Text Structure Analysis Methods and Application to Paraphrase Identification

by

Ngo Xuan Bach

submitted to
Japan Advanced Institute of Science and Technology
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Supervisor: Professor Akira Shimazu

School of Information Science
Japan Advanced Institute of Science and Technology

March, 2014

To my family

Abstract

Analyzing structures of texts is important to understand natural language, both general texts and texts in some specific domains such as the legal domain. For general texts, discourse structures have been shown to have an important role in many natural language processing applications, including text summarization, question answering, information presentation, dialogue generation, and paraphrase extraction. In the legal domain, where legal texts have their own specific characteristics, recognizing logical structures in legal texts does not only help people in understanding legal documents, but also to support other tasks in legal text processing.

In this thesis, we study the structures of texts based on relations between discourse units. Regarding relations between discourse units, we focus on general semantic relations and on logical relations, which are appropriate in some cases such as laws. For general semantic relations, we study a model based on Rhetorical Structure Theory (RST). For logical relations, we study a model for legal paragraphs. Both models are based on the same framework, which consists of two steps, *Recognizing discourse units of texts* and *Building structures of texts from the discourse units*.

In our work on learning discourse structures, we propose an Unlabeled Discourse parsing system in the RST framework (UDRST). UDRST consists of a segmentation model and a parsing model. Our segmentation model exploits subtree features to rerank the N-best outputs of a base segmenter, which uses syntactic and lexical features in a Conditional Random Field (CRF) framework. The advantage of our model is that subtree features are long distance non-local features which can capture whole discourse units. In the parsing model, we introduce an incremental algorithm for building discourse trees. The algorithm builds a discourse tree for each sentence, then for each paragraph, and finally for the whole text. We also propose a new algorithm that exploits the dual decomposition method to combine a greedy model and the incremental model. Our system achieves state-of-the-art results on both the discourse segmentation task and the unlabeled discourse parsing task on the RST Discourse Treebank corpus.

Concerning our study on analyzing logical structures of legal texts, we propose a two-phase framework for analyzing logical structures of legal paragraphs. In the first phase, we model the problem of recognizing logical parts in law sentences as a multi-layer sequence learning problem, and present a CRF-based model to recognize them. In the second phase, we propose a graph-based method to group logical parts into logical structures. We consider the problem of finding a subset of complete subgraphs in a weighted-edge complete graph, where each node corresponds to a logical part, and a complete subgraph corresponds to a logical structure. We propose an integer linear programming formulation for this optimization problem. We also introduce an annotated corpus for the task, the Japanese National Pension Law corpus, and describe our experiments on that corpus.

We then study how to exploit discourse structures for identifying paraphrases. By analyzing paraphrase sentences, we found that discourse units are very important for paraphrasing. In many cases, a paraphrase sentence can be created by applying several operations to the original sentence. Motivated by the analysis of the relation between paraphrases and discourse units, we propose a new method to compute the similarity between two sentences. Unlike conventional methods, which directly compute similarities based on sentences, our method divides sentences into discourse units and employs them to compute similarities. We apply our method to the paraphrase identification task. Experimental results on the PAN corpus, a large corpus for detecting paraphrases, show the effectiveness of using discourse information for identifying paraphrases.

Keywords: Text Structure Analysis, