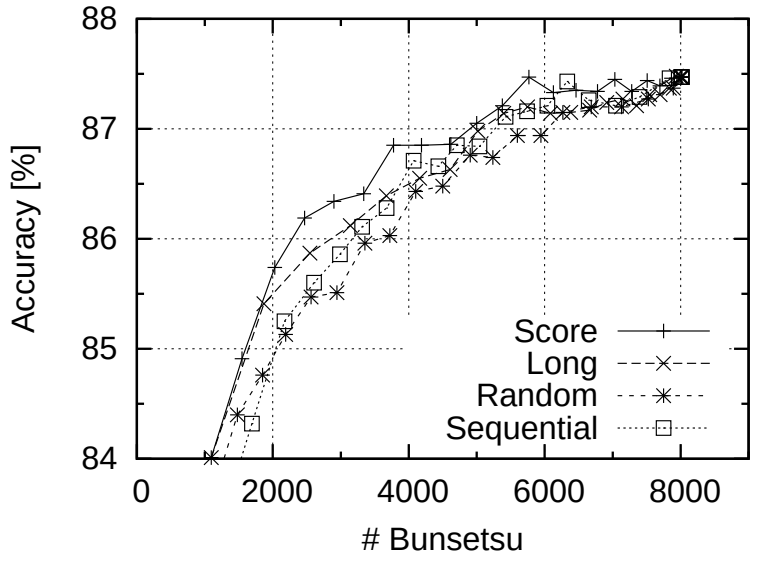


Figure 5.9: Accuracy comparison among the strategies in the number of bunsetsus.

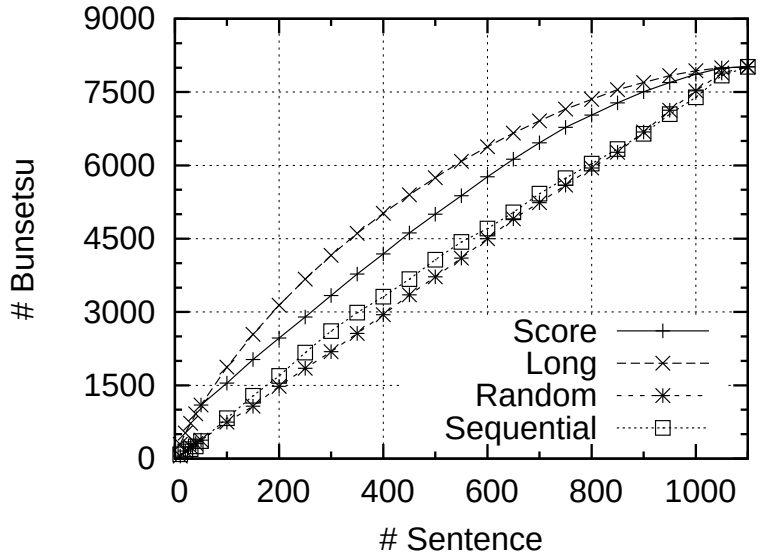


Figure 5.10: The number of selected sentences vs. the number of bunsetsus contained in them.

selected.

5.6 Related Work in Active Learning

Dagan and Engelson [6] introduced an active learning method for a natural language processing task other than text classification. They proposed a majority voting method with multiple classifiers to sample the examples for HMM-based part-of-speech tagging.

There are several works which adopted active learning methods for parsing. Thompson et al. [62] introduces a majority voting-based active sampling method for deterministic shift-reduce semantic parsing. Hwa [12, 13] investigated active sampling methods based on sentence lengths and entropy-based metrics for phrase-based parsing. Tang et al. [61] introduced an active sampling method based on the K-means clustering and an entropy-based metric for shallow semantic parsing. For dependency parsing, Sagae and Tsujii [52] introduced an active sampling method with a combination of SVM and log-linear models for parser domain adaptation. For Japanese natural language processing, Sassano [54] introduced the score-based method with SVMs for Japanese

word segmentation. Our experiments are similar to Sassano's in the sense that we employ SVM scores. We defined the confidence scores of sentences for SVM-based Japanese dependency parsers and applied the scores to active learning.

5.7 Summary

We defined the confidence score of a bunsetsu in an output of each of the SR, the CC and the tournament models. It is based on a signed distance from an SVM's hyperplane corresponding to an input feature vector. The tournament model achieves the highest partial parsing accuracy compared to the other models. With the tournament model, if we extract 60% of the highest-scored bunsetsus, the accuracy among them is about 99%. The high partial accuracy is beneficial in constructing a high-precision system which uses a dependency subtree of a sentence.

The score is also applicable to efficient corpus annotation. With the tournament model, if we extract 30% of the lowest-scored bunsetsus, about 90% of parsing errors are covered. The number of errors are expected to be reduced to one-tenth by correcting the 30% of bunsetsus. However, it is not a realistic scenario because the typical annotation setting is not a such bunsetsu-wise annotation but a sentence-wise annotation. Therefore, we attempt to develop an efficient annotation method which selects sentences to be annotated by using active learning. The best selection strategy chooses the most valuable sentences for training a parser, and the method results in the most accurate parser with the lowest manual annotation cost. In our experiment, the score-based sentence selection strategy achieves the top performance.

Chapter 6

Parser Stacking on the KNP Parser

Stacking is a technique of parser combination in which the output of a parser is incorporated into another parser as guide features. Martins et al. [35], and Nivre and McDonald [45] conducted stacking experiments in multilingual dependency parsing and reported that stacking significantly improves parsing accuracy. They employed transition-based parsers (MaltParser[1]) and graph-based parsers (MSTParser[2]) [45, 35]. Parser combination, including stacking, tends to be more beneficial if properties of the two parsers differ. For example, the features for MSTParser are very restricted to local features. However, those for MaltParser can be designed flexibly, and dynamic features are available for the parser. MSTParser performs exact inference, but MaltParser performs greedy inference.

Improvement by stacking MaltParser and MSTParser for word-based Japanese corpora is smaller than for other languages. Nivre and McDonald [45] reported the results for Japanese: 91.65% (Malt) $\rightarrow$ 92.20% ($Malt_{MST}$[3]). Martins et al. [35] also reported: 91.65% (Malt) $\rightarrow$ 91.63% ($MST2O_{Malt}$[4]).[5] One of the reasons for these low

[1] `http://maltparser.org/`

[2] `http://sourceforge.net/projects/mstparser/`

[3] It is the stacked parser in which MaltParser is stacked on MSTParser. In other words, the output of MSTParser is passed to MaltParser.

[4] It is the McDonald's second-order MST parser stacked on MaltParser. As we showed in Figure 3.8, the MST parser does not consider grandchild interactions if a grandchild is located outside of its parent and grandparent. Therefore, due to Japanese HEADFINAL constraint, the MST parser does not capture grandchild interactions in Japanese sentences.

[5] The accuracies on word-based corpus may seem to be higher or comparable with that of bunsetsu-based corpus. However, as Sassano and Kurohashi [56] showed in their experiments of word-based Japanese dependency parser, bunsetsu-based accuracy of 88.5% is 95.1% in word-based accuracy, for example. Word-based parsing may be less accurate than bunsetsu-based parsing because of available

improvement is probably that the MST parsers are not suitable for Japanese. Therefore, we conduct experiments for stacking the two Japanese parsers on bunsetsu-based Japanese corpora: the tournament model and KNP 3.01[6] [17].

Properties of the two parsers differ in several aspects. The tournament model mainly based on learning syntactic rules by using machine learners such as SVMs. Rich feature combination is available for the parser because of polynomial kernels in SVMs. The parser also considers their relative positions of candidate heads. In contrast, KNP 3.01 is a rule-based parser with reranking by a generative model, that is, the reranker is a fully probabilistic model. Advantages of the parser are mainly based on word co-occurence information constructed from a very large unannotated corpus. The parser evaluates plausibilities of all the predicate-argument structures included in candidate dependency trees by using the information, and outputs the dependency tree with highest probability.

Stacking with KNP is expected to improve parsing accuracy because not only the two parsers diverse but also it is one of the easiest way to incorporate co-occurrence information naturally into discriminative parsing models such as the tournament model. We first evaluate the stacked model with Kyoto Text Corpus. Then, we show stacking also improve accuracy even with a target-domain test data.

6.1 Overview of Experiments with Kyoto Text Corpus

Overview of the stacked parser is shown in Figure 6.1. KNP (level 0 parser) guides the tournament model (level 1 parser). The tournament model refers to the output of KNP and generates the following guide features:

- $GUIDE_{LEFT}$: whether the left candidate head is the head of the focused-on dependent in the output of KNP
- $GUIDE_{RIGHT}$: whether the right candidate head is the head of the focused-on dependent in the output of KNP
- $GUIDE_{NONE}$: whether both $GUIDE_{LEFT}$ and $GUIDE_{RIGHT}$ are 0

features.

[6] We did not specify the compile option to refer to a Japanese thesaurus, `--with-bgh`.

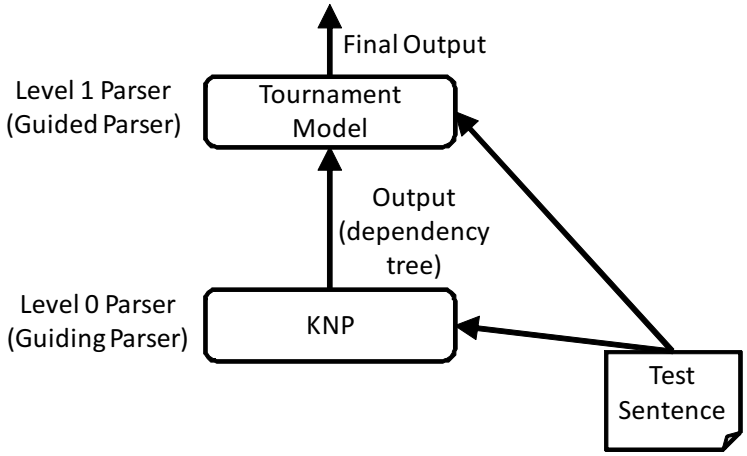


Figure 6.1: Overview of the stacked parser: the tournament model is stacked on KNP.

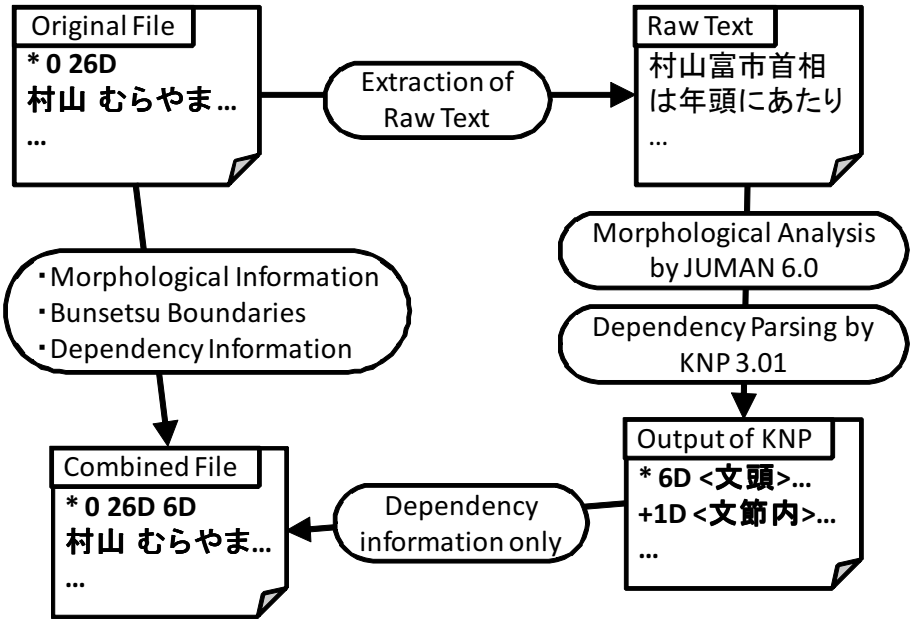


Figure 6.2: The procedure to combine the gold-standard and output of KNP for stacking experiments.

We merged KNP's output to Kyoto Text Corpus Version 3.0 by the procedure shown in Figure 6.2. Since the input format which KNP expects is that of JUMAN morphological analyzer, we first converted the corpus to a raw text. Then, the text was passed to JUMAN 6.0, and then the output was passed to KNP 3.01. The format of KNP's output and the original Kyoto Text Corpus format are different. Also, their bunsetsu boundaries are sometimes different. Therefore, we integrated KNP's output into Kyoto Text Corpus. Bunsetsu boundaries in KNP's output was modified to obey that of Kyoto Text Corpus. Every bunsetsu in the combined format has two heads: the gold-standard head and the head imported from KNP's output. The latter is incorporated into training and test of the tournament model as features.

There are several reasons that the two parsers are not fair in this experiment. The tournament model uses gold-standard morphological information, which annotators of Kyoto Text Corpus had checked. In contrast, KNP uses JUMAN's output, which contains analysis errors. There are another reasons which may imply advantages of KNP. The probabilities used in KNP had been tuned with Kyoto Text Corpus, except for the probabilities which correspond to co-occurrence information. Additionally, KNP and Kyoto Text Corpus had been developed simultaneously. That is, the corpus was constructed by the iterations of manual correction of KNP's output, and grammar engineering of KNP based on findings by the correction [29].

6.2 Experiments with Kyoto Text Corpus

Results are shown in Table 6.1. The upper half of the table corresponds to the accuracies on the same corpus as Table 4.5 and the lower half corresponds to that as Table 4.2. The test data of the lower half is January 10th. The binary classifier in the tournament model is an SVM with the third degree polynomial kernel. The upper half shows that the tournament model without stacking significantly outperforms KNP (+1.4% in dependency accuracy). In comparison with KNP, stacking improves parsing accuracy, even with small number of training sentences. The accuracy with all the training sentences is 1.0% higher than that of the tournament model without stacking, and 2.4% higher than that of KNP.

In experiments shown in the lower half of the table, the stacked model also outperforms the other models by 1.3%. KNP is comparable with tournament model without stacking. The larger the number of training sentences, the higher accuracy of the tournament model. In contrast, parsing accuracies of KNP does not depend on the number

Corpus	Parser	# train. sent.	Dep. Acc.	Sent. Acc.
Kyoto Text	KNP only	(0)	90.55	54.01
Corpus	Stacked	100	90.80	54.28
Version 3.0		250	91.11	54.70
		500	91.33	55.00
		1000	91.51	55.52
		2500	91.85	56.80
		5000	91.95	56.73
		10000	92.25	58.11
		24263	92.93	60.54
	Tournament only	(24263)	91.96	56.93
Kyoto Text	KNP only	(0)	90.06	50.51
Corpus	Stacked	7635	91.39	53.21
Version 4.0	Tournament only	(7635)	90.11	49.02

Table 6.1: Dependency and sentence accuracies of stacked parsers on Kyoto Text Corpus.

of training sentences because it does not employ any machine learners.

6.3 Domain Adaptation Experiments with KNB Corpus

We also conducted the experiments with Kyoto University-NTT Blog Corpus (KNB corpus). We converted the format of the corpus to the Kyoto Text Corpus format, then applied the same procedure as Figure 6.2.

First, we evaluated the accuracies on the corpus with parsers trained with Kyoto Text Corpus. We used all the 4185 sentences of KNB corpus as the test data. Results are shown in Table 6.2. The accuracy of KNP is low and stacking failed to improve accuracy. The main reason is probably that the test data is out-of-domain. That is, topics and styles are different between the training and the test corpora. In this experimental setting, Kyoto Text Corpus is a *source-domain* corpus and KNB corpus is a *target-domain* corpus. Compared to experiments whose test data is in-domain, accuracy of

Corpus	Parser	# train. sent.	Dep. Acc.	Sent. Acc.
Train KC3.0	KNP only	(0)	85.63	51.96
Test:KNBC	Stacked	24263	88.02	57.60
	Tournament only	(24263)	88.23	55.58

Table 6.2: Dependency and sentence accuracies of the stacked parser. KC3.0: Kyoto Text Corpus Version 3.0. KNBC: Kyoto University-NTT Blog Corpus.

the tournament model dropped by 3.7% (91.96 $\rightarrow$ 88.23) and that of KNP dropped by 4.9% (90.55 $\rightarrow$ 85.63).

Stacking may be applied to domain adaptation by replacing the level 1 parser with a parser trained with a target corpus. In general, if the amounts of a source-domain and a target-domain data are same, a parser trained with the target-domain corpus tends to significantly outperform the parser trained with the source-domain corpus. Therefore, to achieve accurate parsing on an out-of-domain corpus, the level 1 parser should be trained with a target corpus to learn the annotations on the corpus, even if an available target-domain training corpus is small. The level 0 parser provides information based on general parsing rules to the level 1 parser only as a guide. Then, the level 1 parser makes decisions based on domain-specific rules and the general rules passed as guide features.

We performed experiments with several parser combinations of level 0 and 1 parsers. We split KNB corpus into 2500 training sentences and 1685 test sentences. Results are shown in Table 6.3. The tournament model trained with KNB corpus without stacking significantly outperforms KNP, even with the small amount of training data. All the stacked models outperform $Tournament_{KNBC}$ without stacking. The accuracy of the stacked model in which $Tournament_{KNBC}$ is stacked on $Tournament_{KC}$ shows that stacking with two parsers trained with different corpora improves accuracy even if their parsing models are same. The second highest accuracy is achieved by stacking with the two level 0 parsers, KNP and $Tournament_{KC}$, and the level 1 parser, $Tournament_{KNBC}$. The highest accuracy is achieved by stacking $Tournament_{KC,KNBC}$, a tournament model trained with both Kyoto Text Corpus and KNB corpus, on KNP.

Level 0 parser(s)	Level 1 parser	Dep. Acc.	Sent. Acc.	# train. sent.
(no stacking)	KNP	85.95	54.81	2500
	$Tournament_{KC}$	88.63	58.77	
	$Tournament_{KNBC}$	88.02	57.27	
KNP		89.50	60.84	
$Tournament_{KC}$		89.20	60.52	
KNP and $Tournament_{KC}$		89.86	61.95	
	$Tournament_{KNBC}$	88.34	59.03	100
		88.73	59.87	250
		88.89	59.68	500
		89.36	61.10	1000
KNP	$Tournament_{KC,KNBC}$	90.00	63.51	2500

Table 6.3: Dependency and sentence accuracies with KNBC. $Tournament_{KC}$ and $Tournament_{KNBC}$ are the tournament models trained with Kyoto Text Corpus Version 3.0 and KNB corpus, respectively. $Tournament_{KC,KNBC}$ is the tournament model trained with the both corpora.

6.4 Discussion

The tournament model guided by KNP significantly outperforms individual parsers in all experiments except for Table 6.2. Stacking with certain sort of parser such as KNP is one of the easiest way to incorporate co-occurrence information naturally into discriminative parsing models such as the tournament model.

KNP and the tournament model are complementary. The main advantage of KNP is generative estimation of plausibility of a subtree such as a predicate-argument structure, based on a wide-coverage case-frame dictionary constructed from a very large unannotated corpus. In contrast, the tournament model makes a decision with considering wider context than KNP. Since the model is a discriminative model, it can consider much more features than generative models such as KNP. The tournament model seems to be good at memorizing annotation guideline of its training corpus. In other words, tournament model is more dependent on particular corpus, i.e., the training data, than KNP. It is not only a disadvantage in the sense of generality but also an advantage in the sense of adaptation to a particular annotation guideline. For example, suppose there is a coordinated structure which consists of three conjuncts. It

is guideline-dependent that its leftmost conjunct modifies whether its rightmost conjunct or its middle conjunct. Unlike 2-tuple parsing models, the tournament model can discriminate the difference in an annotation guideline by considering 3 bunsetsus at once.

Similar situation to Section 6.3, domain adaptation, occurs in the early stage of corpus construction. That is, a large amount of source-domain corpus and a small amount of target-domain corpus are available. The small amount of target-domain corpus corresponds to a corpus which is constructing. We can apply the stacking framework by replacing $Tournament_{KNBC}$ to $Tournament_{Target}$. As shown in Table 6.3, the stacked parser outperforms all the parsers without stacking, even with 250 training sentences. Therefore, once 250 target-domain sentences are annotated, we can get a more accurate parser for the target corpus by stacking. The parser will output more accurate automatic parse, and then efforts of annotators to correct parsing errors will be reduced.

Note that the domain adaptation setting and the corpus construction situation is not exactly the same. Table 6.3 shows that merging a source-domain and a target-domain corpora achieves the best accuracy. The merging is appropriate because the annotation criteria for both of Kyoto Text Corpus and KNB corpus are the same. However, for example, if POS sets of the existing corpus and the constructing corpus is different, stacking is preferable.

6.5 Summary

We combined the two diverse dependency parsers by stacking: the tournament model and KNP. The dependency tree which KNP outputs is passed to the tournament model as guide features. In the in-domain experiment with Kyoto Text Corpus, the stacked parser significantly outperforms the individual parsers. We showed that the stacking method also improves parsing accuracy even in an out-of-domain setting.

In the stacking method, the interface between the two parsers is quite simple: a dependency tree. Any parser can play the role of a level 0 parser by using an appropriate conversion such as Figure 6.2. The conversion can absorb any difference between parsers such as in their POS set or their grammar formalisms. KNP 4.0 can be also incorporated into the stacking framework.

Each of parsers is highly independent. Once we implemented a level 1 parser which can accept any number of level 0 parsers, we can easily incorporate the advantage of a parser as a level 0 parser.

Chapter 7

Joint Inference of Dependency Parsing and Coordination Analysis Using a Dual Decomposition Algorithm

One of the difficult problems for dependency parsers is to identify spans of coordinated structures. As we showed in Section 4.5.2, many errors were observed in coordinated structures. A possible solution to the problem is to employ a coordination analyzer, which is specialized for capturing such structures; a sentence is analyzed by a dependency parser and a coordination analyzer independently. Since the two analyzer may not output compatible structures, employing such an analyzer raises the problem of how to integrate the output of the dependency parser and that of the coordination analyzer. We employ the dual decomposition algorithm, which is a decoding algorithm designed to integrate two analyzer's outputs.

An overview of this chapter is as follows. We first introduce the dual decomposition algorithm via an application to multilingual dependency parsing. Then, we will propose a Japanese dependency parser which is applicable to the algorithm and inherits the advantage of the tournament model. We will also introduce the Japanese coordination analyzer which we employed, to describe which scores in the analyzer are updated by the algorithm. Then, we will define a common representation of the two analyzers to compare and take difference of their outputs, which are requested for updating the penalties. Finally, we will conduct experiments.

Algorithm 4 Overview of the dual decomposition algorithm.

$\{u(i,j)$: penalty variable corresponding to the dependency relation between the head word i and its dependent word $j.\}$
$\{y^*$: optimal dependency structure by the first parser.$\}$
$\{Y$: possible dependency structures by the first parser.$\}$
$\{g(y)$: score of the dependency structure y without considering $u(i,j).\}$
$\{y(i,j)$: indicator function whether y contains the dependency relation $i \leftarrow j.\}$
$\{z^*, Z, f(z)$ and $z(i,j)$: these for the second parser.$\}$

$u(i,j) \leftarrow 0$ for all i, j
loop
  $y^* \leftarrow argmax_{y \in Y} \{g(y) + \sum_{i,j} u(i,j) y(i,j)\}$
  $z^* \leftarrow argmax_{z \in Z} \{f(z) - \sum_{i,j} u(i,j) z(i,j)\}$
  if $y^* = z^*$ **then**
    return y^*
  else
    $u(i,j) \leftarrow u(i,j) + y(i,j) - z(i,j)$
  end if
end loop

7.1 Dual Decomposition Algorithm

Dual decomposition is a decoding algorithm for NP-Hard decoding problems. For multilingual dependency parsing, Koo et al. [23] applied the algorithm to higher-order non-projective dependency parsing, which is an NP-Hard decoding problem [38]. Koo and Collins [22] had proposed a projective dependency parsing algorithm which incorporates second- and third-order features in $O(n^4)$ decoding time. First-order non-projective parsing can be done in $O(n^2)$ using the Chu-Liu-Edmonds algorithm [5, 8, 37]. However, higher-order non-projective parsing is an NP-Hard problem.

Koo's solution using the dual decomposition algorithm employs two parsing models and iteratively updates the penalty variables which correspond to the dependency relations which disagree between the two models. One of the models is the first-order non-projective parsing model [37] which is decoded by the Chu-Liu-Edmonds algorithm [5, 8]. The other model is based on head automata whose decoding algorithm optimizes the set of their dependent words for each word. The model captures higher-

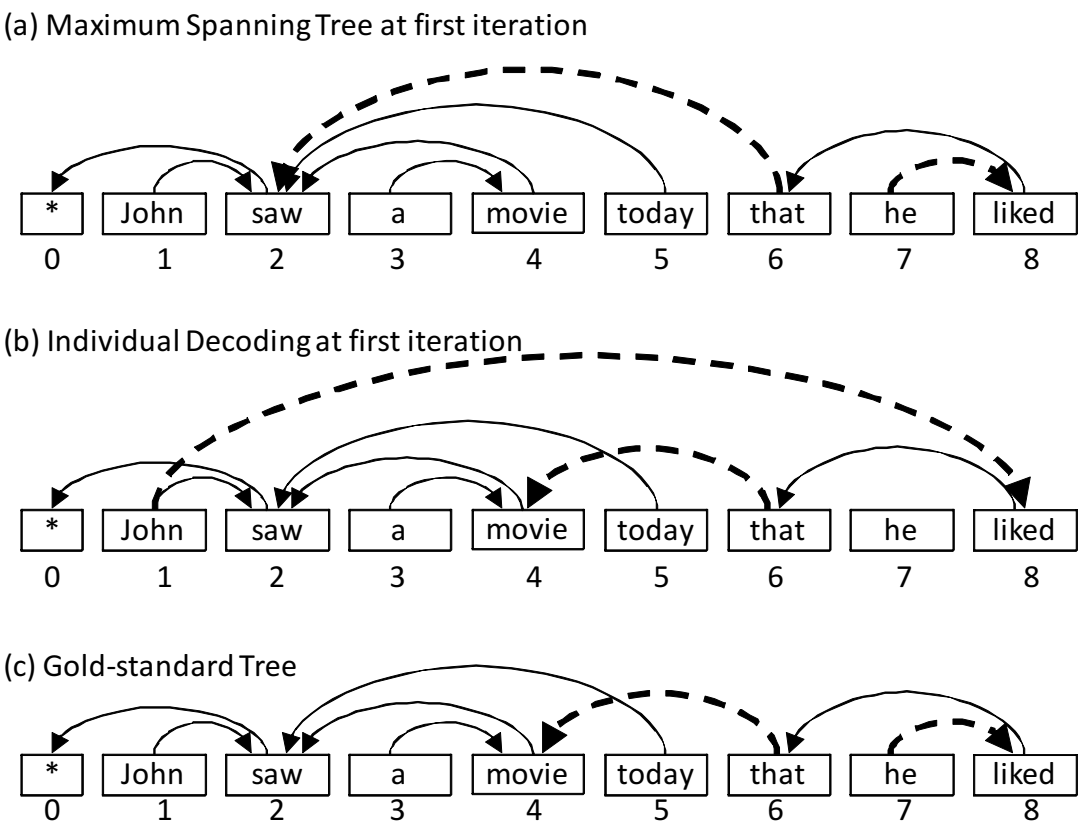


Figure 7.1: An example English sentence. '*' is the artificial sentence ROOT symbol for algorithm convenience. Dependency relations are drawn from dependent words to their head words. This example is cited from the slides of an ACL 2011 tutorial "Dual Decomposition for Natural Language Processing " by Alexander M. Rush and Michael Collins.

order interactions among dependency relations, but there is no guarantee that the output of the model forms a tree. In every iteration of parsing, both the models output their optimal dependency structures independently. The algorithm terminates if their outputs are equivalent. Otherwise, the variables correspond to their disagreed points are updated to penalize their outputs in both models.

The optimal solutions of the two models, y^* and z^*, are defined as

$$y^* = argmax_{y \in Y} \{g(y) + \sum_{i,j} u(i,j)y(i,j)\}$$
$$z^* = argmax_{z \in Z} \{f(z) - \sum_{i,j} u(i,j)z(i,j)\})$$

where Y and Z are the sets of possible dependency structures, and $g(y)$ and $f(z)$ are the scores in these models themselves. An indicator function $y(i,j)$ returns 1 if there is a dependency relation between the head word in the position i to the dependent word in the position j; it returns 0 otherwise. All the penalty variables $u(i,j)$ are initialized as 0 on the beginning of the algorithm.

For example, suppose the two models output the dependency structures shown in Figure 7.1 (a) and (b), respectively. The dependency structure (b) is not a tree. They disagree at the two dependency relations drawn as dashed arrows. The penalty variables $u(i,j)$ corresponding to the disagreements will be updated based on $y(i,j) - z(i,j)$; since all the $u(i,j)$ was initialized as 0, all the nonzero variables are $u(2,6) = +1, u(8,7) = +1, u(8,1) = -1$ and $u(4,6) = -1$ at the end of the first iteration. The dual decomposition algorithm iteratively decodes their optimal solutions with considering the penalties and then updates the penalties until their optimal solutions agree.

7.2 A First-Order Parsing Model based on the Tournament Model

While Koo's dual decomposition employed an MST parser as a dependency parser, the tournament model seems to suitable for Japanese dependency parsing. The algorithm itself is applicable to Japanese bunsetsu-based parsing. However, applying the MSTParser software to Japanese is not a trivial problem because it is not realistic to find the optimal explicit feature combination. Since the number of possible features extracted from a Japanese bunsetsu is much larger than that from an English word, it is

too hard to find the optimal combination. Structure learning algorithms[1] , which is the sort of machine learners employed by MST parsers, does not incorporate polynomial kernels.

Nonetheless, the dual decomposition algorithm requires a score-based parser, but the tournament model is not such a parser. However, as experimented in Chapter 5, the distance from the hyperplane of a binary classifier can be regarded as a score. We will propose a first-order parsing model which inherits the advantage of the tournament model. Then, we will assign first-order scores to the model.

Round Robin Model

We will define a round robin (or all-play-all) model which is an extension of the tournament model. For a dependent which has m candidate heads, the tournament model plays games for some $m-1$ pairs of candidate heads. In contrast to the tournament model, our round-robin model plays games for all $\frac{m(m-1)}{2}$ pairs of candidates. While the time complexity of non-projective dependency parsing with the tournament model is $O(n^2)$, that of the round-robin model is $O(n^3)$.

One of the naive scheme to select the like candidate head is *voting*. That is, the model selects the candidate head whose number of winnings is largest. If the numbers of winnings of two or more candidates are equal, a certain tie-breaking criteria is employed such as selecting the maximally-scored dependency relation.

First-Order Scoring

We will define a score-based round robin model which satisfies the property that the optimal tree is the maximally-scored tree. We employ two classifiers for predicting the head and the label of a dependency relation, respectively.

Unlabeled dependency scores for the model are defined based on the signed distances from the hyperplane which a binary classifier returns. We denote $s_{d,c_1,c_2} = classifyH(d,c_1,c_2)$ where d, c_1 and c_2 are a dependent, its left candidate head and its right candidate head, and $c_1 < c_2$. The function $classifyH(\cdot)$ corresponds to the signed distances from the hyperplane of the binary classifier for *head* prediction. We define

[1]Structure learning for dependency parsing is sort of machine learning algorithms which feed a sentence as a single training example and update its weights based on the difference between the gold-standard tree and the optimal tree decoded with the current weights.

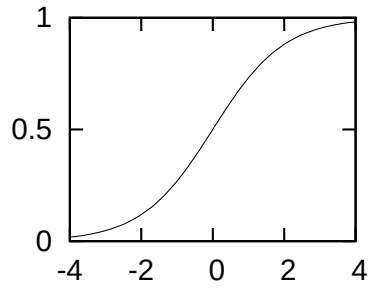


Figure 7.2: $\sigma(x) = \frac{1}{1+e^{-x}}$.

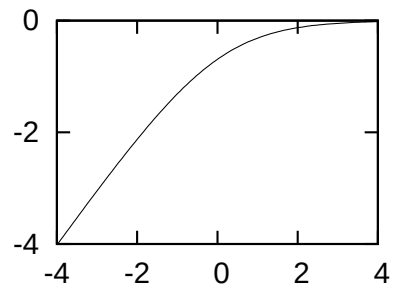


Figure 7.3: $log\sigma(x)$.

$s_{d,c,c} = 0$. We defined the positive class of the classifier as "the left candidate is more appropriate". Therefore, if $classifyH(\cdot)$ returns a positive, the left candidate is judged to be more appropriate, and vice versa. The score of an unlabeled dependency relation is defined as

$$score(d \rightarrow h)_{unlabeled} = \min_{c \in \{d+1,\ldots,n-1\}} \log \sigma(sign(c-h) \cdot s_{d,c,h}).$$

Since $sign(c-h)$ returns -1 if $c < h$, it ensures that "h won" is always the positive class and vice versa. A sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ is used to convert a signed distance from the hyperplane $\in \mathbb{R}$ to pseudo probability $0 < p < 1$, as shown in Figure 7.2. The function $\sigma(\cdot) > 0.5$ when h won and $\sigma(\cdot) < 0.5$ when h lost. Applying the logarithm function is for calculation convenience. A candidate h plays games against each of all other candidates and the equation takes the worst game score as its dependency score. As shown in Figure 7.3, applying $\log \sigma(\cdot)$ preserves their magnitude relation.

The score of a labeled dependency relation is defined as the combination of its labeling score and its unlabeled dependency score. The labeling score is defined as

$$score(d \xrightarrow{l} h)_{labeling} = \log \frac{\sigma(s_{d,h,l})}{\max_{l' \in L} \sigma(s_{d,h,l'})} \tag{7.1}$$

where $s_{d,h,l} = classifyL(d,h,l)$ is a signed distance from the hyperplane which a binary classifier for predicting the *label* returns. The set of possible labels $L = \{D,P,I,A\}$. The score of a labeled dependency relation is defined as

$$score(d \xrightarrow{l} h)_{labeled} = score_{unlabeled}(d \rightarrow h) + score_{labeling}(d \xrightarrow{l} h). \tag{7.2}$$

We assume that dependency relations are independent each other. Therefore, the

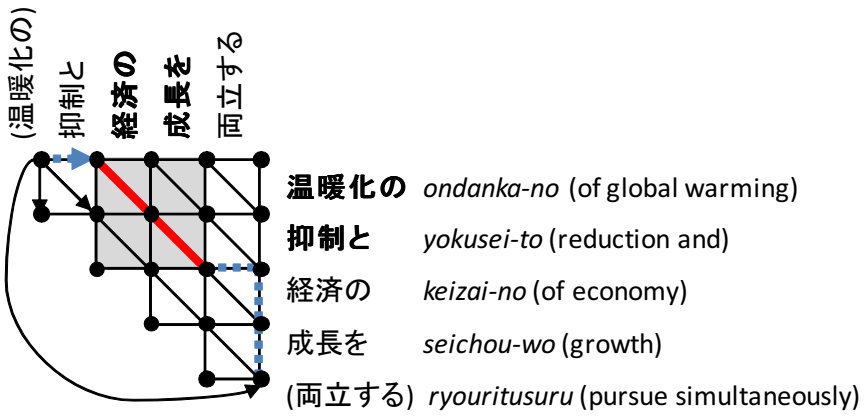


Figure 7.4: An example edit graph which the Okuma's Japanese coordination analyzer [48] constructs. All the links are directed arcs. However, we will draw directed arcs in edit graphs as undirected arcs. "Pursue reduction of global warming and growth of economy simultaneously."

score of a dependency tree y and the optimal tree y^* are defined as

$$g(y) = \sum_{(d \xrightarrow{l} h) \in y} score(d \xrightarrow{l} h)$$

$$y^* = argmax_{y \in Y}\ g(y)$$

These definitions, maximizing the sum of arc scores, are identical to those of a non-projective first-order MST model, which is employed in Section 7.1. While the decoding algorithm for the MST model is Chu-Liu-Edmonds algorithm, the round robin model can be decoded by a simpler algorithm because of the Japanese HEADFINAL constraint. The round robin model cannot incorporate dynamic features, in contrast to the tournament model.

7.3 Japanese Coordination Analyzer

We employed the Okuma's similarity-based Japanese coordination analyzer [48, 49], which is an extension of a Shimbo and Hara's English coordination analyzer [58]. The

Shimbo and Hara's machine learning-based analyzer is an extension of the rule-based Kurohashi and Nagao's coordination analyzer for Japanese [30]. All of these models, including Okuma's, capture syntactic and semantic similarities between components of coordinated conjuncts.

For example, in the sentence shown in Figure 7.4 are *similar* in the following senses. The two conjuncts "温暖化の抑制"(reduction of global warming) and "経済の成長"(growth of economy) are coordinated. The two conjuncts are similar because both of the conjuncts contain the particle "*no*" (の). Moreover, "温暖化"(global warming) and "経済"(economy) are defined as semantically similar in some language resources such as thesauri. Additionally, "抑制"(reduction) and "成長"(growth) are also semantically similar. The particle "*to*" (と) is a marker of the existence of a coordinated structure, which corresponds to an English coordinate conjunction "and". However, not all the instances of the particle are markers of coordinated structures.

At the beginning of coordination analysis of a sentence, the Okuma's analyzer constructs the triangular directed graph shown in Figure 7.4. All the bunsetsus of the sentence is written on both the top and the right of the graph. For example, the topmost-leftmost cell corresponds to the similarity between "温暖化の" and " 抑制と ". The leftmost bunsetsu of a sentence does not have its corresponding column, Similarly the rightmost bunsetsu does not have its corresponding row. The graph is referred to as an edit graph, or alignment graph.

A node typically has three outgoing arcs. A diagonal arc corresponds to the *Substitution* edit operation. In other words, the two bunsetsus written above and on the right of the arc are aligned. A horizontal arc corresponds to the *Insertion* edit operation, which indicates that the two bunsetsus does not be aligned. Similarly, a vertical arc corresponds to the *Deletion* edit operation. In Figure 7.4, "温暖化の" and "経済の" are aligned, and "抑制と" and "成長を" are also aligned.

The optimal alignment path is decoded from the top-left corner node to the bottom-right corner node using a Viterbi algorithm. Scores are assigned to every arc of the graph calculated with similarity features and learned weights of these features.

The decoded path may contain some coordinated structures. The gray box in the figure denotes a coordinated structure. More specifically, a label is assigned for each arc to distinguish whether the arc corresponds to inside or outside of a coordinated structure. The path shown in Figure 7.4 denotes the box because the labels of the two *Substitution* arcs are *Inside*, and the labels of the other two arcs are *Outside*. A coordinated structure consists of $N(\geq 2)$ conjuncts is represented as $N-1$ boxes on

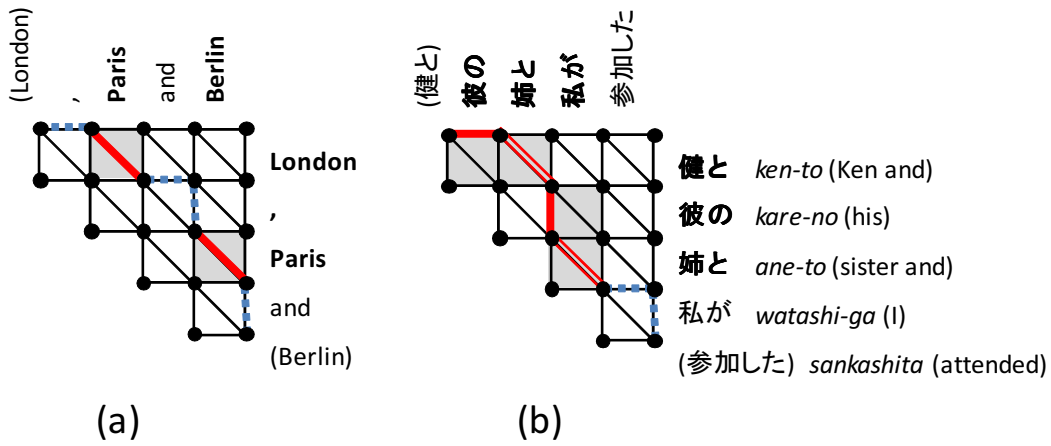


Figure 7.5: Representations of an English and a Japanese coordinated structures consist of three conjuncts. Bold solid lines, dotted lines and double lines correspond to *Inside*, *Outside* and *End*, respectively. The sentence (b): "Ken, his sister and I attended."

the graph. Each box represents the correspondence of neighboring conjuncts. An edit graph in English word-based coordination analysis is shown in Figure 7.5(a). The set of arc labels $\{Inside, Outside\}$ is sufficient because coordinate conjunctions such as "and" or "," are always placed between conjuncts. In contrast, the property does not hold in Japanese bunsetsu-based coordination analysis. As shown in Figure 7.5(b), a coordinate conjunction "*to*" (と) is included in a conjunct bunsetsu "抑制と". Therefore, with the set of labels, a coordination analyzer cannot distinguish between a 3-conjunct coordinated structure and two 2-conjunct coordinated structures. To solve the problem, another arc label *End* was employed, which marks the ends of conjuncts.[2]

The differences between the Okuma's model and a Shimbo and Hara's model is not only in their features but also that the former introduced the bypass arc. The arc connects the top-left corner node and the bottom-right corner node of an edit graph directly. The arc corresponds to that there are no coordinated structures in the sentence. Without the bypass arc, an analyzer may reach the same conclusion based on local features assigned to each arc. In contrast, global features such as sentence length are available for the bypass arc.

[2]The implementation we used for experiments employs a more complicated label set and a possible consecutive arc list.

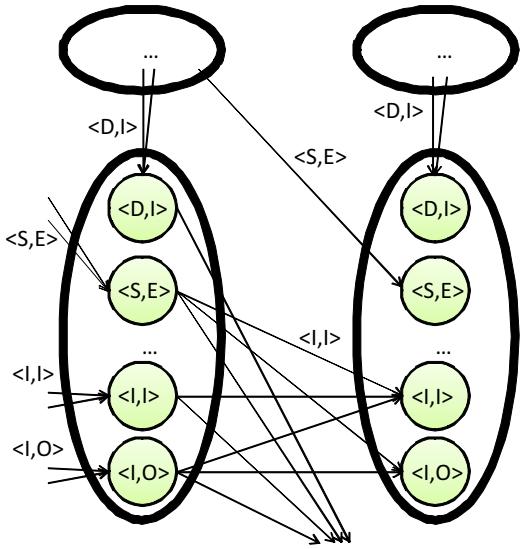


Figure 7.6: A node is implemented as a set of child nodes with different labels. An oval is a node group containing child nodes. A node group corresponds to each of nodes in an edit graph. Not all the arcs are drawn in this figure.

Implementation Specification

We used Okuma's implementation[3]. We modified the code to output the score table and possible consecutive arc label list of the analyzer. The model assigns scores to not only arcs but also nodes. As we will describe in Section 7.4, our dual decomposition algorithm modifies the scores of nodes.

The score of a node represents the score of consecutive arcs. As shown in Figure 7.6, a node in an edit graph is implemented as a set of child nodes with different labels. The label of a child node is identical to the label of its incoming child arcs. A consecutive label pair (p,s) is implemented as a child arc with the label s which outgoes from the child node with the label p. The name of an arc label is a pair of the first characters of an edit operation $\in \{S(ubstitution), I(nsertion), D(eletion)\}$ and a position of the arc in a coordinated structure $\in \{I(nside), O(utside), E(nd)\}$. For example, the label $\langle S,E\rangle$ denotes a substitution at the end of a coordinated structure. The possible consecutive arc label list prohibits some of unnatural consecutive arcs. There are no child arcs between such label pairs.

[3] An implementation of a Shimbo and Hara's model is released at `http://cl.naist.jp/project/coordination/en/?Code`.

On decoding from an optimal path to boxes, the analyzer refers to the two consecutive label list: *BoxStartList* and *BoxEndList*. *BoxStartList* enumerates the label pairs which are regarded as the beginning of a box, such as $(\langle I,O\rangle, \langle S,E\rangle)$. Similarly, *BoxEndList* enumerates the conditions to be the end of a box such as $(\langle S,E\rangle, \langle D,O\rangle)$. The coordination analyzer assigns beginnings and ends of boxes to a path by using the lists, and then makes pairs of a beginning and an end from the assigned marks. The model was not designed for detecting nested coordinated structures.

Suppose $score_{Viterbi}(z)$ for a path z is the resulting score of a Viterbi algorithm applied for an edit graph with the bypass arc. The optimal path z^* is

$$f(z) = score_{Viterbi}(z)$$

$$z^* = argmax_{z \in Z}\ f(z).$$

7.4 Designing Penalties in Dual Decomposition

Common Representation

We have to define a common representation to integrate the outputs of the dependency parser and the coordination analyzer by using a dual decomposition algorithm. In the example of dual decomposition described in Section 7.1, both of the parsers output a dependency graph. The common representation of the two parsers is the indicator functions $y(i,j)$ and $z(i,j)$. In contrast, our situation cannot be so straightforward; the parser outputs a dependency graph and the coordination analyzer outputs the spans of coordinated structures. A common representation is necessary to judge the equivalence of their optimal solutions, which corresponds to $y^* = z^*$ in the previous example, and to calculate the difference between them, which corresponds to $y(i,j) - z(i,j)$.

We define a common representation as shown in Figure 7.7. The path in Figure 7.5(b) can be represented as Figure 7.7(b) with bunsetsu indices. Each of the bunsetsus in the positions 0 and 3 is a single-bunsetsu conjunct. The other conjunct spans the bunsetsus in the positions 1 and 2. In our common representation, the output of the coordination analyzer is regarded as a partial dependency tree shown in (b'). This conversion from (b) to (b') ignores bunsetsu indices of the beginnings of conjuncts and add dependency relations based on the indices of the ends of conjuncts. All the labels of such dependency relations are P, which is the label for a parallel relation in Kyoto

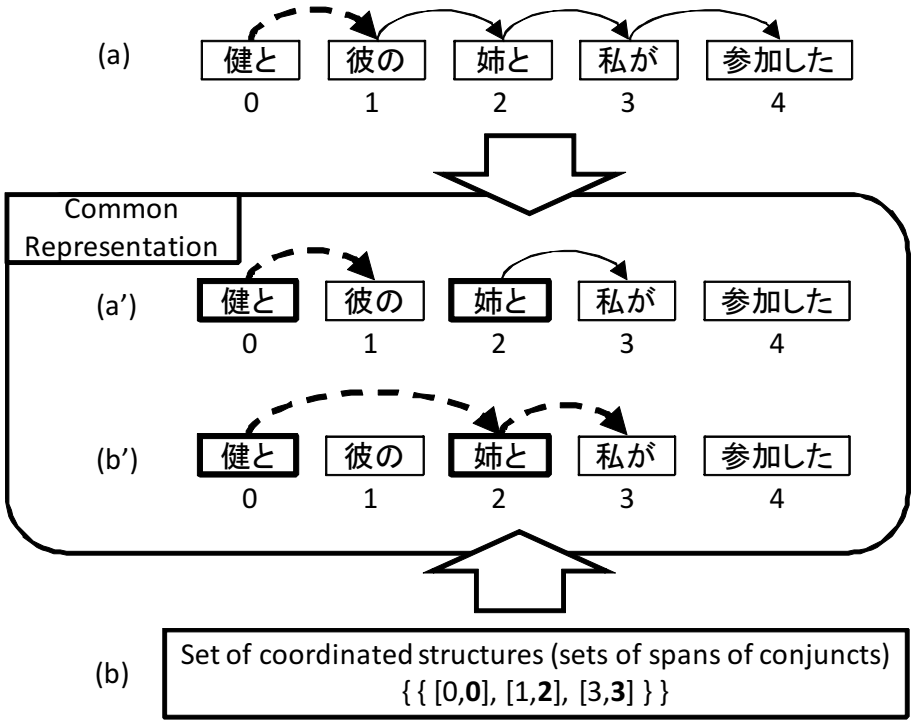


Figure 7.7: The common representation between a dependency parser's and a coordination analyzer's outputs. Dashed arcs are parallel (label P) dependency relations. Solid arcs are normal (label D) dependency relations.

Text Corpus. The conversion from a dependency parser's output (a) to our common representation (a') refers to the converted coordinated structure (b'). As shown in (a'), all the dependency relations $(d \xrightarrow{l} h)$ such that $(d \xrightarrow{l'} h') \in$ (b') are extracted.

Then, we introduce the penalty variables u, just like the example in Section 7.1. However, we add the dependency label, and inverse the positions of a dependent j and its head i to adapt the Japanese notation. Therefore, $u(j \xrightarrow{l} i)$ denotes the penalty for the dependency relation from j to i with the label l.

Suppose the conversion functions for the dependency parser and the coordination analyzer are $C_d(\cdot)$ and $C_c(\cdot)$, respectively. The functions are defined as indicator functions, similar to $y(i,j)$ and $z(i,j)$ in Section 7.1. That is, they return 1 if there is a dependency relation from j to i with the label l, or return 0, otherwise. Suppose Figure 7.7(a) is a dependency tree y and (b) is an alignment path z, $C_d(y)(0 \xrightarrow{P} 1) = 1$, $C_d(y)(2 \xrightarrow{D} 3) = 1$, $C_c(z)(0 \xrightarrow{P} 2) = 1$ and $C_c(z)(2 \xrightarrow{P} 3) = 1$ are all of their nonzero elements.

The Algorithm

The dual decomposition algorithm for our task is shown in Algorithm 5. Its basic structure is similar to Algorithm 4. This algorithm terminates when not only their common representations agree $(c_d(y^*) = c_c(z^*))$ but also they disagree τ times (lines 6-7). The parameter τ is the maximum number of iterations. As Koo et al. [23] described in their implementation detail, update step sizes should be dropped according to iterations proceed for efficient convergence. We set the update step size in the t^{th} iteration to $\frac{1}{t}$ of that in the first iteration (line 9).

The function $y(j \xrightarrow{l} i)$ to penalize the score of a dependency tree $g(y)$ is almost an indicator function. Unlike $y(i,j)$ in Section 7.1 which returns 1 or 0, $y(j \xrightarrow{l} i)$ returns $UnitScore_{parse} \cdot \delta$ when there is the dependency relation from j to i with the label l in y, instead of returning 1. Returning the value plays the role of normalization. Unlike probabilistic models, the scores of our parser and coordination analyzer were not normalized each other. The value $UnitScore_{parse}$ is defined as the score of the optimal dependency tree in the first iteration divided by the number of dependency relations in the sentence, which is $n-1$ for a sentence consists of n bunsetsus. That is,

$$UnitScore_{parse} = \left| \frac{g(y^*)}{n-1} \right|.$$

Algorithm 5 The dual decomposition algorithm for our task.

$u(j \xrightarrow{l} i) \leftarrow 0$ for all i, j, l
$t = 1$
loop
  $y^* \leftarrow argmax_{y \in Y} \{g(y) + \sum_{i,j,l} u(j \xrightarrow{l} i) y(j \xrightarrow{l} i)\}$
  $z^* \leftarrow argmax_{z \in Z} \{f(z) - \sum_{i,j,l} u(j \xrightarrow{l} i) z(j \xrightarrow{l} i)\}$
  if $c_d(y^*) = c_c(z^*)$ **or** $t = \tau$ **then**
    return $\langle y^*, z^* \rangle$
  else
    $u(j \xrightarrow{l} i) \leftarrow u(j \xrightarrow{l} i) + \frac{1}{t}\{c_d(y^*)(j \xrightarrow{l} i) - c_c(z^*)(j \xrightarrow{l} i)\}$
  end if
  $t \leftarrow t + 1$
end loop

---

function $y(j \xrightarrow{l} i)$
  if $j \xrightarrow{l} i \in y$ **then**
    return $+UnitScore_{parse} \cdot \delta$
  else
    return 0
  end if
end function

---

function $z(j \xrightarrow{l} i)$
  if $l = P$ **then**
    if $\xrightarrow{p} (j,i) \xrightarrow{s} \in z$ **and** $(p,s) \in BoxEndList$ **then**
      return $+UnitScore_{coord} \cdot \kappa$
    else
      return 0
    end if
  else
    $A \leftarrow \{\xrightarrow{p} (j,i') \xrightarrow{s} \mid \xrightarrow{p} (j,i') \xrightarrow{s} \in z$ **and** $(p,s) \in BoxEndList\}$
    return $\sum_{(j,i') \in A} -UnitScore_{coord} \cdot \kappa$
  end if
end function

The parameter δ specifies the update step size for dependency parser's scores in the first iteration.

The function $z(j \xrightarrow{l} i)$ penalizes the scores of a coordination analyzer's path $f(z)$. How the scores are updated depends on its label l. The definition is similar to $y(j \xrightarrow{l} i)$ if $l = P$, as shown in the lines 22-26 in Algorithm 5. It is almost an indicator function about whether the following two conditions are satisfied. Suppose $\xrightarrow{p}(j,i)\xrightarrow{s}\in z$ is the pair of the incoming child arc with the label p and the outgoing child arc with the label s of the node at the position (j,i) in an edit graph. The first condition is that a path z includes the consecutive arc. The other is that the arc pair (p,s) corresponds to the end of a coordinated structure. The unit score for our coordinated analyzer is defined as.

$$UnitScore_{coord} = \left|\frac{f(z^*)}{n-1}\right|.$$

The parameter κ is for our coordination analyzer, similar to δ. These parameters also play the role of controlling which of dependency parser's scores and coordination analyzer's scores are updated more drastically. Unlike dependency trees, the length of a path of our coordination analyzer is not a constant. Therefore, we defined the unit score by dividing by $n-1$ instead of its path length.

The definition for $l \neq P$ is more complicated because it is not a trivial problem to encode such dependency relations into an edit graph. We focused the fact that if a dependency tree includes the dependency relation $j \xrightarrow{l} i$, the tree cannot include another dependency relation $j \xrightarrow{l'} i'$ such that $l \neq l'$ and/or $i \neq i'$ because of the SINGLEHEAD constraint. As shown in the lines 28 and 29, when $u(j \xrightarrow{l} i)$ is being updated with a displacement $\Delta < 0$, the scores for coordinated structures which end at the node (j,i') for each $i' \in \{j+1,\ldots n-1\}$ are decreased. This updating procedure indirectly promotes dependency relations $j \xrightarrow{l} i$ such that $l \neq P$ relatively, with degrading their disjoint dependency relations. It contrasts with the direct interaction which occurs when $l = P$; the score for coordinated structures which end at the node (j,i) is promoted.

At most one arc $a \in z$ is penalized by an indirect update. A valid coordinated structure is denoted as a box whose both width and height are at least 1 on an edit graph. Therefore, each arc in an alignment path z is penalized in the summation at line 5 at most once.

7.5 Experiments

7.5.1 Settings

Our experimental settings are as follows. We used Kyoto Text Corpus Version 4.0. We corrected a few critical errors that crash the parser in the corpus. The binary classifier for dependency head prediction was *opal*, which is an implementation of Passive-Aggressive-I binary classifiers, with the second degree polynomial kernel. The other options for *opal* are `N=0 a=PA1 ave=1 s=1 iter=40 c=0.00005 M=0`. We used the standard features and the additional features. We set the number of iterations for training the Okuma's coordination analyzer to 5000. Parameter averaging was performed.

Another PA-I classifiers were employed for dependency labeling. Their options and features are as same as for the PA-I for head prediction. The labeling algorithm is one-versus-rest. That is, we trained the four binary classifiers: label D versus the other labels, label P versus the other labels, label I versus the other labels and label A versus the other labels. On testing, the four classifiers were applied to a dependency relation and the label corresponds to the classifier with the maximum signed distance is selected.

These parser and coordination analyzer are the baseline. They were also the input of the dual decomposition algorithm.

7.5.2 Preliminary Experiment: Accuracy of Coordination Analyzer

The accuracies of the coordination analyzer without the dual decomposition algorithm are shown in Table 7.1. The two top experiments are regular experimental settings, whose training data and test data are disjoint. The two bottom experiments are unrealistic self-training ones, whose training data and test data are identical. The definition of sentence accuracies is identical to that for dependency parsing. These sentence accuracies are relatively high compared to the other columns because around 45% of the sentences of the corpus does not contain any coordinated structures.

A coordinated structure was evaluated as one or more boxes. Similar to the description in Section 7.3, a coordinated structure which consists of N conjuncts is decomposed into the $N-1$ neighboring pairs, and then evaluated as the $N-1$ individual components. The evaluation measure *Scope of coordinations* is the tighter measure;

Training Data	Test Data	Sent. Acc.	Scope of coordinations			End of conjuncts		
			P	R	F	P	R	F
Jan. 1st - 8th	Jan. 9th	57.4	30.2	33.3	31.7	45.5	50.1	47.7
Jan. 1st - 8th	**Jan. 10th**	55.3	31.1	36.2	33.4	45.2	52.6	48.6
Jan. 1st - 8th		61.0	38.7	39.0	38.9	55.8	56.2	56.0
Jan. 10th		74.7	60.8	58.8	59.8	74.4	72.0	73.2

Table 7.1: Coordination Analysis Accuracy.

the correctness of a conjunct pair is judged in whether its all the bunsetsu indices are completely identified. The other measure *End of conjuncts* is the looser measure which ignores all the indices of the beginning of conjuncts. For example, the 3-conjunct coordinated structure $\{[0,0],[1,2],[3,3]\}$ in Figure 7.7(b) is decomposed to $\langle[0,0],[1,2]\rangle$ and $\langle[1,2],[3,3]\rangle$. With the former measure, $\langle[0,0],[1,2]\rangle$ is correct if and only if $\langle[0,0],[1,2]\rangle \in$ gold-standard. With the latter measure, $\langle[0,0],[1,2]\rangle$ is correct if and only if $\langle[*,0],[*,2]\rangle \in$ gold-standard where $*$ is don't-care. The latter measure is similar to that for a dependency tree because a label-P dependency relation connects the ends of conjuncts. We evaluated precision, recall and F-measure.

As shown in Table 7.1, our coordination analysis performance is low on the whole. In the upper half experiments, their F-measures with the latter measure are below 50%. Even in the unrealistic self-training experiments with the larger training data, the F-value is 56%. Unlike the SVMs in dependency parsers, whose self-training accuracy is almost 100%, the coordination analyzer seems not to learn sufficient information. However, compared to the corpora in Shimbo and Hara, and Okuma's experiments, Kyoto Text Corpus seems to be difficult to analyze for similarity-based coordination analyzers. Especially, the conjuncts of a predicate coordination are rarely similar each other.

In the following experiments, we use the setting: the training data is January 1st-8th and the test data is January 10th. It is the baseline performance. The output with the setting are input into the dual decomposition algorithm.

7.5.3 Dual Decomposition with Unlabeled Dependency Parser

First, we evaluated how much the dual decomposition algorithm improves labeled dependency accuracies with only the coordination analyzer's output. In this experi-

ment, the dependency parser does not estimate dependency labels but assume all the labels are D, instead. The label D is the most frequent label in Kyoto Text Corpus. The labels of around 90% of dependency relations are D. We added the following statement to the initialization section (lines 1 and 2) of Algorithm 5.

$$score(d \xrightarrow{P} h)_{labeled} \leftarrow score(d \xrightarrow{D} h)_{labeled} - \varepsilon; \text{ for all } d, h$$

where $\varepsilon > 0$ and $\varepsilon \ll UnitScore_{parse}$. The labeling score for a label-P dependency relation is defined as slightly smaller than that for label D. In this experiment, the set of possible dependency labels $L \in \{D, P\}$.

Results are shown in Table 7.2. The maximum number of iterations, τ , of the dual decomposition algorithm was set to 30. We first tuned δ, and then κ, greedily. The number of correct dependency relations is improved the most with $\delta = 0.25, \kappa = 0.08$. Since this test data consists of 13997 dependency relations, improvement of +33 is +0.24% in a dependency accuracy. The number of correct labeled dependency relations, defined as whether both their heads and labels are correct, improved by +267, which is +1.91% in a labeled dependency accuracy. Around 320 sentences were not converged within τ iterations. In these cases, the accuracy of the two analyzers are evaluated individually by ignoring the fact that the outputs of the two analyzers do not agree. The two bottom parameter settings of Table 7.2 intend to simulate the situation that one of the two analyzers blindly accepts the other analyzer's output. Except for their unlabeled dependency accuracies, both their dependency and coordination accuracies drastically dropped.

Coordination analysis accuracies are improved with some parameter settings. The F-measure was achieved the highest with $\delta = 0.25, \kappa = 0.2$. $\#tp = -\#fn$ because both of them are conjunct pairs $\in$ gold-standard conjunct pairs. Table 7.2 shows that this experimental setting does not have ability to reduce false positives. The improvement in F-measure is provided by turning false negatives into true positives.

However, this experimental setting seems to be unpractical. We can construct a labeled dependency parser with labeled dependency annotations, or unlabeled dependency annotations and coordinated structure annotations with a certain automatic conversion. If such labeled dependency annotations are available, the one-versus-rest dependency labeler is expected to achieve higher accuracy than this experimental setting because of high-performance machine learners such as SVMs and PAs.

Parameters		# sent.	# correct dep.		Scope of coordinations			
δ	κ	!conv.	unlabeled	labeled	# tp	# fp	# fn	F(%)
0.01	1.0	349	+5	+161	0	+94	0	-1.30
0.05		341	+9	+166	+2	+91	-2	-1.12
0.1		330	+11	+168	+2	+91	-2	-1.12
0.2		313	+18	+178	+2	+93	-2	-1.15
0.25		305	+24	+183	+1	+92	-1	-1.20
0.3		297	+18	+190	+2	+91	-2	-1.12
0.5		277	+9	+185	0	+83	0	-1.15
1		33	-281	+118	-14	+65	+14	-1.91
0.25	0.01	328	+31	+265	0	0	0	0.00
	0.05	324	**+33**	+266	+1	0	-1	+0.07
	0.08	323	**+33**	**+267**	+2	0	-2	+0.14
	0.1	323	**+33**	+264	+1	+3	-1	+0.03
	0.2	318	+31	+257	**+8**	+7	**-8**	**+0.48**
	0.5	316	+26	+230	+5	+35	-5	-0.14
	1.0	305	+24	+183	+1	+92	-1	-1.20
	2.0	311	+26	-16	-4	+276	+4	-3.81
0.0	1000.0	738	0	0	-339	+2992	+339	-31.43
1000.0	0.0	3	-256	+190	0	0	0	0.00

Table 7.2: Results of dual decomposition with the unlabeled dependency parser. # sent. !conv: the number of sentences which were not converged. # tp, # fp, # fn: the numbers of true positives, false positives and false negatives, respectively.

Parameters		# sent.	# correct dep.		Scope of coordinations			
δ	κ	!conv.	unlabeled	labeled	# tp	# fp	# fn	F(%)
0.01	1.0	447	+3	+2	+10	+61	-10	-0.15
0.05		437	+8	+7	+9	+57	-9	-0.16
0.1		427	+14	+9	+9	+56	-9	-0.15
0.15		416	+18	+16	+9	+56	-9	-0.15
0.18		407	+29	+24	+13	+52	-13	+0.19
0.3		376	+16	+5	+11	+44	-11	+0.16
0.5		330	-7	-22	+13	+36	-13	+0.42
1.0		241	-68	-129	-1	+6	+1	-0.16
0.18	0.01	415	+33	+27	0	0	0	0.00
	0.03	415	**+36**	**+29**	0	+2	0	-0.03
	0.05	415	+33	+27	+1	+1	-1	+0.06
	0.1	412	+34	+27	+4	-7	-4	+0.39
	0.3	407	+32	+27	+13	**-10**	-13	+1.09
	0.4	405	+32	+27	**+14**	-7	**-14**	**+1.11**
	0.5	405	+28	+23	+13	-1	-13	+0.95
	1.0	407	+29	+24	+13	+52	-13	+0.19
0.0	1000.0	454	0	0	-78	+1644	+78	-17.46
1000.0	0.0	0	-256	-345	0	0	0	0.00

Table 7.3: Results of dual decomposition with the labeled dependency parser.

7.5.4 Dual Decomposition with Labeled Dependency Parser

Table 7.3 shows the results with the labeled dependency parser. In this experiment, labeled dependency scores had been assigned with the one-versus-rest classifiers as in Equations 7.1 and 7.2. The number of correct dependency relations is improved most with $\delta = 0.18, \kappa = 0.03$. With the two bottom parameter settings, all the accuracies are drastically dropped.

In this experimental setting, false positives were reduced. The F-measure of coordinated structure is highest with $\delta = 0.18, \kappa = 0.04$. The improvement of 1.11% is higher than that of the previous experiment.

Parser	δ, κ	Unlabeled Acc.	Labeled Acc.
Test data: January 10th			
unlabeled	-	90.06 (12605/13997)	83.01 (11619/13997)
	0.25, 0.08	90.29 (12638/13997)	84.92 (11886/13997)
labeled	-	90.06 (12605/13997)	87.83 (12293/13997)
	0.18, 0.03	90.31 (12641/13997)	88.03 (12322/13997)
Test data: January 9th			
unlabeled	-	89.56 (9838/10986)	82.81 (9097/10986)
	0.25, 0.08	89.65 (9849/10986)	84.86 (9323/10986)
labeled	-	89.56 (9838/10986)	87.47 (9610/10986)
	0.18, 0.03	89.59 (9842/10986)	87.47 (9609/10986)

Table 7.4: Dependency accuracies before and after applying dual decomposition with the development and a test data. Parameters for the experiments without the dual decomposition algorithm are denoted with '-'.

7.5.5 Accuracy on Another Test Data

Table 7.4 shows the results on another test data. The best parameters estimated with the development data were applied to the test data.

The improvements on the data are relatively small. Our preliminary experiments on the data with various parameter settings suggest that the dual decomposition algorithm seems not to be suitable for the test data, rather than the parameters are not suitable. The reason may be that their parsing accuracy without the dual decomposition algorithm was significantly dropped because of the following reasons, compared to Table 4.2. Unlike the machine learner of the former parser is a PA with the second-degree polynomial kernel, that of the latter is an SVM with the third-degree polynomial kernel. The former does not use dynamic features but the latter does. The low accuracy implies that the scores in the round robin model on the test data is low-quality and therefore prevents the dual decomposition algorithm from improvement. Moreover, coordination analysis accuracy without dual decomposition is also lower than that on January 10th, as shown in Table 7.1.

7.5.6 Passive Selection of $\langle y^*, z^* \rangle$ based on Best Primal Score

In this subsection, we investigate how to select the best $\langle y^*, z^* \rangle^{(t)}$, which is the pair of y^* and z^* at the t^{th} iteration. In the experiments above, we used the constant τ and evaluated accuracies with $\langle y^*, z^* \rangle$ at the final iteration, that is, we evaluated with $\langle y^*, z^* \rangle^{(\tau)}$ if they have never agreed. We define *PrimalScore*, a function for evaluating $\langle y^*, z^* \rangle$ without using the gold-standard data, as

$$PrimalScore(\langle y, z \rangle) = \frac{g(y) + \sum_{i,j,l} u(j \xrightarrow{l} i) y(j \xrightarrow{l} i)}{UnitScore_{parse}} + \frac{f(z) + \sum_{i,j,l} u(j \xrightarrow{l} i) z(j \xrightarrow{l} i)}{UnitScore_{coord}}.$$

Then, we replace the line 7 of Algorithm 5 with

$$\textbf{return } argmax_{\langle y^*, z^* \rangle^{(t)}} \ PrimalScore(\langle y^*, z^* \rangle^{(t)}).$$

We also investigate the upper bound with the algorithm; the optimal number of iterations is given as an oracle for each sentence. That is, we replace the line 7 with

$$\textbf{return } argmax_{\langle y^*, z^* \rangle^{(t)}} \ Accuracy(\langle y^*, z^* \rangle^{(t)})$$

where $Accuracy(\cdot)$ refers the gold-standard data.

Results are shown in Table 7.5. The selection criteria based on the best PrimalScore results modest improvements but it avoids drastic degradations of accuracies. The selection criteria is one of the solutions for the problem that accuracies depend on parameter selection critically. The highest number of labeled dependency relations is achieved with $\delta = 1.0, \kappa = -1.0$ and the selection based on PrimalScore, except for the settings with oracles. With $\kappa < 0$, scores of the coordination analyzer is updated to the inverse direction. The settings with oracles achieved +121 and +123 improvements in the numbers of correct unlabeled and labeled dependencies, respectively. They are about 0.9% improvement. It is the upper bound of the accuracy by the best iteration selection.

7.5.7 Impact of Coordination Analysis Accuracy

In this subsection, we investigate the impact of the accuracy of the coordination analyzer on the final accuracies after applying the dual decomposition algorithm. We used

Parameters		# Unlabled correct dep.			# Labeled correct dep.		
δ	κ	final	bestprimal	oracle	final	bestprimal	oracle
with unlabeled parser							
1.0	-1.0	-65	**+15**	+88	+235	**+293**	**+606**
0.01	1.0	+5	+1	+5	+161	+268	+551
0.1	1.0	+11	+5	+25	+168	+269	+566
0.2	1.0	+18	+8	+45	+178	+266	+581
0.25	0.01	+31	+9	+59	+265	+266	+586
0.25	0.08	**+33**	+10	+62	**+267**	+261	+589
0.25	0.1	**+33**	+9	+62	+264	+265	+589
0.25	0.5	+26	+9	+58	+230	+264	+591
0.25	1.0	+24	+9	+56	+183	+262	+590
0.5	1.0	+9	+5	+77	+185	+259	+604
1.0	1.0	-38	+19	**+98**	+162	+273	+602
5.0	1.0	-281	-73	+92	+118	+244	+516
1000.0	0.0	-256	0	+59	+190	0	+447
with labeled parser							
1.0	-1.0	-68	+16	+110	-129	+14	+112
0.01	1.0	+3	+1	+5	+2	+1	+3
0.1	1.0	+14	+3	+31	+9	+2	+26
0.18	0.01	+33	+5	+61	+27	+5	+59
0.18	0.03	**+36**	+5	+64	**+29**	+5	+60
0.18	0.1	+34	+5	+64	+27	+4	+60
0.18	0.5	+28	+5	+64	+23	+4	+61
0.18	1.0	+29	+5	+60	+24	+4	+59
0.5	1.0	-7	+11	+100	-22	+8	+104
1.0	1.0	-52	**+19**	**+121**	-125	**+16**	**+123**
5.0	1.0	-221	-16	+96	-346	-29	+91
1000.0	0.0	-256	0	+59	-335	0	+57

Table 7.5: Results of dual decomposition by selecting the $\langle y^*, z^* \rangle$ of the final iteration (final), based on primal scores (bestprimal) and based on accuracies (oracle). Test data is January 10th.

Parameters		# sent.	# correct dep.		Scope of coordinations			
δ	κ	!conv.	unlabeled	labeled	# tp	# fp	# fn	F(%)
with unlabeled parser								
0.1	1.0	206	+20	+592	-45	+125	+45	-6.31
0.2	1.0	190	+39	+608	-42	+120	+42	-5.99
0.25	0.08	169	+51	+668	+2	+2	-2	+0.08
0.3	1.0	178	+40	+609	-41	+118	+41	-5.87
0.5	1.0	161	+59	+630	-37	+111	+37	-5.43
0.5	0.02	136	+63	+682	+3	-1	-3	**+0.23**
0.5	0.08	135	**+64**	**+685**	+4	+2	-4	+0.21
0.5	0.2	133	+63	+683	+4	+10	-4	-0.01
0.8	1.0	139	+51	+632	-39	+105	+39	-5.41
1.0	1.0	124	+45	+645	-38	+100	+38	-5.22
0.0	1000.0	668	0	0	-550	+2870	+550	-56.01
1000.0	0.0	0	-6	+666	0	0	0	0.00
with labeled parser								
0.1	1.0	233	+27	+26	-13	+64	+13	-2.62
0.18	0.03	212	+48	+54	+5	-2	-5	+0.39
0.2	1.0	213	+46	+54	-13	+65	+13	-2.64
0.3	0.05	182	**+62**	+74	+7	-3	-7	+0.55
0.3	1.0	189	+59	+69	-12	+61	+12	-2.47
0.5	0.05	149	+55	**+80**	+7	-3	-7	+0.55
0.5	0.1	147	+54	+79	+9	-2	-9	**+0.66**
0.5	1.0	156	+55	+72	-10	+53	+10	-2.12
0.8	1.0	111	+60	+73	-7	+42	+7	-1.63
1.0	1.0	94	+47	+60	-12	+39	+12	-1.88
0.0	1000.0	267	0	0	-126	+972	+126	-25.59
1000.0	0.0	0	-6	+27	0	0	0	0.00

Table 7.6: Results of dual decomposition with the unrealistic coordination analyzer whose both training and test data are January 10th.

one of the unrealistic settings shown in Table 7.1: both the training and the test data are January 10th. The accuracy without dual decomposition is much higher than the setting of previous experiments but is far from 100%. Therefore, we regard the coordination analyzer's output with the setting as the output of a more accurate coordination analyzer virtually.

Results are shown in Table 7.6. Compared to Tables 7.2 and 7.3, improvements in dependency accuracies doubled and in F-measures of coordinated structures are decreased. The results with the unlabeled parser shows that their labeled dependency accuracies are comparable to those of the one-versus-rest dependency labeler. These results imply that dual decomposition desires a more accurate coordination analyzer and another common representation which models higher level of interactions.

7.6 Summary

One of the difficult problems for dependency parsers is to identify spans of coordinated structures. A possible solution to the problem is to employ a coordination analyzer, which is specialized for capturing such structures; a sentence is analyzed by a dependency parser and a coordination analyzer independently.

We integrated a dependency parser's and a coordination analyzer's output by using a dual decomposition algorithm to improve accuracies of the two analyzers each other. The algorithm employs two analyzers and iteratively updates the penalty variables based on the difference of the two analyzer's current outputs until their outputs agree. For example, if an analyzer outputs a certain dependency relation and the other analyzer does not, the dependency relation will be less preferable to the parser and more preferable to the other parser in the next iteration than in the current iteration. Since the algorithm requires certain sort of analyzer whose optimal solution is the maximally-scored one, we first developed a score-based round-robin model, a labeled parsing model which inherits the advantage of the tournament model. Scores of dependency labels are estimated a one-versus-rest algorithm with SVMs. Then, we define a common representation for the two analyzers, which enables the algorithm to compare and calculate the difference of the two analyzer's outputs. In the representation, an output of a coordination analyzer is regarded as a partial dependency tree which consists of parallel (label-P) dependency relations connecting the rightmost bunsetsus of neighboring conjuncts of a coordinated structure.

The dual decomposition algorithm we defined for the integration is not straightforward compared to Koo's [23] dual decomposition algorithm for multilingual dependency parsing. Since the internal structure of the coordination analyzer cannot represent normal (label-D) dependency relations, the algorithm cannot penalize a dependency relation with the label D. Instead, the algorithm promotes all the label-P dependency relations whose dependent is same as the label-D relation because they are disjoint each other.

We conducted several experiments with Kyoto Text Corpus. We first investigated the accuracy of the coordination analyzer without dual decomposition and show that its accuracy is low; identification F-measure of scopes of coordinations is below 35%, and that of ends of conjuncts, the looser evaluation measure, is below 50%. Then, we conducted the experiment with the unlabeled round-robin model, whose dependency labels are initialized as D. With greedy tuning of the parameters of the dual decomposition algorithm, unlabeled dependency accuracy improved by 0.23%, labeled accuracy improved by 1.91%, and F-measure of scopes of coordinations improved by 0.48%. However, the labeled accuracy is much lower than the one-versus-rest labeler. We also conducted experiments with the labeled round-robin model. Unlabeled dependency accuracy improved by 0.25%, labeled accuracy improved by 0.20%, and F-measure of scopes of coordinations improved by 1.11%. The dual decomposition algorithm slightly improves both parsing accuracy and coordination analysis performance.

We conducted further investigations. The experiment with optimal number of maximum iterations as the oracle show that the upper bound of improvements in both unlabeled and labeled dependency accuracies are around 1%. The experiment with another test data show that the dual decomposition algorithm brings less improvement than that with the original test data. In the final experiment, we employed the unrealistic coordination analyzer whose training and test data are identical. Its F-measure of scopes of coordinations is around 60%. With the analyzer, improvements in both unlabeled and labeled dependency accuracies are doubled.

The dual decomposition algorithm can slightly improve both dependency parser and coordination analyzer. However, a more accurate coordination analyzer is requested for a practical level of improvement.

Chapter 8

Use of Coordinated Structure Annotation for Efficient Dependency Structure Annotation

In this chapter, we propose a method for boosting annotation efficiency with introducing a new annotation criteria which resolves several problems in the Kyoto Text Corpus annotation criteria. We have been participating the project to construct the Balanced Corpus of Contemporary Written Japanese (BCCWJ). Our mission is to annotate dependency structures and coordinated structures on the corpus. Kyoto Text Corpus encodes coordinated structures in dependency structures by using certain dependency labels. In contrast, we have been annotating dependency structures and coordinated structures separately.

The separate annotation of dependency structures and coordinated structures of BCCWJ enables to describe more flexible structures. It also enables to prohibit introducing unnatural dependency relations even for a certain sort of complex coordinated structures. A coordination is a sort of ellipsis occurred by human for reducing the number of words which the person has to write or speak. A coordinated structure is sometimes quite complex to compress the original sentence significantly. Although a coordinated structure is embedded in a dependency tree in Kyoto Text Corpus, it cannot represent complex coordinated structures because of the constraints on a dependency tree. The embedded representation also requests to introduce unnatural dependency relations for a certain sort of structures. The separate annotation resolves several such problems.

When constructing a corpus with dependency and coordinated structures, it is effi-

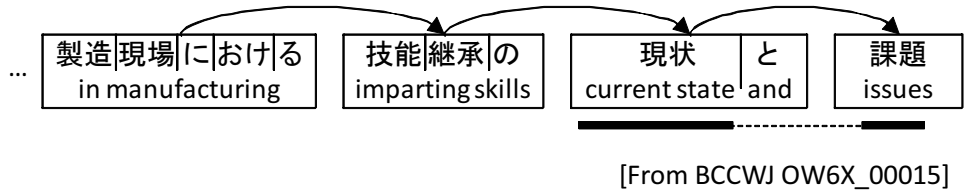


Figure 8.1: Notation of dependency relations and a coordinated structure in this chapter. “Current state and issues of imparting skills in manufacturing.”

cient to annotate coordinated structures first. This is because definition of coordinated structures is more understandable than that of dependency structures, and automatic coordination analysis is less accurate than dependency parsing. Dependency structure annotators work with the merged structure of dependency parser’s output and coordinated structure annotation. Such sequential annotation keeps consistency between dependency structure annotation and coordinated structure annotation. This annotation order also enables annotators to grasp the structure of a long sentence instantly by the given coordinated structure annotation.

The coordinated structure annotation is not fully exploited. In this chapter, we construct a more precise dependency parser using the constraints derived from the annotation. The constraints also detect dependency annotation errors.

8.1 Construction of the BCCWJ Corpus

Since the constraints have a high correlation with BCCWJ annotation criteria, this section illustrates the annotation steps and the criteria for constructing the corpus.

8.1.1 Annotation Steps

Prior to annotating coordinated and dependency structures, National Institute for Japanese Language and Linguistics took samples from texts such as books or magazines, typed them into computer, split the electronic data into sentences, assigned morphological information to them and then split them into bunsetsus, sequentially.

A single annotator assigned coordinated structures on the preprocessed data. The beginning point and the ending point of a coordinated structure must be a word bound-

ary[1]. We illustrate a coordinated structure as shown in Figure 8.1. A rectangle is a bunsetsu. A word[2] boundary is represented as a vertical line from the top to middle of a rectangle. A coordinated structure is represented as bold underlines and dotted underlines. A bold underline is the span of an element of a coordinated structure, which is referred to as a *conjunct*. Dotted underlines group the conjuncts into a coordinated structure. The annotator assigned spans of conjuncts and their groupings (Step A). Coordinated structures were assigned to all the sentences of BCCWJ core data, which consists of almost 1 million words. Apposition structures were also assigned in the same manner.

In the next step, we assigned dependency structures by using a dependency parser, and then annotators correct errors in the dependency parser's output. We overwrote the morphological information of Kyoto Text Corpus with the Short Unit Words by using MeCab[3] with UniDic [7] morphological dictionary. A dependency parser using the tournament model was trained with the modified Kyoto Text Corpus. The coordinated structure annotation (output of Step A) was merged with the parser's output. Total of 11 annotators are correcting parser's errors in parallel (Step B). As we will describe in Section 8.1.2, dependency structure annotation criteria of BCCWJ differ from that of Kyoto Text Corpus. Therefore, the difference will cause parsing errors. For 80% of BCCWJ core data, Step B was completed.

Finally, one or two expert annotators will correct both coordinated and dependency structures annotations (Step C). This step will ensure the consistency of both of the annotations.

The research described in this chapter corresponds to Step B.

8.1.2 Annotation Criteria of Coordinated and Dependency Structures

This subsection illustrates the differences in annotation criteria between Kyoto Text Corpus and BCCWJ. One of the major differences is the representation of an incomplete coordinated structure. BCCWJ represents it not by using certain dependency labels but as the span of coordinated structure and dependency relations which modify

[1] More specifically, the points must be a boundary of the Short Unit Word defined by National Institute for Japanese Language and Linguistics.

[2] Every word is the Short Unit Word.

[3] `http://mecab.sourceforge.net/`

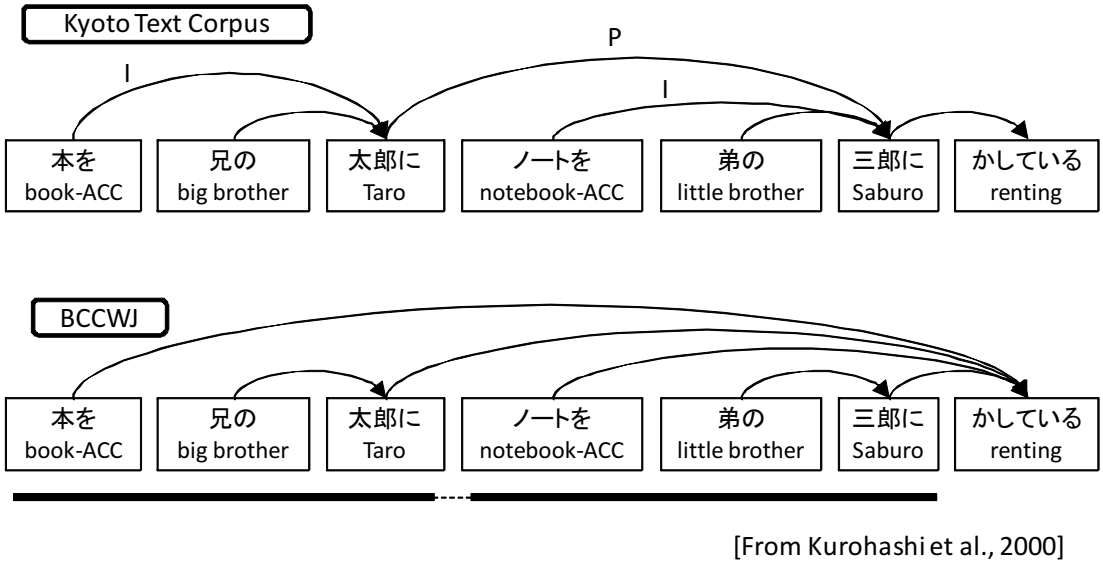


Figure 8.2: Annotation for an incomplete coordinated structure on Kyoto Text Corpus and BCCWJ. Dependency label D is omitted in the figures in this chapter. "(I am) renting a book to (my) big brother, Taro, and a notebook to (my) little brother, Saburo."

their true heads. Note that verb phrase coordinations, which are a kind of coordinations in Kyoto Text Corpus, are regarded as not coordinations but just dependency relations between verbs.

Incomplete Coordinated Structures

In BCCWJ, incomplete coordinated structures are represented by unlabeled dependency relations and spans of coordinated structures. Dependency relations in Kyoto Text Corpus are labeled dependency relations which enable marking coordinated structures and appositions; label P represents parallel relation; I is for relations inside of an incomplete coordinated structure; A represents apposition relation; D represents normal dependency relation. Unlike Kyoto Text Corpus, the separate annotation of dependency and coordinated structures obsoleted dependency labels.

In Kyoto Text Corpus, some of the dependency relations inside of an incomplete coordinated structure does not modify their true heads. As shown in Figure 8.2, Kyoto Text Corpus annotation criteria prohibit such dependency relations from going outside of their conjuncts. For example, despite the true head of "book" is "renting", "book"

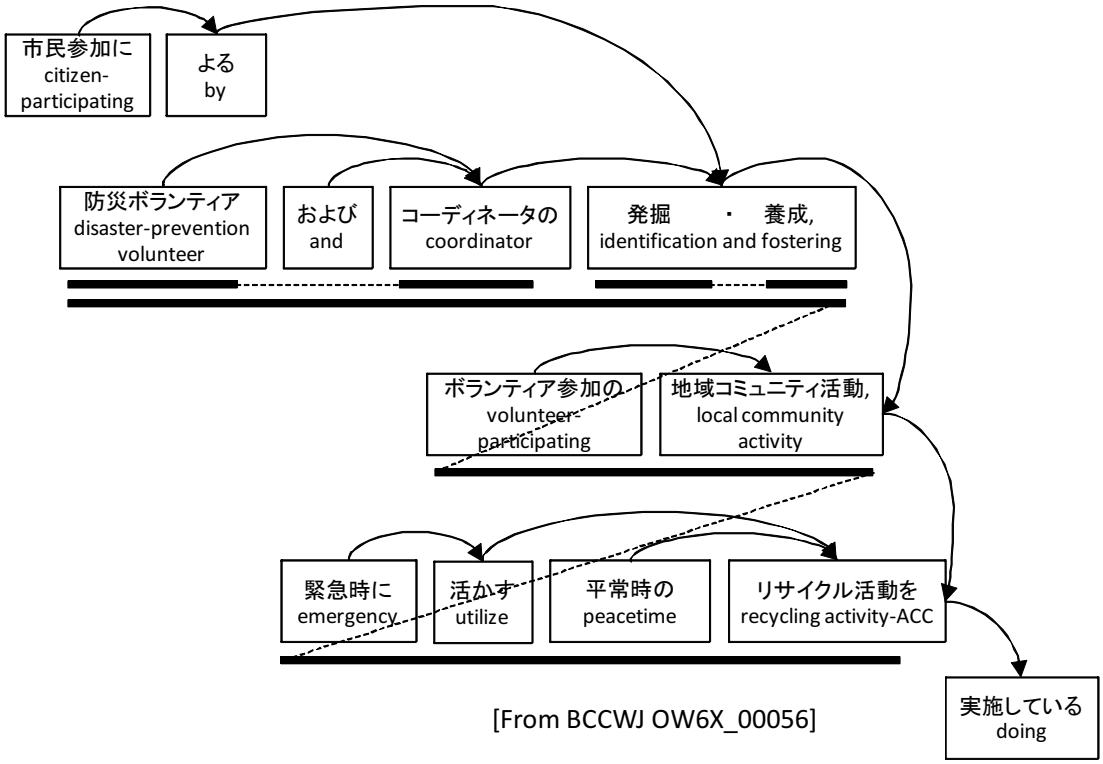


Figure 8.3: A complex coordinated structure related to various annotation rules. "(The organization is) doing identification and fostering of disaster-prevention volunteers and coordinators, volunteer-participating local community activities, and recycling activities in peacetime to utilize (them) in emergencies, where citizens participate. "

modifies "Taro" with dependency label I. This rule forces projectivity to dependency structures by avoiding crossings of dependency relations.

Since dependency labels were obsoleted in BCCWJ, dependency relations may cross each other. BCCWJ allows non-projective dependency structures.

Coordinated Structures with Three or More Conjuncts

Rightmost bunsetsus of each of conjuncts must modify the rightmost bunsetsu of their neighboring conjunct. The example sentence shown in Figure 8.3 contains a coordinated structure with three conjuncts: "identification and fostering of disaster-prevention volunteers and coordinators", "volunteer-participating local community activities" and "recycling activities in peacetime to utilize them in emergencies". "iden-

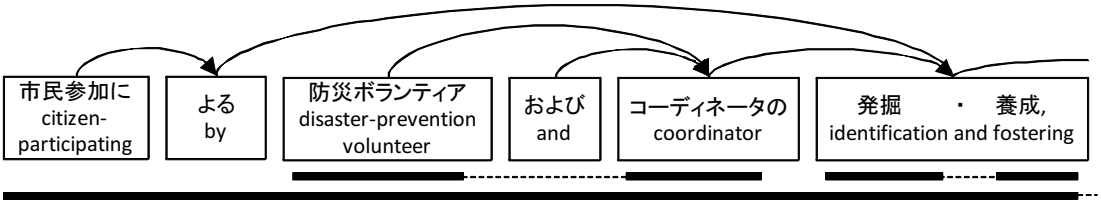


Figure 8.4: Another coordinated structure annotation on the sentence of Figure 8.3.

tification and fostering" modifies "local community activities". Similarly, "local community activities" modifies "recycling activities".

Connective Expressions

Typical connective expressions are coordinate conjunctions which correspond to "and" or "or" of English. A connective expression is regarded not as an element of a conjunct but as an inter-conjunct element. A connective expression must modify the rightmost bunsetsu of its neighboring conjunct. For example, "および"(and) in Figure 8.3 modifies "coordinator".

Bunsetsus Which Semantically Modify All the Conjuncts

A dependency relation from outside to inside of a coordinated structure does not indicate single dependency relation, semantically. Such a dependency relation is regarded as relations to all the corresponding bunsetsus of every conjuncts. Such dependency relation must modify the corresponding bunsetsu of the leftmost conjunct. For example, "by" in Figure 8.3 modifies "identification and fostering". This annotation implies that the dependent "by (citizen-participating)" also semantically modifies "local community activity" and "recycling activity".

If, as shown in Figure 8.4, "by (citizen-participating)" is included in the leftmost conjunct, the semantic meaning which the annotation represents will change. "by (citizen-participating)" semantically modifies only "identification and fostering". The annotation does not mean neither "citizen-participating local community activity" nor "citizen-participating recycling activity". Although the dependency structures of Figure 8.3 and Figure 8.4 are identical, the meaning differs because of the scopes of the coordinated structure annotations.

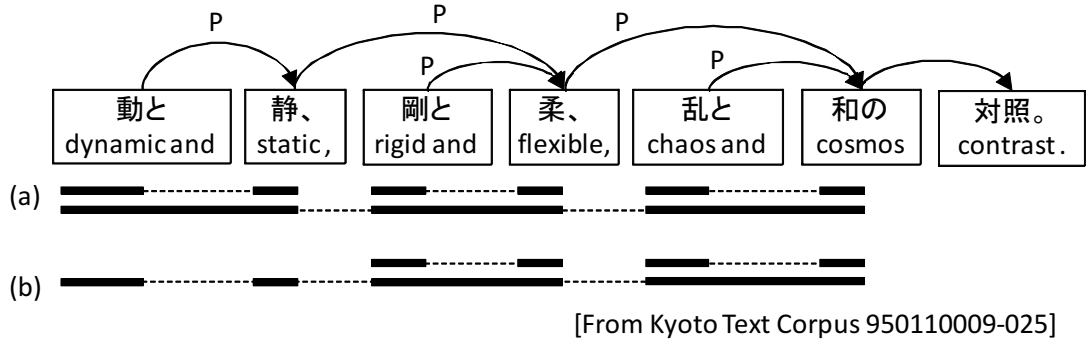


Figure 8.5: Two interpretations for nested coordinated structures on Kyoto Text Corpus. “Contrast of dynamic and static, rigid and flexible, and chaos and cosmos.”

Nested Coordinated Structures

Coordinated structures can be nested. In Figure 8.3, the leftmost conjunct of the outer 3-conjunct coordinated structure contains another two coordinated structures: “disaster-prevention volunteers and coordinators” and “identification and fostering”.

Kyoto Text Corpus also allows nested coordinated structures. However, two or more interpretations may be possible for such structures. The sentence shown in Figure 8.5 includes such a structure. Interpretation (a) is probably correct, but interpretation (b) is also possible.

BCCWJ does not inherit the problem because every coordinated structure of nested coordinated structures is annotated independently.

8.1.3 Inconsistencies

The final correction (Step C) has not finished yet. We know the following problematic and inconsistent annotations corresponding to coordinated structures.

1. A bunsetsu belongs to two or more conjuncts of a coordinated structure. An examples is the annotation shown in Figure 8.6 (a). The bunsetsu “強化・継承の”(reinforcement and imparting) is shared by two conjuncts “力の強化”(reinforcement of skills) and “継承”(imparting). Major reason of the problem is probably annotation errors or file operation errors when merging annotations.

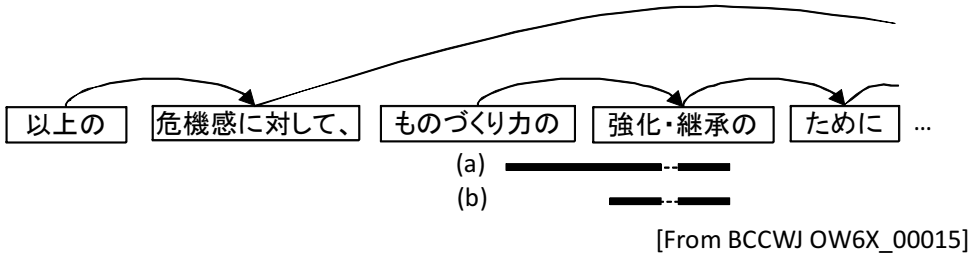


Figure 8.6: Annotation error in spans of conjuncts. The coordinated structure is currently annotated as (a), but (b) is probably correct.

An exception of this pattern is a coordinated structure which consists of single bunsetsu such as "identification and fostering". We does not regard them as problematic ones.

2. Violation of the rule described in the section **Coordinated Structures with Three or More Conjuncts**; rightmost bunsetsu of each of conjuncts must modify its neighboring conjunct. However, it modifies the rightmost conjunct. For example, if "identification and fostering" modifies "recycling activity", the dependency relation is an instance of this sort of violation.

3. Violation of the rule described in the section **Bunsetsus Which Semantically Modify All the Conjuncts**; dependency relations from outside to inside of a coordinated structure must modify the leftmost conjunct. However, it modifies the rightmost conjunct. For example, if "by" modifies "recycling activity", the dependency relation is an instance of this sort of violation.

8.2 Proposed Method: Constraining Possible Dependency Structures

Typical Japanese dependency parsers regard all the bunsetsus that are located on the right of the dependent as candidate heads.[4] Unlike such parsers, we constrain the sets of possible candidate heads by using the coordinated structure annotations.

[4]The PROJECTIVE constraint is usually employed due to the reasons such as characteristics of a parsing algorithm or a corpus.

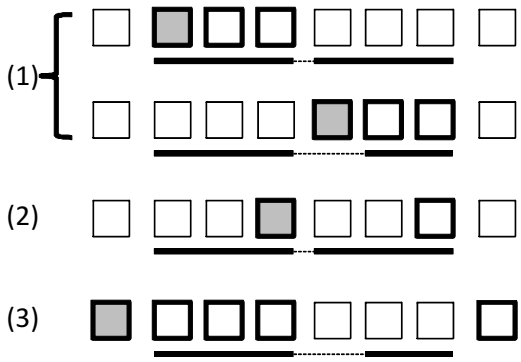


Figure 8.7: Constraints which we can derive from coordinated structure annotations. A gray box is the dependent bunsetsu. Thick boxes are its possible heads. Thin boxes are not candidate heads of the dependent.

We excluded certain coordinated structures from experiments. Since a dependency relation is a relation between bunsetsus, we ignored the coordinated structures which consist of a single bunsetsu. We also ignored the problematic structures corresponding to the first pattern of Section 8.1.3.

Constraints Derived from Coordinated Structure Annotations

As shown in Figure 8.7, a set of possible candidate heads is constrained as follows.

Constraint 1

- A bunsetsu inside of a conjunct, except for the rightmost bunsetsu, modifies any bunsetsu in the conjunct. For example, all the possible heads of “emergency” in Figure 8.3 are “utilize”, “peacetime” and “recycling activity”.
- A connective expression modifies its neighboring conjunct.

Constraint 2

Rightmost bunsetsus of conjuncts, except for the rightmost conjunct, modify the rightmost bunsetsu of their neighboring conjunct. For example, the only possible head of “identification and fostering” is “local community activity”.

Constraint 3

- A dependency relation from outside to inside of a coordinated structure modifies the leftmost conjunct. For example, all the possible heads of "by" are "disaster-prevention volunteer", "and", "coordinator", "identification and fostering" and "doing". It can modify neither "local community activity" nor "recycling activity".
- It is prohibited to modify from outside of a coordinated structure to a connective expression.

For convenience of experiments, we automatically corrected all the instances which violate Constraint 2.[5] We cannot automatically correct the violations of Constraint 3 because it is not trivial that how to correct them, even if we can detect the violation.[6]

These constraints are not appropriate for incomplete coordinated structures. Since there are few incomplete coordinated structures, overall improvement is expected to be very limited even if we consider them. Moreover, parsing algorithms for such structures are expected to be complex.

8.3 Experiments

The sub-corpora of BCCWJ we used in the experiments are white paper (OW, for short), *Yahoo! Chiebukuro* Q&A (OC), and newspaper (PN). We used the corpora of as of July 22nd, 2011. Since annotation (Step B) on OC and PN have not been completed, we used only the completed parts of these corpora. We corrected a few critical errors such as in bunsetsu indices that crash the parser. Then, we split each of the corpora into training and test data as shown in Table 8.1. Note that the numbers in the table include sentences whose lengths are less than 3. Dependency structures of such sentences are uniquely determined. Such sentences are never used for parser training.

We employed the dependency parser using the tournament model with the *opal* binary classifier. Experiments were conducted with the second degree and the third degree polynomial kernels. The other options for opal is identical to that used in Chapter

[5] As we will describe below, there are a small number of irregular coordinated structures. This automatic correction is inappropriate for such structures.

[6] Automatic correction for such a instance is probably possible if its head is the rightmost bunsetsu of a conjunct.

Table 8.1: Corpus split. Sentence length is in the number of bunsetsus. coords: coordinated structures.

	Training data [sent.]	Test data [sent.]	# Total bunsetsus	# Total coords	Avg. sent. length	Avg # coords
OW	4500	1326	70126	3901	12.04	0.67
OC	5000	1613	36811	786	5.57	0.12
PN	1500	445	14560	528	7.49	0.27

Table 8.2: Unlabeled dependency accuracies of experiments with and without the constraints. The braced numbers are improvements compared to the experiment with the second degree polynomial kernel.

	Degree of	Constraints			
Corpus	kernel	nothing	1 and 2	1,2 and 3	1, 2 and 3R
OW	2	85.08	88.65 (+3.57)	**88.74 (+3.66)**	87.55 (+2.47)
	3	84.70	88.19 (+3.11)	88.33 (+3.25)	87.10 (+2.02)
OC	2	87.69	89.25 (+1.56)	89.21 (+1.52)	**89.87 (+2.18)**
	3	86.99	88.56 (+0.87)	88.57 (+0.88)	89.22 (+1.52)
PN	2	84.08	86.96 (+2.88)	87.05 (+2.97)	**87.83 (+3.75)**
	3	83.93	86.82 (+2.74)	86.88 (+2.80)	87.63 (+3.55)

7. Training procedure and accuracy calculation criteria are same as described in Section 4.4.1. That is, we never restricted the sets of candidate heads in training. In experiments in this section, the parser performs non-projective parsing.

8.3.1 Automatic Dependency Parsing Accuracies

Since the corpora contain violations of Constraint 3, four experiments were performed: no constraints, Constraints 1 and 2 only, all the constraints, and Constraints 1, 2 and 3R. We define another constraint 3R for convenience. Constraint 3R is identical to Constraint 3 except that a dependency relation mentioned in the constraint must modify the *rightmost* conjunct.

Table 8.2 shows that all the constraints contribute to improve parsing accuracy. The great improvement in OW is understandable because OW contains a lot of coordinated

Table 8.3: Unlabeled dependency accuracies among sentences containing one or more coordinated structures.

	Degree of	Constraints			
Corpus	kernel	nothing	1 and 2	1,2 and 3	1, 2 and 3R
OW	2	83.31	88.72 (+5.41)	**88.85 (+5.54)**	87.06 (+3.75)
	3	83.22	88.50 (+5.19)	88.71 (+5.40)	86.84 (+3.53)
OC	2	80.41	88.27 (+7.86)	88.06 (+7.65)	91.41 (+11.00)
	3	80.26	88.20 (+7.79)	88.27 (+7.86)	**91.55 (+11.14)**
PN	2	79.37	86.07 (+6.70)	86.27 (+6.90)	**88.08 (+8.71)**
	3	79.50	86.20 (+6.83)	86.34 (+6.97)	**88.08 (+8.71)**

structures as shown in Table 8.1.

The result also shows that the second degree polynomial kernel is more appropriate than the third degree one. The reason is probably inconsistencies in the annotations. The third degree kernel learns more specific context than the second degree one. However, the learned specific context worked negatively due to the inconsistencies.

Since our method can benefit only for sentences containing coordinated structures, the accuracies calculated among such sentences are shown in Table 8.3. The accuracy improvements among such sentences are 6-11%. The shorter the average sentence length, the greater the improvement. Especially for OC, our method reduces more than half of the parsing errors. The relative error reductions of OW, OC and PN are 33%, 57% and 42%, respectively. That is, the number of parsing errors becomes half or one-third by incorporating the constraints. Therefore, human effort to correct parsing error is significantly reduced.

Table 8.4 shows the numbers of dependency relations which cannot modify their correct heads due to the constraints. In other words, the gold-standard head of such a dependency relation is not an element of the set of its possible candidate heads. Their reasons are incomplete coordinated structures, connective expressions and annotation errors. An incomplete coordinated structure contains dependency relations from inside of conjuncts to outside of the coordinated structure. Such dependency relations are prohibited by Constraints 1 and 2. These are all the patterns which the proposed method makes mistakes. In other words, our method efficiently detects annotation errors by checking instances corresponding to Table 8.4. If such an instance corre-

Table 8.4: The numbers of dependency relations which cannot modify their gold-standard heads due to the constraints.

	Constraints			
Corpus	nothing	1 and 2	1,2 and 3	1, 2 and 3R
OW	(0)	12	23	342
OC	(0)	0	43	1
PN	(0)	3	42	21

sponds to neither an incomplete coordinated structure nor a connective expression, the instance is an annotation error. Table 8.4 also implies a lot of inconsistent instances regarding Constraint 3. That is, the annotators probably mistakenly understand that not Constraint 3 but Constraint 3R is the correct annotation criterion.

8.3.2 Example Sentences

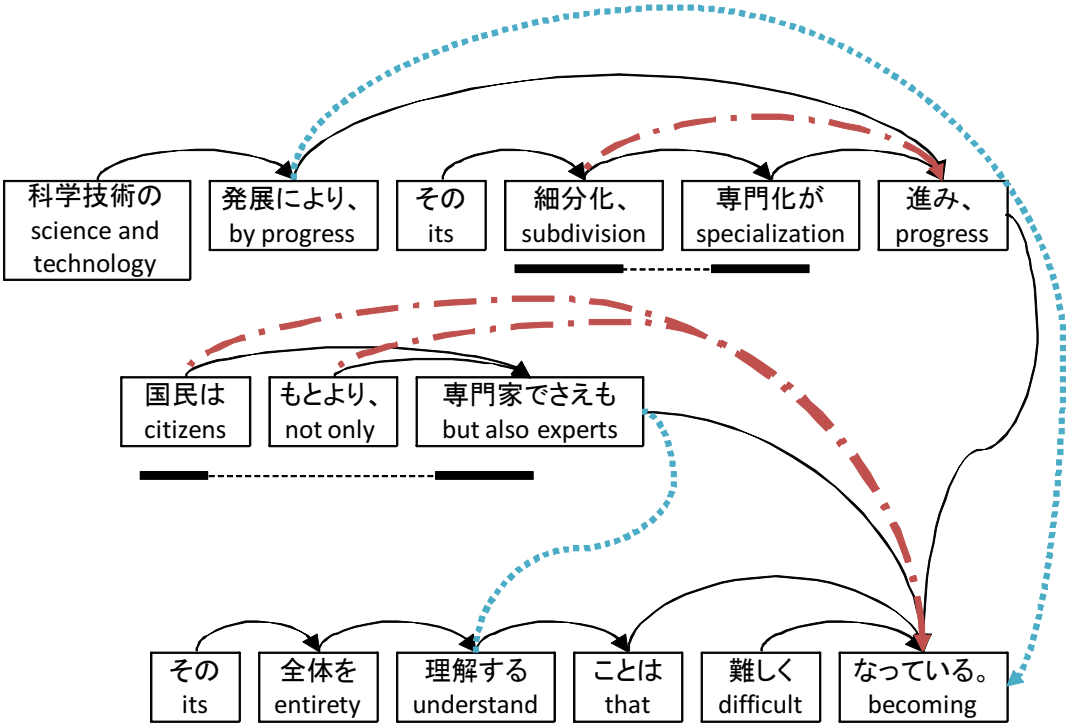
The example sentence shown in Figure 8.8 contains three corrected dependency relations by the constraints. Constraint 2 uniquely determined the heads of "subdivision" and "citizens". Constraint 1 also uniquely determined the head of "but also experts" since it is a conjunct which consists of a single bunsetsu.

Another example sentence is shown in Figure 8.9. A dependency relation was corrected because Constraint 3 prohibits "outreach activity" from modifing "not only". On the other hand, the head of a connective expression "not only" became incorrect due to Constraint 1.[7]

Figure 8.10 is a sentence containing an incomplete coordinated structure. The appropriate head of all the bunsetsus is "adopted". However, several mistakes were made because several bunsetsus are prohibited from modifying "adopted" by Constraint 1.

Note that the automatic correction of violations of Constraint 2 made inappropriate 'correction'. The gold-standard head of "in United Nations Third Committee" is "in United Nations General Assembly plenary session" and its appropriate head is

[7]It is an exceptional structure. We decided that "science and technology"(科学技術に) and "not only"(限らず、) are included in the coordinated structure. This is because we interpreted "限らず" not as "not limited to" but as "not only (but also)", which is a kind of coordinate conjunction of English. However, there may be readers who disagree with the decision.



[From BCCWJ OW6X_00088]

Figure 8.8: An example sentence which contains corrected dependency relations by the constraints. Black solid arcs point their gold-standard heads. Red dashed-dotted arcs points original heads of corrected dependency relations. Blue dotted arcs are parsing errors which were not corrected. "By progress of science and technology, its subdivision and specialization progress; it is becoming difficult for not only citizens but also experts to understand its entirety."

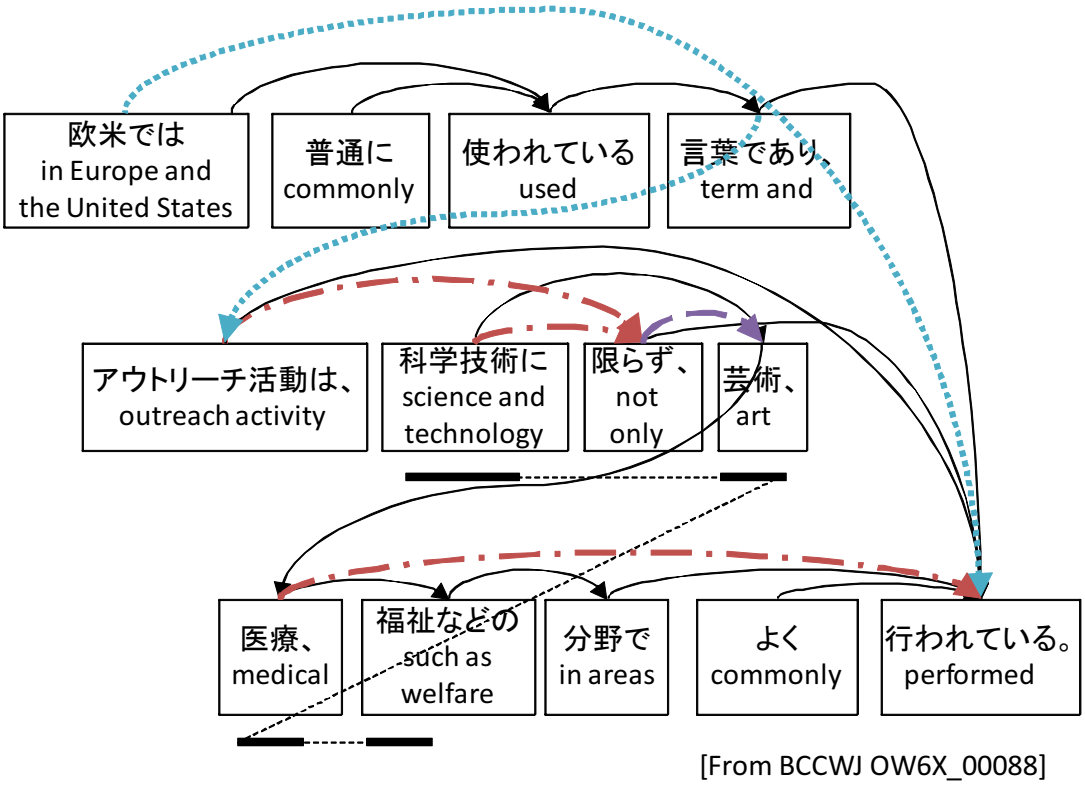
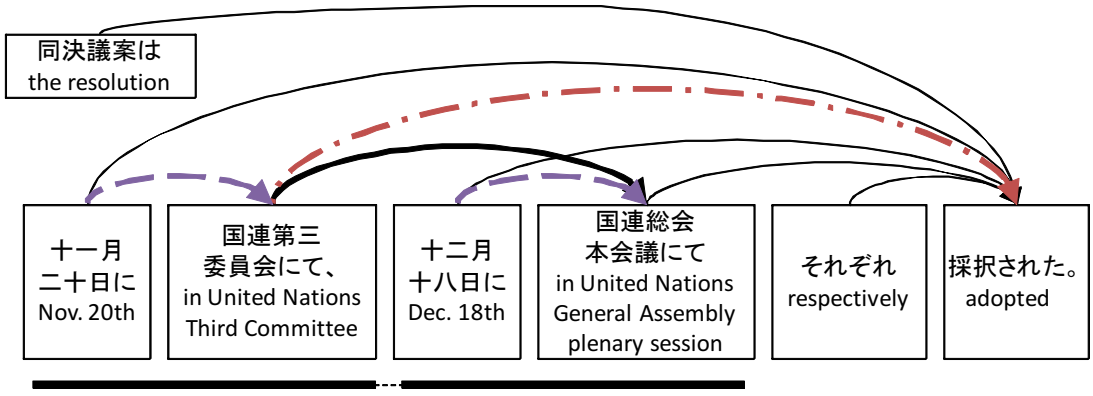


Figure 8.9: Another example sentence corrected by the constraints. Purple dashed arcs are dependency relations which became incorrect due to the constraints. “Outreach activity is a commonly-used term in Europe and the United States; outreach activities are commonly performed in areas not only science and technology but also art, medical, welfare and so on.”



[From BCCWJ OW6X_00168]

Figure 8.10: An example sentence which made mistakes due to the constraints: incomplete coordinated structure. "The resolution was adopted in United Nations Third Committee on November 20th and in United Nations General Assembly plenary session on December 18th, respectively."

"adopted". The 'correction' of the head of "science and technology" in Figure 8.9 from "not only" to "art" is also inappropriate.[8]

The automatic correction is just a preprocessing for convenience of evaluating parsing accuracies with less consistent corpora. It will not be performed in practice. The steps of annotation using our method will be as follows.

1. Manual coordinated structure annotation (Step A in Section 8.1.1)
2. Automatic precise dependency parsing using the constraints derived from the coordinated structure annotations
3. Manual correction (Step B)
4. Manual check of dependency relations which cannot modify their correct heads due to the constraints (Table 8.4)
5. Final manual correction (Step C)

[8]We evaluate whether a dependency relation is correct or not by comparing with its gold-standard head, which is the head after the automatic correction. Therefore, it is interpreted as 'made mistake' (red dashed-doted line) if the head of "in United Nations Third Committee" is "adopted" without the constraints and "in United Nations General Assembly plenary session" with the constraints.

Our method improves parsing accuracy in Step 2 and helps to find annotation errors in Step 4.

8.4 Discussion about Ideal Dependency Structure Annotation Criteria

Although we just illustrated and used the annotation criteria of BCCWJ in this chapter, what is the ideal dependency structure annotation? Since a coordinated structure is a special sort of dependency structure, the discussion about ideal dependency structure is almost independent from whether a dependency structure contains coordinated structures or not.

Appropriateness of a particular set of annotation criteria can be discussed from several viewpoints: legitimacy of the criteria, difficulty of parsing, and convenience for applications of parser's output.

With Preserving Constraints of Well-formed Dependency Structure

Surprisingly, legitimacy and difficulty of parsing are strongly correlated each other in statistical approaches especially those with very large-scale corpora; a frequently-observed structure is a plausible structure for human, and therefore the structure is a *correct* structure. For example, Mochihashi et al. [39] demonstrated a fully unsupervised word segmentation with a Japanese classic, The Tale of Genji, and showed the result is a *plausible* segmentation. The case frame dictionary which KNP employs is also a kind of frequency-based dictionary for evaluating plausibility of a predicate-argument structure. A human probably feels that the plausibility of an output of such an analyzer is higher than its accuracy calculated with the gold-standard. Since a fully unsupervised analyzer cannot know the annotation criteria of a test data, its accuracy tend to be underestimated. For example, in contrast to a parsing error of the tournament model, that of KNP tends to raise sympathy for human in the sense that the latter is more plausible.

Acceptability for a particular structure is different for different person because of reasons such as his/her belonging communities. If a set of annotation criteria or a grammar formalism is familiar to a person, s/he may feel that an output of an unsu-

pervised analyzer is inappropriate. A *grammar* in an unsupervised parser may be just clusters of sentences.

A structure similar to a familiar structure is also plausible for a human. A new sentence or phrase is constructed by association. A person can reuse a sentence when s/he constructs another sentence by replacing content words in the original sentence with other words. Similarly, an idiom can be reused when s/he constructs a sentence in another language by translating the words in the original language to corresponding target language words. For example, an English idiom "not only but also" is reused in the Japanese sentences shown in Figures 8.8 (もとより、さえも) and 8.9 (限らず). Since the idiom is a marker of a coordination, we annotated those structures as coordinated structures. However, not everyone agrees with the decisions that those are coordinated structures. Especially, since the latter Japanese translation, "限らず", is a verb, the bunsetsu should modify another verb if its annotation criteria does not make any exception for idioms.

Plausibility of a dependency relation is strongly correlated with how many times its dependent and head are observed as a substring of a sentence. Pitler et al. [50] proposed to exploit a web-scale n-grams into English base noun phrase (NP) parsing. Their parser identifies internal structure of a noun compound such as *retired (science teacher)* and *(social science) teacher*. Such a bracketed representation can be converted into a dependency tree with certain head rules. The parser incorporates features for an SVM based on pointwise mutual information (PMI) calculated with the N-gram corpus [33] enhanced from Google N-grams Version 1, which is created from 1 trillion words:

$$PMI(x,y) = \log \frac{p(\text{“}x\ y\text{''})}{p(\text{“}x\text{''})p(\text{“}y\text{''})}$$

where $p(\text{“}x\ y\text{''})$ is the number of occurrences of the two words x and y contiguously. Similarly, they define PMI for coordinations:

$$PMI_{and}(x,y) = \log \frac{p(\text{“}x\ and\ y\text{''})}{p(\text{“}x\ and\text{''})p(\text{“}and\ y\text{''})}.$$

Their experiments show that the larger the number of unique n-grams, the higher the base NP parsing accuracy. While a base NP is a *contiguous* substring of a sentence, it is an example that a dependency relation can be identified based on its frequency.

Since most of constituents can be ellipted in a Japanese sentence, definition of Japanese dependency relation seems that the string consisting of its dependent and head is accepted as a sentence. Note that functional expressions should be chunked

appropriately before evaluation of acceptability of a particular bunsetsu pair. Not all dependents modify their neighboring bunsetsus. However, a pair of two bunsetsus constituting a dependency relation is observed contiguously when there are essentially no modifiers between the dependent and the head, or modifiers are ellipted. The hypothesis is supported by the fact that about 60% of bunsetsus modify their neighboring bunsetsus. Therefore, Japanese dependency parsing can be performed with n-grams such as Google Japanese N-gram[9]. However, since the *n* of the n-gram is maximally 7 words, coverage of a string consisting of a dependent and a candidate head is not sufficiently high. To achieve a practical accuracy, appropriate smoothing method should be employed for resolving data sparseness.

The policy that frequent structures are correct structures also suitable for constructing a language resource from a huge amount of parsed sentences. Correct structures are expected to frequently occur. Dependency relations corresponding to parsing errors tend to be not plausible and therefore they do not frequently occur. Negative effect caused by parsing errors decreases according to increasing the amount of corpus. However, such negative effect can be significantly decrease by employing heuristics such as using only short sentences, in which a parser hardly make an error.

According to the policy, dependency relations in a coordinated structure should be established between bunsetsus which are expected to frequently occur contiguously. Kawahara and Kurohashi [18] proposed to resolve ambiguity of coordinated structures not by using similarity among conjuncts but as a set of dependency relations. They shows that the case frame of KNP can also disambiguate coordinated structures with a high accuracy. In Figure 8.9, it is inappropriate that “限らず” modifies “行われている”(performed). The form of string “限らず verb” hardly occurs. The fact that the form of string “限らず noun” frequently occurs recommends “芸術”(art) as its head. The dependency relation satisfies the constraints defined in Section 8.2. With respect to Constraints 2 and 3, modifying the rightmost conjunct is slightly more appropriate than modifying the leftmost conjunct. The string corresponding to the latter occurs in coordinated structures consisting of more conjuncts than that corresponding to the former, and therefore the former tend to be more frequent.

[9] http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2009T08

Without Constraints of Well-formed Dependency Structure

From the viewpoint of convenience for applications of parser's output, appropriateness depends on each application. Dependency structure is a commonly-used intermediate representation of a sentence which is not the best but a good representation for wide-range of applications. The best representations depend on applications. A sort of applications such as corpus-based linguistics research and proofreading requires a representation preserving the original sentence. In contrast, another sort of applications such as information extraction intends to extract a certain semantic representation of a sentence. The latter may not require a dependency structure, or requires to annotate relations between much more pairs of bunsetsus such as Kuroda and Iida [28] did, without preserving the constraints to be a tree. The structure they proposed does not satisfy SINGLEHEAD constraint. A sentence containing coordinated structures should be decomposed into sentences without coordinated structures. To annotate information for such an automatic decomposer as dependency relations, the dependency structure usually violates SINGLEHEAD constraint.

8.5 Summary

We developed the methods for exploiting coordinated structure annotation for efficient annotation on a corpus which dependency structures are coordinated structures are annotated separately such as the BCCWJ corpus. We described the differences in annotation criteria for Kyoto Text Corpus and BCCWJ and benefits by separating dependency and coordinated structure annotations. The BCCWJ annotation criteria represents an incomplete coordinated structure without introducing unnatural dependency relations. The criteria also resolves the interpretation ambiguity in nested coordinated structures. We defined the constraints derived from a coordinated structure annotation and restrict possible dependency structure for a dependency parser.

The constrained parser improves accuracy by 2-4% with three sub-corpora of BCCWJ compared to the parser without the constraints. Our method can benefit only for sentences containing coordinated structures. The accuracy improvements among such sentences are 6-11%. The improvements are relative error reductions of 33-57%. That is, the number of parsing errors becomes half or one-third by incorporating the constraints. Therefore, human effort to correct parsing error is significantly reduced.

Our method is also applicable to annotation error detection by checking dependency

annotations violating the constraints. Such annotation corresponds to an exceptional coordinated structure or an annotation error.

Chapter 9

Future Work

This chapter describes the problems which should be solved to achieve more accurate parsing and to boost efficiency of corpus annotation.

9.1 Refinement of Annotation Guideline

The BCCWJ annotation guideline, illustrated in Section 8.1.2, fixed several problems in the Kyoto Text Corpus annotation guideline. However, annotation guideline should be refined in several points.

One of the reasons of inconsistent annotations is that some dependency relations are essentially ambiguous. That is, there are two or more interpretations of the sentence, and their difference is subtle or it is impossible to select one interpretation by using only the information in the sentence.[1] In such a case, forcing annotators to select one of the interpretations causes inconsistent annotation. A solution for the problem is to allow ambiguous annotation, that is, a bunsetsu can have two or more heads. Tsuboi et al. [63] proposed a training method with partial or ambiguous annotation for sequence labeling tasks. Even if we employ such a method, no specialized parser will be required. We can convert such a sentence into a well-formed dependency tree by using certain rules. Then, we can train a parser with the converted corpus and parse test sentences by a parsing algorithm designed for well-formed dependency trees.

[1] There are Japanese sentences which lead inconsistent annotations but their interpretations do not depend on other sentences. In such a sentence, it tends to be quite difficult to explain the difference among the interpretations in English; it is a language-dependent ambiguity which is usually related to a functional expression.

Partial annotation is sometimes more preferable than such ambiguous annotations. Introducing ambiguous annotations, that is, removing SINGLEHEAD constraint, is not an essential solution to the ambiguity. An annotator may wonder whether a bunsetsu has two heads or three heads.

Similarly, such ambiguous dependency relations are harmful also in the evaluation of a dependency parser. Some sort of such dependency relations may not be useful for any application. Especially, informal documents such as blog articles seems to contain a lot of such relations. A possible solution is application-specific evaluation. For example, a parser is evaluated only with predicate-argument relations and the other dependency relations are ignored.

Yamamoto and Masuyama [69] defines patterns of parsing errors. Their experiments include the one about whether dependency relation annotations are appropriate. They randomly sampled at most 30 incorrect dependency relations for each of 11 patterns. The result suggests that 19% of annotations corresponding to the sampled 327 dependency relations are wrong, and 8% of annotations are difficult to decide a single most appropriate head for each of them.

Partial annotation solves another problem. The annotation guideline for Kyoto Text Corpus provides a rule; if it is quite ambiguous which bunsetsu is the head of a dependent, the rightmost bunsetsu of the sentence is regarded as its head. "Quite ambiguous" usually means that there is no appropriate head of a bunsetsu in the sentence. This sort of ambiguities tends to occur because the writer ignored grammatical concord of adverbs, especially in long sentences. The annotation rule can be harmful for training because the dependency relations annotated according to the rule are sometimes exceptional. If we allow a dependent to have no head, a dependent of such an exceptional relation will have no head. The solution can be introduced without allowing partial or ambiguous annotations. A possible representation of the phenomena is to prepare artificial ROOT bunsetsu, which is the head of the rightmost bunsetsu. The ROOT is regarded as the head of such ambiguous relations.

9.2 Deriving Finer Constraints from Coordinated Structure Annotation

Finer constraint definition should be introduced. It may be valuable to distinguish exceptional coordinated structures from regular ones. If we can automatically dis-

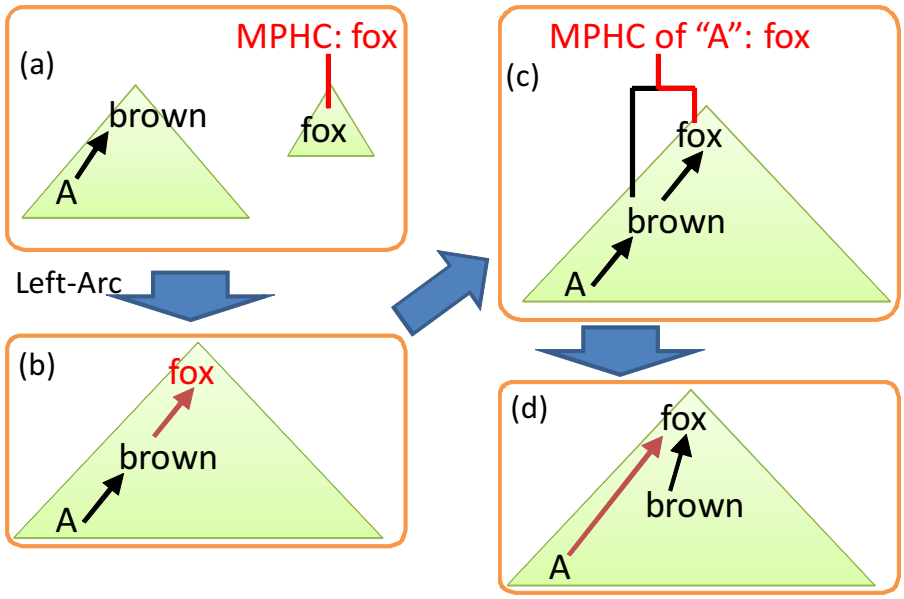


Figure 9.1: Another tree-based model which hold a tournament in both top-down and bottom-up constructions.

tinguish them completely by using clues such as their POS sequences, the effort of manual checking may be reduced. For example, if a coordinated structure consists of single-bunsetsu conjuncts, the dependency structure inside of the structure is uniquely determined by Constraints 1 and 2 of Section 8.2.

9.3 Another Direct Comparison-based Models

We proposed the round robin model in Section 7.2. The parser can be regarded as a first-order MST model. Similarly, we could construct second-order or third-order MST models. An effective higher-order feature may be the case frame information, which KNP incorporates. Similar features are available for the tournament model as dynamic features, i.e., the case particle features. However, a higher-order round robin model enables to perform exact inference with the features.

Kitagawa's tree-based models [19, 20] have room for improvement. The models incorporate tournaments into transition-based parsers. These models choose attachment points by tournaments. In other words, a tournament is held in the subtree whose most probable head candidate (MPHC) will be the head if its corresponding action is

selected. The parser may recover from preceding wrong decisions *if and only if* the subtree is constructing *top-down*.

Tournaments can also be incorporated when a subtree is constructing bottom-up. Figure 9.1 shows how preceding wrong decisions are recovered in our proposal. In Figure 9.1(a) and (b), the MPHC of the right subtree, which consists of a single word, was selected by a tournament with no games. This is the tournament which Kitagawa's tree-based models incorporated. The *Left-Arc* action was selected and a dependency arc was added from "brown" to "fox". In Figure 9.1(c) and (d), since a word "fox" has become the new root of the subtree, another tournament is triggered. The tournament determines the heads of all the words in the subtree except for the root word, by comparing plausibilities of their current heads and the new root word. The new root word "fox" is selected as the new head of "A", and then the preceding wrong decision is recovered.

Chapter 10

Conclusions

There are several factors to achieve accurate Japanese dependency parsing. An accurate parsing model is requested. Quality and quantity of its training corpus have a great impact to its parsing accuracy. Since corpus annotation requires a lot of human effort, it is essential to support efficient and high-quality corpus annotation. Coordinated structures relate to constructing accurate parsers and performing efficient annotation.

We proposed three approaches to parse a sentence more accurately: developing a more accurate single parsing model, combining two or more dependency parsers to exploit their advantages, and combining a dependency parser with a coordination analyzer.

We developed an accurate Japanese dependency parsing model using a tournament model. Our model takes a dependent bunsetsu and its two candidate heads into consideration, and selects the better candidate head out of those two. This step is repeated in a step-ladder tournament to get the best candidate head. Our model has higher context discriminative performance than previous models which considers only one candidate. The accuracy of our model significantly outperforms that of the previous models in most experimental settings. Though the problem of parsing speed remains, our research showed that considering two or more candidate heads in a binary classification can achieve more accurate parsing.

We investigated partial parsing accuracies of three SVM-based Japanese dependency parsing models, and showed that the tournament model achieves the highest partial parsing accuracy among the three. If we select the top 60% of bunsetsus based on the scores defined with SVMs, the accuracy among the selected bunsetsus achieves 99%.

We showed that stacking with KNP significantly improves parsing accuracy. Since the tournament model and KNP are quite different parser, advantages of both of the

parsers are exploited. Stacking with KNP can be regarded as one of the easiest ways to incorporate co-occurrence information into the tournament model.

One of the difficult problems for dependency parsers is to identify spans of coordinated structures. We employ a coordination analyzer, which is specialized for capturing such structures. We demonstrated how we can integrate the outputs of a dependency parser and a coordination analyzer by using the dual decomposition algorithm. We first proposed a first-order-scored round-robin parsing model which preserves the advantage of the tournament model. Then, we define a common representation of the two analyzers to compare and take difference of their outputs, which are requested for updating the penalty variables. The dual decomposition algorithm with the common representation slightly improves dependency parsing and coordination analysis accuracies.

We also proposed the methods for efficient corpus annotation. Generally, parsing accuracy significantly drops if domains of its training and test data are different. For example, the training data is a newspaper corpus and the test data is a blog corpus. One of the ways to achieve an accurate parser for the blog data is to construct a sufficient amount of annotated blog corpus and to train a parser with the annotated corpus. We proposed two approaches for efficient corpus annotation: employing active learning, and exploiting coordinated structure annotation. The former reduces the number of sentences to be annotated required to achieve a certain objective parsing accuracy. The latter reduces the annotation cost for each sentence. Once a certain amount of target-domain corpus has been constructed, stacking can be applied to boosting parsing accuracy with a large amount of a source-domain corpus.

We explored an application of the SVM scores for active learning. Experiments showed that our SVM score-based active sampling method is applicable to assigning each unannotated sentence with its annotation priority. We can construct a training corpus to achieve accurate parsers with less manual effort. Our scoring method also supports corpus annotators by spotting which annotations are likely to be incorrect.

We developed a method to boost efficiency of dependency structure annotation using coordinated structure annotations. The constraints derived from the coordinated structure annotations restrict possible dependency structures of each sentence. Experiments with BCCWJ corpus showed that the method improves dependency parsing accuracy by 2-4%. The constraints can be utilized for detecting dependency structure annotation errors. The method is effective especially for annotation on a corpus containing a lot of coordinated structures.

Appendix A

Details of Feature Set

Tables A.1 and A.2 show the details of all the features used in our experiments. Note that fine-grained POSs, i.e., inflection types of verbs, are not used.

Figure A.1 shows the example of a feature vector. The sentence to be parsed is: “会談後、 山花氏は 相違点の 核心が 新党結成時期だったと 認め、 首相は 記者団への 説明で 「統一地方選前なんて できっこ ない。 タイミングと 言うだけだった。 私は 理解できないと 言った」と 述べ、 激しい 応酬だった ことを うかがわせた。”. The feature vector is generated by the tournament model when the focused-on dependent is “首相は”, the left candidate head is “言った」と” and the right candidate head is “うかがわせた。”.

#	Name	Description
Information of dependent's syuji		
S1	dep-syuji-form	lexical form
S2	dep-syuji-cpos	coarse-grained part-of-speech
S3	dep-syuji-pos	fine-grained part-of-speech
S4	dep-syuji-infform	inflection form of inflectable words
Information of dependent's gokei		
S5	dep-gokei-form	same as syuji's
S6	dep-gokei-cpos	
S7	dep-gokei-pos	
S8	dep-gokei-infform	
Dependent contain such word?		
S9	dep-commaperiod	contains comma (、) or period (。) ?
S10	dep-openbracket	lexical forms of all left brackets, or "nothing"
S11	dep-closebracket	lexical forms of all right brackets, or "nothing"
S12	dep-firstbunsetsu	"yes" if it is leftmost bunsetsu of sentence
Information of candidate bunsetsu		
S13-23	cand-*	S1-11 of candidate's
S24	cand-lastbunsetsu	"yes" if it is rightmost bunsetsu of sentence
bunsetsus between dependent and candidate		
S25	gap-distance	Distance between dependent and candidate 1(neighbor), 2-5, or 6+
S26	gap-josi	lexical forms of all particles gap contains
S27	gap-bracket	nothing, only open, only close or both
S28	gap-comma	yes or no
S29	gap-period	yes or no

Table A.1: Details of the standard feature templates.

#	Name	Description
Additional Features		
A1-4	cand-leftmost-*	S1-4 of the leftmost word of candidate
A5	dep-kakujosi	lexical form of all case particles contained in dependent
A6	cand-kakujosi	that of candidate's
A7	candright-form	whole lexical form of the candidate's right neighbor
Case Particle Features: all dependents of the candidate		
C1	cand-child-kakujosi	lexical forms of all case particles contained in such bunsetsus
C2	cand-child-kakujosi-common	that only the focused-on dependent also contains

Table A.2: Details of the additional feature and the case particle feature templates.

```
// standard features from dependent
dep-syuji-form:首相, dep-syuji-cpos:名詞, dep-syuji-pos:普通名詞,
dep-gokei-form:は, dep-gokei-cpos:助詞, dep-gokei-pos:副助詞,
dep-commaperiod:nothing,
dep-openbracket:nothing, dep-closebracket:nothing,

// standard features from candidate 1
cand-syuji-form:言った, cand-syuji-cpos:動詞, cand-syuji-infform:タ形,
cand-gokei-form:と, cand-gokei-cpos:助詞, cand-gokei-pos:格助詞,
cand-commaperiod:nothing,
cand-openbracket:nothing, cand-closebracket:」,

// standard features from candidate 2
cand2-syuji-form:うかがわ, cand2-syuji-cpos:動詞,
cand2-syuji-infform:未然形,
cand2-gokei-form:せた, cand2-gokei-cpos:接尾辞,
cand2-gokei-pos:動詞性接尾辞, cand2-gokei-infform:タ形,
cand2-commanperiod:period,
cand2-openbracket:nothing, cand2-closebracket:nothing,
cand2-lastbunsetsu:yes,

// standard features from between dependent and candidate 1
gap-distance:6+,
gap-josi:へ, gap-josi:の, gap-josi:で, gap-josi:なんて, gap-josi:と,
gap-josi:だけ, gap-josi:は,
gap-bracket:open, gap-comma:no, gap-period:yes,

// standard features from between dependent and candidate 2
gap2-distance:6+,
gap2-josi:へ, gap2-josi:の, gap2-josi:で, gap2-josi:なんて,
gap2-josi:と, gap2-josi:だけ, gap2-josi:は, gap2-josi:を,
gap2-bracket:both, gap2-comma:yes, gap2-period:yes,

// additional features and case particle features
cand-leftmost-form:言った, cand-leftmost-cpos:動詞, cand-leftmost-infform:タ形,
candright-form:述べ、, cand2-leftmost-form:うかがわ, cand2-leftmost-cpos:動詞,
cand2-leftmost-infform:未然形, cand-child-kakujosi:と, cand2-child-kakujosi:を
```

Figure A.1: An example of a feature vector.

Appendix B

Examples of Parsing Errors

This appendix lists examples of parsing errors. We employ the patterns of parsing errors defined by Yamamoto and Masuyama [69]. In contrast to their experiments, we extract dependency relations which none of parsers correctly parsed. Since the stacked parser described in Section 6.2 achieves the highest accuracy among our experiments, we employ the stacked parser, KNP and the tournament model without stacking. Experimental settings are same as in Section 6.2.

Each of extracted dependency relations is quite difficult to be parsed correctly since it is a relation which all of the parsers failed to identify its correct head. One of the reasons is an annotation error. The author rated appropriateness of each of the examples. For example,

1: 国選弁護人に なったと しても、 常に [適正、] [k 迅速な] [o 訴訟を] [t,s 追求し続けてほしい。]

describes that the focused-on dependent is “適正、” and its gold-standard head (o) is annotated to be “訴訟を”. KNP (k) predicted its head as “迅速な” and, the tournament model (t) and the stacked model (s) predicted its head as “追求し続けてほしい。”. Its appropriateness is rated as 1 (not the annotation but one of the parsers is correct) because the head of “適正、” should be “迅速な”, which is one of the parsers predicted.

A rating is one of the following: 5 (the annotation is correct), 4 (the annotation seems to be correct), 3 (the annotation and one of the parsers are equally appropriate), 2 (one of the parsers seems to be correct), 1 (one of the parsers is correct), x (both the annotation and the parsers are incorrect) or n (there is no appropriate head in the sentence).

We randomly sampled 10 dependency relations for each of the 12 patterns. We split

the dependency relations which do not match any of the 10 patterns of Yamamoto's based on whether the syuji POSs of their correct and predicted heads agree. Since parsing errors are classified into the patterns by rules based on POSs, dependency labels and other information, the following result contains automatic classification errors.

Examples and Their Ratings

Pattern 1 (a noun modifies a noun)

2: これを 受け、 同党は 同日 夕、 緊急の 三役懇談会を 開き、 十七日に 山花氏らの 会派離脱を 認めるかどうかを 協議する [臨時の] [o 三役懇談会と] [t,s,k 中執委を] 開く ことを 決めた。

2: 若い リード氏を 前面に 押し立て、 [単なる] [t,s,k 宗教組織から] [o 一般政治組織へと] イメージチェンジを 図るのが 狙いだった。

3: 「トックビルは こう 書いています。 『高貴な 人間は なかなか いない。 [町の] [t,s 法律屋や] [o 商売人や、] [k 時には] 下層階級の 人は いるが』と」

4: 先人が 情熱を 込めて [積み重ねてきた] [t,s,k 活字情報の] 膨大な [o 蓄積を、] 今 しっかりと 電子情報化しておく ことの 重要さをも 教えてくれる 書物である。

5: マルチメディアの 本は 溢れているが、 米国で [進み始めている] この [t,s,k 情報化の] 新しい [o 形態に] ついて、 その 真実に 迫っている ものは 少ない。

3: 「地球環境時代」に [対応した] [t 国の] [s,k 基本方針の] [o 在り方に] ついて 協議してきた 首相の 私的諮問機関 「２１世紀地球環境懇話会」は 十七日、 「新しい 文明の 創造に 向けて」と 題する 提言を まとめた。

2: [埼玉県内の] [t,s,k 愛犬家ら] [o 連続失跡事件で、] 埼玉・群馬両県警合同捜査本部は 十七日 午前 九時から、 埼玉県行田市の 会社役員、 川崎明男さんの 遺骨や 遺品が 捨てられた 疑いが 濃い、 として 群馬県利根郡川場村の 数カ所で 捜索を 始めた。

5: [船の] [t,s 固定資産税、] [o 登録免許税の] 軽減や 日本人船員の [k 雇用確保の] ための 助成が 柱と なっている。

5: 九、 十月に [報告された] [t,s,k 日本人の] [o エイズ患者・感染者は] 六十人と 過去最高を 記録した。

5: その 一つが、 検査に [出かけた] [t,s,k 検査院の] [o 職員が、] 対象機関の 職員から 酒食の もてなしを 受けたり、 観光地を 案内してもらったりし

ていたと いう ものだ。

Pattern 2 (the dependent is an argument of a predicate)
5: イスラエルの ラビン首相は １２日、 アンマンを 訪問し、 ヨルダンの [フセイン国王と] [t,s,k 首脳会談を] [o 行った。]
2: 正午ごろには、 金沢市清水谷町の 北陸自動車道で、 大型トラックや 乗用車 二、 [三台が] [t,s 凍結した] 坂道を [o 上れなくなって] [k 立ち往生した。]
5: とくに 吉川英治らの 剣技描写に 比べ、 藤沢の [それが] いかに [t,s リアルで] [o 細密かを、] 文章を 引いて 比較した くだりは [k 鮮やか。]
5: 「東京・立川の ハーフマラソンは 健常者が 車いすで 出場し、 [障害者と] [t,s,k レースを] [o 競う。] 車いすテニスには 健常者と 障害者が ペアで プレーする ニューミックスと いう 種目も あります。 将来は こうした、 障害者と 健常者とを 結ぶ、 スポーツ情報誌を 目指したい」と 抱負を 語る。
5: 人間は [数千年前から] [t,s,k 洋の東西を問わず、] 植物の 芳香や 薬としての 効果を [o 利用してきた。]
5: そして 国際社会で 環境立国を 唱える [日本政府が、] 霞ケ浦の 利水計画を [s,k 進めるのに] [t 際し、] 実は 環境影響評価法も 情報公開制度も [o 持ち合わせていない] ことを、 現場に 即して 指摘している。
1: このうえは、 来春闘で [要求と] [t,s,k 獲得実績が] [o 離れ過ぎないように、] 各産別は 全力を 挙げねばならない。
4: 被告の 国は 和解に 前向きな 姿勢を 示している [ものの、] 薬害エイズの 法的責任は [o 認めず、] 森井忠良厚相は 「厚生省も 当時は 最大限の 努力を した」と 述べて 原告側の 反発を [t,s,k 招いた。]
3: だが、 土地政策の [中で、] この 税制改正を どう [t,s,k 位置付けるか、] 明快な 理論付けが [o ないまま、] 緩和と いう 事実を 政治的に 先行させる ことに なった。
5: 二〇〇〇年の 完成を [目標に] 青森県六ケ所村に [o 建設中の] 再処理工場の 建設・運転計画にも 影響が [t,s,k 及ぶ。]

Pattern 3 (a noun modifies a predicate with the particle ‘‘*wa*’’(topicalizetion) or ‘‘*mo*’’(and))
4: [首相は] 十三日、 訪米から 帰国後、 さきがけ幹部と 電話で [o 会談し、] 新進党との 連携を 探る 山花貞夫新民連会長への 対抗策と して、 社会、 さきがけ両党の 連携を 深める 必要が あるとの 認識で [t,s,k 一致した。]

5: 子供が 父親から 認知された 途端に 児童扶養手当を 打ち切る よう 定めた [児童扶養手当法施行令は、] 非嫡出子に 対する 差別だと [o して、] 関西の 「婚外子差別と 闘う 会」など 三団体が 十三日、 厚生省に 施行令の 改正を 求める 要望を [t,s,k した。]
1: 両信組の 不良債権は、 十三日に 設立された 東京共同銀行が、 いったん全額を 引き継いだ 後、 四百億円に [ついては、] 東京都信用組合協会内に [o 設置する] 債権回収機関に 簿価で [t 譲渡、] [s,k 処理する。]
5: 戦後半世紀 たっても、 勝者と 敗者の 明暗は 歴然とし、 [前者は] [o 戦勝記念祝典を、] 後者は いま 戦後処理に [t,s,k 悩んでいる。]
4: 今年こそ [消費者は] [t,s,k 賢くなって、] ごみを [o 少なくしなければと] 思う。
3: 京都大理学部の 尾池和夫教授に よると 「今回の 地震は 四国から 近畿地方に かけて 走る 中央構造線北側の 西南日本内帯で 起きた。 ここには 小さい 活断層が 集中しており、 [地震は] 都市部の 直下型に [k なる] 傾向が [t,s 強く、] 被害が 大きくなるのが [o 特徴。] 内地での 地震は 一九四八年の 福井地震が 最後で、 広域に わたり 地震エネルギーが ため込まれていたと考えられる」と いう。
5: その [第一は] [t,s,k 言うまでも] なく 雇用の [o 確保だ。]
4: そう [なっても、] 社会規範から [o 外れない] 集団に [t,s,k する。]
2: ならば、 [財政危機宣伝は] 単なる [o 増税キャンペーンに] [t,s,k 過ぎぬ、と] 受け止められても やむを 得ないだろう。
2: 専門家の 話では、 この [クモは] 攻撃性が [t,s なく、] 素手で 触らない 限り、 [k かまれる] ことは [o ない。]

Pattern 4 (the dependent is a head of a subordinate clause)
4: 今年の 大阪会合でも、 米国の 早急な 自由化の 主張に [対し、] 途上国の 立場を [o 強調する] 考えを [t,s,k 示している。]
5: その ため、 キチンは [使い] [o 勝手を] [t,s,k 考えた] コンパクトなものに した。
5: 今から [二十年前の] [t,s,k 七〇年代] [o 半ばの] ことである。
5: 基地周辺の 警戒に 当たる [警察官の] [o 一人の] [t,s,k 話では、] 緊張と 恐怖を まぎらす ためか、 ロシア軍兵士が 酔っぱらって 基地周辺で 自動小銃を 発砲し 死傷者が 出た ことも あったと いう。
1: 老壮青 三世代、 この ライカ・マニアの 影の 指南番が、 [本書の] [t,s,k 著者、] [o 田中長徳氏である。]

4: こうした 実態から、 今年の 「海運白書」では 「このまま [推移すれば、] 二〇〇〇年には 日本籍船は 百隻を 割り、 日本人船員は 四千人を [o 下回る] 可能性が [t,s,k 大きい」とまで] 予測した。
3: 新大綱の 性格を 一口で [言えば、] 米国の 「東アジア戦略報告」を 基調に しながら、 日米安保条約の 「再定義」の 内容を [o 先取りした] [t,s,k ものだ。]
2: 和平協定の 核に あるのは、 敵対してきた 民族が 元通り 一緒に 住むのは [無理であるから、] 居住地域を [o 分離して] 紛争再発を [t,s,k 防ごうと] いう 考えだ。
3: これらに [バトンタッチするまで、] 仮設住宅の 生活が 社会から [o 取り残されない] よう 行政は 知恵を [t,s,k 絞ってほしい。]
4: [高級官僚の] [t,s,k 接待] [o 漬け。]

Pattern 5 (the dependent is an argument, and the dependency label is P or I)
1: 同省では この ほか、 [中小企業信用保険公庫と] [t,s,k 中小企業金融公庫の] [o 一部統合などの] 案も 検討している。
5: 今回は アジアからの 視点で これら 4つの テーマを 見つめ直し、 日本と アジアの 過去、 [現在、] [o 未来を] [t,s,k 考える。]
2: 市営地下鉄は 全線不通と なった ため、 [豊中市方面から] [o 大阪市内へ] [t,s,k 向かう] 新御堂筋など 市内の 主要道路は マイカーに 切り替えた 通勤サラリーマンらで、 大混雑と なった。
1: 一つの 時代の 終わりに 当たって、 ラグビーのみならず、 すべての スポーツの 未来の [ために、] スポーツの [o 今日的意味などに] ついて、 真剣な 論議を 起こす よう [t,s,k 要望しておきたい。]
5: それが [健康と] [o 命を] [t,s,k 守る] 費用なら 納得も いく。
5: 難しかろうが、 会議の [安全と] [t,s 街の] [o にぎわいを] [k 両立させる] スマートな 運営もまた、 国際都市の 条件と 言えまいか。
5: [熱帯から] [o 亜熱帯に] [t,s かけ] [k 生息し、] 国内では 沖縄・八重山諸島で 見つかった 記録が あるだけである。
5: この 背後には、 [金大統領と] [s,k 野党・新政治国民会議の] [o 金大中総裁の] [t 宿命の] 対立が あると いうのが、 ソウルの 政界通の 分析だ。
5: 市場価格の 低落を 受けて わずかでも 引き下げを 目指す [政府側と、] 「来るべき 総選挙を にらみ、 政治判断で 決める」と 据え置きを [t,s 主張する] 与野党の [o 農林議員が、] 例年のように 攻防を [k 繰り返した。]

5: 日米韓三国が、 [外交交渉と] [o 政策立案で] これほど [t,s,k 協力し合った] ことは なかったからだ。

Pattern 6 (a label-I relation whose dependent's particle is ''*wa*'' or ''*mo*'')

5: 「中流意識」を 細かく 分析すると、 「中の [上」では] 男性より 女性の [o 方が、] 「中の 下」は 女性より 男性の [k 方が] それぞれ 五ポイント前後 [t,s 多い。]

5: [広告会社社長には] 同社から [o 一七%、] 販売を 「孫請け」 した 投資家には 広告会社社長から 七、 [k 八%の] 販売手数料が [t,s 支払われていた。]

3: 九日連続で 乾燥注意報が 発令中の 東京都内で 十五日、 体の 不自由な老婦人が 焼死、 今年に 入っての [焼死者は] [o 十六人と] [t,s,k なった。]

5: 米ナショナル・フットボール・リーグの プレーオフ最終戦が １５日 行われ、 アメリカン・カンファレンスでは サンディエゴ・チャージャーズ、 [ナショナル・カンファレンスでは] [o サンフランシスコ・フォーティナイナーズが] それぞれ [t,s,k 優勝。]

5: [昼は] [o 勘九郎、] 夜は 吉右衛門を ゲストに 招き、 藤山直美、 国広富之が 参加した [t,s,k 一座。]

5: 昼は 勘九郎、 [夜は] [o 吉右衛門を] ゲストに 招き、 藤山直美、 国広富之が [k 参加した] [t,s 一座。]

5: [「一九四三年には] [o 朝鮮、] 一九四五年には [k 台湾でも] 徴兵制が しかれた。 国民徴用令などに よって 技術者、 学生、 女子らも 軍需工場に 動員され、 さらに 朝鮮人の ほか 中国人も 強制連行され 苛酷な 労働を [t,s 強いられた」。]

5: 確かに [医師は] [o 処方せん発行料を、] 薬局側は 調剤料や [k 薬歴管理料を] [t,s 取る。]

5: 科目数も [昨年は] [o 五十一、] 今年は [s,k 八十五と] [t 多彩。]

5: すでに [ヨーロッパ諸国は] 一九七九年に [o 「長距離越境大気汚染条約」を、] 米国と カナダは 九一年に [k 「酸性雨防止協定」を] 結び、 原因物質の硫黄酸化物と 窒素酸化物減らしを [t,s 進めてきた。]

Pattern 7 (a verb modifies a verb with label P)

5: 混迷を 深める 社会党の 党内問題の [収拾策や、] 伊東秀子衆院議員の [s,k 出馬で] 揺れる [o 北海道知事選に] ついて [t 協議すると] みられる。

5: 鉄道の [遅れや] [t,s,k 高速道で] [o 渋滞するなどの] 影響が 出たが、この 日から 三連休の 人出を 期待する 各地の スキー場は ホクホク顔。
1: 六九年 十一月、 ワシントンで 日米首脳会談が [行われ] 「核抜き、 本土並み、 七二年 [o 返還」が] [t,s,k 約束された。]
1: エチオピアの アラミス地区で 発掘された ４４０万年前の 最古の 人類化石で、 エチオピアで [見つかり、] これまで 最古と [o されていた] アファール猿人より 約１００万年 人類史を [t,s,k さかのぼる。]
5: 現在の 日本は、 [国連安保理常任理事国入りや、] [k アジアでの] リーダーと しての [s 役割に] 代表される [o 国際化、] [t 経済大国と] いう 勲章の 代償と して 産業空洞化など、 多くの 課題を 抱えている。
3: この ほか、 清水歯科医院の 清水英寿さんは、 ラベンダー、 ミント、 桐、ユズなどの 香りを 診療前の 患者に [かがせ、] 不安を [t,s,k 取り除き、] 精神的に [o 沈静させる] ために 使っている。
5: 例えば、 介護保険を 実施するには 年間 四兆円 かかると [され、] だれが、 どう [t,s,k 負担するかが] 大問題に [o なっている。]
3: だが、 その ためには 与野党の 目先の 思惑を [排し、] 静かに 落ち着いて 話が [t,s,k 聞ける] 環境が [o 必要である。]
5: 新党は 「三党とも 過半数割れ」を スローガンに、 国民党の 金権腐敗選挙、 暴力団との [つながり、] 李総統の [o“台独”傾向などを] 批判して 選挙戦を [t,s,k 戦ったが、] 国民党の 「過半数割れ」は 得票率では 現実と なった。
5: 「政府・与党の 幹部に 財政収支の 対国内総生産比は、 イタリアにも [抜かれ、] [o Ｇ７中最悪だと] [t,s,k 説明したら、] 一様に ショックを 受けていたようだ」と 大蔵省幹部が 話している。

Pattern 8 (apposition)
4: 就職、 結婚、 子供の 誕生、 転職、 妻の [大病と、] さまざまな [o 出来事が] [t,s,k あったが、] 新しい 世界へ、 広い 世界へと 私の 人生は 開けてきた 歳月であったような 気が する。
5: また、 大阪市立大文学部志望の [同府立吹田東高三年、] [o 山本和子さんは] 「私は 文学部志望だけど、 博物館の 学芸員の 資格を とりたい」と [t,s,k 話していた。]
5: マレーシアの マハティール首相、 インドネシアの [スハルト大統領など] この [t,s,k 地域の] [o 有力指導者に] 大阪会議の 意義を アピールする 予定と なっている。

5: 神奈川県の スポーツ事情に 詳しい 鳴海正泰・関東学院大教授は 「温暖な 気候や 高い [所得水準など] [t,s,k スポーツの] [o 環境が] 整っている。 横浜が 欧米文化の 窓口に なった ため、 新しもの好きな ところも ある」と 説明する。
5: 山花貞夫・新民連会長は 十六日の 記者会見で、 [村山富市首相ら] [o 社会党執行部と] [t,s,k さきがけが] 連携強化を めざした 問題に ついて 「私たちの 行動が 新しい 政界の 動きを 作ったと いえる。 統一会派を 超えて 将来の 日本の 政治の 在り方を 考えると いうのであれば、 そういう 議論に 加わる ことも ある」と 述べた。
5: 会議には シュルツ元米国務長官、 [エバンズ豪外相ら] [t,s,k アジア・太平洋地域] 八カ国の [o 政治指導者が] 出席した。
5: 各社は 回答の 中で、 「表」と 呼ばれる 電気設備工事の 営業担当の 部課長クラス 数人の ほか、 「調査部」 [「計画部」など] [t,s,k 社に] よって 名称は 異なるが、 「裏」と 呼ばれ 談合を 専門に 行う [o 部の] 責任者の 氏名、 簡単な 経歴などに ついても 文書で 明らかにしている。
5: 企業の 生産性、 所得水準、 [住宅など] [t,s,k 東西の] [o 経済格差は] まだ 大きく、 西からの テコ入れは 少なくとも 今世紀 いっぱい 必要と みられている。
4: アメリカの [自然保護団体、] [t,s,k シエラクラブや] [o オーデュボン協会、] 世界の 各地で 行動する グリーンピースや 地球の友などの 調査研究の 水準は、 しばしば 政府の それを 上回る。
5: 歩道や 出入り口の 段差、 歩道橋や 駅の [階段など] [t,s,k 移動を] 阻む 「物理的な [o 障壁」、] 大学入試や 資格制度や 就職試験で 体験する 「制度の 障壁」、 視覚障害者や 聴覚障害者を 悩ませる 「文化・情報面の 障壁」、 そして 障害者への 偏見・差別から 優越感の 裏返しのような 憐れみまで 「意識上の 障壁」である。

Pattern 9 (the dependent is an adverb)
5: Ｔ・Ｓエリオットの 詞を もとに アンドリュー・ロイド＝ウェバーが 作曲、 トレバー・ナンが 演出した ミュージカルだが、 登場するのは [すべて] [o 猫と] いう 意表を [t,s,k ついた] 作品である。
4: 近所の 飲み屋で ほろ酔いで 演じた [ところ] [o 拍手喝さい、] 五年前から 製図学校を 息子さんに まかせ、 大道芸人を 各地に [k 派遣する] 会社に 所属し、 プロに [t,s なってしまった。]
5: その [ひとつ] [o カワチは、] [t,s,k 宗教都市であった。]

5: けが人が なかった ことも あり、 マンションの 自治組織は、 計四棟 ある マンションの [うち、] 問題の [o 棟にだけ] 注意を 呼び掛ける ビラを [k 張り出した] 以外に 特段の 対策を [t,s 講じなかった。]
5: 地球の [ちょうど] [o 反対側に] [t,s,k ある] ブラジルは、 日本とは 多くの 面で 対照的だ。
2: 地域安保システムと それへの 貢献に ついて、 [もっと] [k 体系的に] [t,s 取り組む] 姿勢を [o 打ち出すべきではなかったか。]
1: 「疑惑が 持たれた 場合には [自ら] [k 真摯な] 態度を [o もって] 疑惑を [t,s 解明し、] その 責任を 明らかにする」と 政治倫理綱領には 明記されている。
1: [いずれも] 気の [o 遠くなるような] 困難な [t,s,k 仕事であり、] 失敗は 許されない。
3: この [ほか] 政府開発援助や 阪神大震災を 契機と [k した] 公共土木施設の 耐震安全性など 六件に ついても [t,s 検査し、] 「特記事項」と して 指摘、 事業の 改善などを [o 求めた。]
5: 国は [既に] 三次に [t,s,k わたる] 補正予算で、 三兆円を 超える 震災地向け予算を 組み、 緊急の 支援策に 全力を [o 挙げたと] いう。

Pattern 10 (the dependent is a conjunction)
2: 「うちは 官公庁に 強く、 受注の 六割を 占めるが、 景気の 影響で 需要が 落ちてきた。 [しかし、] 役に 立つ サービスを [o すれば] 収益は [t,s,k ついてくる」]
3: [また、] 十四日 スタートの 挑戦艇シリーズの 組み合わせ抽選会が 同日 [t,s 行われ、] ニッポンチャレンジは 第一日に ワン・オーストラリアと [k 対戦する] ことに [o なった。]
3: [一方、] 米東部は 暖冬異変と [o なり、] いくつかの 地域で この 時期と しては 史上 最高の 気温を [t,s,k 記録した。]
4: [一方、] 村山富市首相は 十六日、 さきがけ代表の 武村正義蔵相と 首相公邸で [o 会い、] 両党の 連携強化と 政界再編に 向けた 政策研究会の 設置でも 合意、 緊迫した 局面と [t,s,k なった。]
2: [だが、] 安全構造の 崩壊が [o 言われ、] 防犯対策を 希求する 国民の心情と およそ [t,s,k かみ合わないのではなかろうか。]
4: [一方、] ガリ事務総長は 昨年 「開発への 課題」と 題する 報告書を 国連総会に [o 提出したが、] 平和維持活動の 行き詰まりも あって、 国連は 安全保障よりも 経済社会開発を 重視する 傾向を 一段と [t,s,k 強めている。]

3: [そして] 流通ルートが [o 多様化され、] 来年 六月からは 新規登録業者も 交え 競争が [t,s,k 激化する。]
2: [しかし、] 道は 厳しくとも コメは 特別 扱いを [o 外れ、] 市場経済の中の 一商品と しての 歩みを 着実に [t,s,k 進めなければならない。]
3: [しかし、] 身分が [o あいまいで、] 国際調査の 窓口には [t,s,k ならない。]
4: [にもかかわらず、] 今回、 特例とはいえ あっせんを 復活させた [o ことは、] カネの かからない 政治を 目指すと いう 政治改革の 原点から 外れているだけでなく、 時代に 逆行した [t,s,k 行為だ。]

Others (syuji POSs of heads agree)
5: 政党助成を 行う 主要国の 例は ドイツが 年間 約百五十億円で 国民 一人当たり [百八十四円] [k ▽フランス] [t,s 約百五億円、] [o 同百八十三円] ▽スウェーデン 約十九億円、 同二百十七円――などだ。
2: ジョンソンは、 十二日 午前 四時 四十五分ごろ、 東京都港区六本木の ビルで、 女性 ３人が エレベーターに 乗ろうとしたのを 邪魔し、 １人の 顔を [殴って] １０日間の けがを [t,s,k させ、] 警視庁麻布署に 傷害で [o 現行犯逮捕された。]
5: 豚の 背脂入りの パックの せいでしょう、 こう 言っちゃあ なんですが、枯れた 味わいのはずの タラが、 二十歳は [若返って] [t,s,k はつらつとした] おいしさに [o 変身します。]
5: 文章の 端正、 物語上手、 時代考証の [確かさ、] [t,s,k 下級武士や] 市井の 人々への 優しい [o 視点、] 英雄嫌いなど 藤沢周平の 魅力を 次々に 解き明かしていく。
5: スポーツクラブ 「セントラル・スポーツ」の 綿貫道雄常務は 「運動不足を 解消したい [サラリーマン、] [t,s,k ダイエット目的の] [o 女性が] 多い」と 話す。
1: 久しぶりに 映画 「男は つらいよ」 「釣りバカ日誌」の 最新作を [見て、] [o 共通した] ある メッセージに 気が [t,s,k ついた。]
n: この 時 知り合い 「兄貴」と 慕った 暴力団関係者＝八四年 二月 [失跡、] 当時 [o 五十一歳＝には] 八三年 十月に 風間博子容疑者と 結婚する 際、 前妻との 離婚を [t,s,k 仲介してもらった。]
1: 問題は、 [今回のような] 極端な [o 矛盾と] [t,s,k 不合理を、] 税務当局が どう 考え、 是正の いかなる 努力を していたか、が 国民の 目には 見えにくい ことだ。

5: さまざまな [思い、] [o 利害関係が] [t,s,k もつれ、] 各地で 推進派と反対派が 泥仕合を 演じてきた。
1: 具体的には、 [国税である] [t,s,k 地価税と] [o 地方税の] 固定資産税の役割分担などを 総合的に 検討するのが 本来の 筋であった。

Others (syuji POSs of heads disagree)
5: 今回の 事件で はからずも 表面化した 通り、 [常に] 生きものを [t,s,k 殺す] 薬品を [o 所持しているのだ。]
x: 二十二日投開票の 愛媛県知事選で、 同県選管が 県教委の 協力で 県内に [ある] [o すべての] [t 公立、] 私立の 幼稚園、 小、 中学校の 児童、 [s,k 生徒らに] 保護者への 投票を 呼びかける 啓発ビラを 家庭に 持ち帰らせていた ことが 十三日 分かった。
2: 十四日 午前 八時 [五十分ごろ、] 富山県新湊市立町の 川で 同市八幡町、無職、 清水正一さんが 漁船 「裕高丸」の 雪下ろしを [o していた] ところ、船ごと 川に [t,s,k 水没。]
5: 文字や [数字など、] いわば 頭の [o 教育は、] 学校が
嫌というほど [t,s,k してくれます。]
5: 大きさは もちろん、 体形まで [そっくりで、] 樹上の ゴイサギを [t 見つけた] 市民や 入場者から 「飛ばないはずの ペンギンが 木の 上に いる」と、通報が [s,k 寄せられる] [o ほどだ。]
5: 政府は 同日 午前、 国土庁で [災害対策関係省庁連絡会議、] 非常災害対策本部の [o 初会合を] 相次いで 開き、 対策を [t,s,k 協議した。]
5: リサイタルの 収支は、 五百人程度の 会場が [いっぱいになって] [o トントンと] [t,s,k いうのだから、] 自ら チケット販売に 力を 入れざるを得ない。
5: 昨年 四月の 経済同友会の 提言は 「少しでも 序列の 高い 大学への 入学を 競う 偏差値競争が 教育の 病理を 深刻化させると いう 事態を 招いた。これに ついては、 産業界の 責任も 大きい」と 自らの 非を 認め、 「指定校制度の 完全撤廃は もちろん、 出身大学名を [聴取しない、] [t 学校歴ではなく] 学習歴を [o 重視する] [s,k ことを] 目に 見える 形で 示していく」 ことを 求めた。
1: 国選弁護人に なったと しても、 常に [適正、] [k 迅速な] [o 訴訟を] [t,s 追求し続けてほしい。]
4: ユーザーに [とって、] パソコンＯＳの 多くが 同じ ソフトに [t,s なる] ことは 使い勝手が よくなり、 [o 望ましいと] [k いえる。]

Pattern	Rating							Total	Total	Expected
	5	4	3	2	1	x	n	(Sampled)		# 5,4
1	4	1	2	3				10	278	139
2	6	1	1	1	1			10	810	567
3	3	3	1	2	1			10	583	350
4	3	3	2	1	1			10	200	120
5	7			1	2			10	95	67
6	9		1					10	10	9
7	6		2		2			10	224	134
8	8	2						10	28	28
9	5	1	1	1	2			10	215	129
10		3	4	3				10	37	11
Others (agree)	5			1	3		1	10	190	95
Others (disagree)	6	1		1	1	1		10	433	303
Total	62	15	14	14	13	1	1	120	3103	1952

Table B.1: Annotation appropriateness ratings for each of parsing error patterns.

Discussion

Table B.1 shows the distribution of patterns and ratings. The number of examples whose annotations are appropriate (ratings 5 and 4) are 77 (64%). There are 28 (23%) inappropriate (ratings 2, 1 and x) examples. There are 15 (13%) examples each of which is difficult to decide a single appropriate head (ratings 3 and n). Annotations corresponding to patterns 5 (noun phrase coordination), 6 (incomplete coordinated structure), and 8 (apposition) are relatively appropriate. This suggests coordinations and appositions are difficult to be parsed correctly.

We can roughly estimate the upper bound of dependency parsing accuracy from the table. As shown in Table B.1, there are 3103 dependency relations which no parsers correctly parsed. Since the test data consists of 80695 dependency relations, if an oracle selector is given, $80695 - 3103 = 77592$ dependency relations will be correct; the oracle selector selects a parser which correctly parses a particular dependency relation

among the stacked model, KNP, and the tournament model. It is the dependency accuracy of 96.15%. The expected number of appropriate (ratings 5 and 4) dependency relations is 1952. These could be correctly parsed by a very accurate parser. The accuracy becomes 98.57% (79544/80695). The other 1151 dependency relations are hardly parsed *correctly* even by a human.

Bibliography

[1] Sabine Buchholz and Erwin Marsi. CoNLL-X Shared Task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 149–164, 2006.

[2] Xavier Carreras. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 957–961, 2007.

[3] Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. Improving dependency parsing with subtrees from auto-parsed data. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 570–579, 2009.

[4] Wenliang Chen, Youzheng Wu, and Hitoshi Isahara. Learning reliable information for dependency parsing adaptation. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 113–120, 2008.

[5] Yoeng-Jin Chu and Tseng-Hong Liu. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400, 1965.

[6] Ido Dagan and Sean P. Engelson. Selective sampling in natural language learning. In *Proceedings of IJCAI-95 Workshop on New Approaches to Learning Natural Language Processing, Fourteenth International Joint Conference on Artificial Intelligence*, pages 41–48, 1995.

[7] Yasuharu Den, Toshinobu Ogiso, Hideki Ogura, Atsushi Yamada, Nobuaki Minematsu, Kiyotaka Uchimoto, and Hanae Koiso. A language resource for Japanese corpus linguistics: Development of an electronic dictionary for morphological analysis and its application (in Japanese). *Japanese Linguistic Science*, 22:101–122, 2007.

[8] Jack Edmonds. Optimum branchings. *Journal of Research of the Natural Bureau of Standards*, 71B:233–240, 1967.

[9] Yoav Goldberg and Michael Elhadad. An efficient algorithm for easy-first non-directional dependency parsing. In *Proceedings of the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750, 2010.

[10] Kazuo Hara, Masashi Shimbo, Hideharu Okuma, and Yuji Matsumoto. Coordinate structure analysis with global structural constraints and alignment-based local features. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 967–975, 2009.

[11] Liang Huang and Kenji Sagae. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086, 2010.

[12] Rebecca Hwa. Sample selection for statistical grammar induction. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 45–52, 2000.

[13] Rebecca Hwa. On minimizing training corpus for parser acquisition. In *Proceedings of the Fifth Computational Natural Language Learning Workshop*, pages 1–6, 2001.

[14] Ryu Iida, Kentaro Inui, Hiroya Takamura, and Yuji Matsumoto. Incorporating contextual cues in trainable models for coreference resolution. In *Proceedings of the Tenth EACL Workshop 'The Computational Treatment of Anaphora'*, 2003.

[15] Masakazu Iwatate, Masayuki Asahara, and Yuji Matsumoto. Japanese dependency parsing using a tournament model. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 361–368, 2008.

[16] Hiroshi Kanayama, Kentaro Torisawa, Yutaka Mitsuishi, and Jun'ichi Tsujii. A statistical Japanese dependency analysis model with choice restricted to at most three modification candidates (in Japanese). *Journal of Natural Language Processing*, 7(5):71–91, 2000.

[17] Daisuke Kawahara and Sadao Kurohashi. A fully-lexicalized probabilistic model for Japanese syntactic and case structure analysis. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 176–183, 2006.

[18] Daisuke Kawahara and Sadao Kurohashi. Coordination disambiguation without any similarities. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 425–432, 2008.

[19] Kotaro Kitagawa and Kumiko Tanaka-Ishii. Tree-based deterministic dependency parsing — an application to Nivre's method —. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 189–193, 2010.

[20] Kotaro Kitagawa and Kumiko Tanaka-Ishii. Tree-based deterministic dependency parsing (in Japanese). In *Proceedings of the 17th Annual Meeting of the Association for Natural Language Processing*, pages 1079–1082, 2011.

[21] Terry Koo, Xavier Carreras, and Michael Collins. Simple semi-supervised dependency parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 595–603, 2008.

[22] Terry Koo and Michael Collins. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11, 2010.

[23] Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298, 2010.

[24] Taku Kudo and Yuji Matsumoto. Japanese dependency structure analysis based on support vector machines. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 18–25, 2000.

[25] Taku Kudo and Yuji Matsumoto. Japanese dependency analysis using cascaded chunking. In *Proceedings of the Sixth Conference on Natural Language Learning*, pages 63–69, 2002.

[26] Taku Kudo and Yuji Matsumoto. Fast methods for kernel-based text analysis. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, pages 24–31, 2003.

[27] Taku Kudo and Yuji Matsumoto. Japanese dependency parsing using relative preference of dependency (in Japanese). *IPSJ Journal*, 46(4):1082–1092, 2005.

[28] Kow Kuroda and Ryu Iida. Proposing a multidimensional representation scheme that effectively specifies the (co-)argument structures of (potentially) any words – using the simplified version of (parallel) pattern matching analysis– (in Japanese). *Technical Report of IEICE*, 106(191):1–6, 2006.

[29] Sadao Kurohashi. Which should we build first, a corpus, or a parser? (in Japanese). *IPSJ Magazine*, 41(7):769–773, 2000.

[30] Sadao Kurohashi and Makoto Nagao. A syntactic analysis method of long Japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20(4):507–534, 1994.

[31] Sadao Kurohashi, Igura Yuiko, and Masako Sakaguchi. Annotation manual of the morphologically- and syntactically-tagged corpus (in Japanese). Technical report, Kyoto University, 2000.

[32] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, 1994.

[33] Dekang Lin, Kenneth Church, Heng Ji, Satoshi Sekine, David Yarowsky, Shane Bergsma, Kailash Patil, Emily Pitler, Rachel Lathbury, Vikram Rao, Kapil Dalwani, and Sushant Narsale. New tools for web-scale n-grams. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation*, pages 2221–2227, 2010.

[34] Kikuo Maekawa. Balanced corpus of contemporary written Japanese. In *Proceedings of the 6th Workshop on Asian Language Resources*, pages 101–102, 2008.

[35] Torres Martins, André Filipe, Dipanjan Das, Noah A. Smith, and Eric P. Xing. Stacking dependency parsers. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 157–166, 2008.

[36] Ryan McDonald and Fernando Pereira. Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 81–88, 2006.

[37] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajiĉ. Non-projective dependency parsing using spanning tree algorithm. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530, 2005.

[38] Ryan McDonald and Giorgio Satta. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of the Tenth International Conference on Parsing Technologies*, pages 121–132, 2007.

[39] Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. Bayesian unsupervised word segmentation with nested pitman-yor language modeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 100–108, 2009.

[40] Shinsuke Mori. A stochastic parser based on an SLM with arboreal context trees. In *Proceedings of the 19th International Conference on Computational Linguistics*, 2002.

[41] Joakim Nivre. An efficient algorithm for projective dependency parsing. In *Proceedings of the eighth International Workshop on Parsing Technology*, pages 149–160, 2003.

[42] Joakim Nivre. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553, 2008.

[43] Joakim Nivre. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359, 2009.

[44] Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. The CoNLL 2007 Shared Task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL-2007*, pages 915–932, 2007.

[45] Joakim Nivre and Ryan McDonald. Integrating graph-based and transition-based dependency parsers. In *Proceedings of 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 950–958, 2008.

[46] Joakim Nivre and Jens Nilsson. Psuedo-projective dependency parsing. In *Proceedings of 43rd Annual Meeting of the Association for Computational Linguistics*, pages 99–106, 2005.

[47] Joakim Nivre and Mario Scholz. Deterministic dependency parsing of English text. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 64–70, 2004.

[48] Hideharu Okuma, Kazuo Hara, Masashi Shimbo, and Yuji Matsumoto. Bypassed alignment graph for learning coordination in Japanese sentences. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 5–8, 2009.

[49] Hideharu Okuma, Kazuo Hara, Masashi Shimbo, and Yuji Matsumoto. Bypassed alignment graph for learning coordination in Japanese sentences: supplementary material. Technical report, Nara Institute of Science and Technology, 2009.

[50] Emily Pitler, Shane Bergsma, Dekang Lin, and Kenneth Church. Using web-scale n-grams to improve base NP parsing performance. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 886–894, 2010.

[51] Kenji Sagae and Alon Lavie. Parser combination by reparsing. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 129–132, 2006.

[52] Kenji Sagae and Jun'ichi Tsujii. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL-2007: the Joint Conferences on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1044–1050, 2007.

[53] Erik F. Tjong Kim Sang and Sabine Buchholz. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132, 2000.

[54] Manabu Sassano. An empirical study of active learning with support vector machines for Japanese word segmentation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 505–512, 2002.

[55] Manabu Sassano. Linear-time dependency analysis for Japanese. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 8–14, 2004.

[56] Manabu Sassano and Sadao Kurohashi. A unified single scan algorithm for Japanese base phrase chunking and dependency parsing. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 49–52, 2009.

[57] Satoshi Sekine, Kiyotaka Uchimoto, and Hitoshi Isahara. Statistical dependency analysis using backward beam search (in Japanese). *Journal of Natural Language Processing*, 6(3):59–73, 1999.

[58] Masashi Shimbo and Kazuo Hara. A discriminative learning model for coordinate conjunctions. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 610–619, 2007.

[59] Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. The CoNLL-2008 Shared Task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177, 2008.

[60] Akihiro Tamura, Hiroya Takamura, and Manabu Okumura. Japanese dependency analysis using the ancestor-descendant relation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 600–609, 2007.

[61] Min Tang, Xiaoqiang Luo, and Salim Roukos. Active learning for statistical natural language parsing. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 120–127, 2002.

[62] Cynthia A. Thompson, Mary Elaine Califf, and Raymond J. Mooney. Active learning for natural language parsing and information extraction. In *Proceedings of the 16th International Conference on Machine Learning*, pages 406–414, 1999.

[63] Yuta Tsuboi, Hisashi Kashima, Shinsuke Mori, Hiroki Oda, and Yuji Matsumoto. Training conditional random fields using incomplete annotations. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 897–904, 2008.

[64] Kiyotaka Uchimoto, Masaki Murata, Satoshi Sekine, and Hitoshi Isahara. Dependency model using posterior context (in Japanese). *Journal of Natural Language Processing*, 7(5):3–17, 2000.

[65] Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Isahara. Japanese dependency structure analysis based on maximum entropy models (in Japanese). *IPSJ Journal*, 40(9):3397–3407, 1999.

[66] Yotaro Watanabe, Masakazu Iwatate, Masayuki Asahara, and Yuji Matsumoto. A pipeline approach for the syntactic and semantic dependency parsing. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 228–232, 2008.

[67] Hiroyasu Yamada and Yuji Matsumoto. Statistical dependency analysis with support vector machines. In *Proceedings of the eighth International Workshop on Parsing Technology*, pages 195–206, 2003.

[68] Yuji Yamamoto and Shigeru Masuyama. Statistical Japanese dependency analysis reflecting relative distances among modifiee candidates (in Japanese). In *IPSJ SIG Technical Report 2007-NL-182*, pages 15–22, 2007.

[69] Yuji Yamamoto and Shigeru Masuyama. Parsing error patterns and structural ambiguities in Japanese dependency parsing (in Japanese). In *Proceesings of the 15th Annual Meeting of the Association for Natural Language Processing*, pages 789–792, 2009.

[70] Yuji Yamamoto and Shigeru Masuyama. Statistical Japanese dependency analysis reflecting relative distances from a modifier among modifee candidates (in Japanese). *IEICE TRANSACTIONS on Information and Systems*, J93-D(6):1036–1047, 2010.

[71] Naoki Yoshinaga and Masaru Kitsuregawa. Kernel slicing: Scalable online training with conjunctive features. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1245–1253, 2010.

List of Publications

Journal Papers

1. Masakazu Iwatate, Masayuki Asahara and Yuji Matsumoto. Japanese Dependency Parsing Using a Tournament Model (in Japanese). Journal of Natural Language Processing, Vol.15, No.5, pages 169–185, 2008.

2. Masaki Murata, Tamotsu Shirado, Kentaro Torisawa, Masakazu Iwatate, Koji Ichii, Qing Ma and Toshiyuki Kanamaru. Extraction and Visualization of Numerical and Named Entity Information from a Very Large Number of Documents Using Natural Language Processing. International Journal of Innovative Computing, Information and Control, Vol. 6, No. 3(B), pages 1549–1568, 2010.

International Conferences (Reviewed)

1. Masakazu Iwatate, Masayuki Asahara and Yuji Matsumoto. Japanese Dependency Parsing Using a Tournament Model. In Proceedings of the 22nd International Conference on Computational Linguistics, pages 361–368, August 2008.

2. Masaki Murata, Masakazu Iwatate, Koji Ichii, Qing Ma, Tamotsu Shirado, Toshiyuki Kanamaru and Kentaro Torisawa. Extraction and Visualization of Numerical and Named Entity Information from a Large Number of Documents. In Proceedings of the 2008 IEEE International Conference on Natural Language Processing and Knowledge Engineering, pages 122–139, October 2008.

Other Publications

1. Masakazu Iwatate, Masayuki Asahara and Yuji Matsumoto. Japanese Dependency Parsing Using a Tournament Model (in Japanese). In Proceedings of the

14th Annual Meeting of the Association for Natural Language Processing, pages 237–240, March 2008.

2. Masaki Murata, Masakazu Iwatate, Koji Ichii, Qing Ma, Tamotsu Shirado, Toshiyuki Kanamaru, Sachiyo Tsukawaki and Hitoshi Isahara. Text Mining and Visualization System for Numerical and Named Entity Information from a Large Number of Documents (in Japanese). IPSJ SIG Technical Report, 2008-NL-184, pages 25–32, March 2008.

3. Yotaro Watanabe, Masakazu Iwatate, Masayuki Asahara and Yuji Matsumoto. A Pipeline Approach for Syntactic and Semantic Dependency Parsing. In Proceedings of the Twelfth Conference on Natural Language Learning (Shared Task Session), pages 228-232, August 2008.

4. Masaki Murata, Tamotsu Shirado, Kentaro Torisawa, Masakazu Iwatate, Koji Ichii, Qing Ma, Toshiyuki Kanamaru, Sachiyo Tsukawaki and Hitoshi Isahara. Sophisticated Text Mining System for Extracting and Visualizing Numerical and Named Entity Information from a Large Number of Documents. The 7th NTCIR Workshop, pages 555-562, December 2008.

5. Masakazu Iwatate, Masayuki Asahara and Yuji Matsumoto. Partial Accuracy Evaluation of Dependency Parsers and Its Application (in Japanese). IPSJ SIG Technical Report, 2009-FI-93 and 2009-NL-189, pages 41-48, January 2009.

6. Asad Habib, Masakazu Iwatate, Masayuki Asahara and Yuji Matsumoto. Different Input Systems for Different Devices. Workshop on Advances in Text Input Methods, pages 26-30, November 2011.

7. Asad Habib, Masakazu Iwatate, Masayuki Asahara and Yuji Matsumoto. Keypads for Large Letter-Set Languages and Small Touch-Screen Devices. IPSJ SIG Technical Report, 2011-NL-204, No. 7, pages 1-7, November 2011.

8. Akihiro Inokuchi, Ayumu Yamaoka, Takashi Washio, Yuji Matsumoto, Masayuki Asahara, Masakazu Iwatate and Hideto Kazawa. Mining Rules to Rewrite States in a Dependency Parser (in Japanese). In Proceedings of the First Workshop on Data Oriented Constructive Mining and Simulation. SIG-DOCMAS-B101-5, December 2011.

9. Masakazu Iwatate, Masayuki Asahara and Yuji Matsumoto. Constraints Derived from Coordinated Structure Annotation Make Dependency Structure Annotation More Efficient (in Japanese). IPSJ SIG Technical Report, 2012-IFAT-105 and 2012-NL-205, No. 6, pages 1-7, January 2012.

Awards

1. Outstanding Paper Award, the 14th Annual Meeting of the Association for Natural Language Processing. Japanese Dependency Parsing Using a Tournament Model (in Japanese). In Proceedings of the 14th Annual Meeting of the Association for Natural Language Processing, pages 237–240, March 2008.

2. Best Student Award. Nara Institute of Science and Technology, March 2009.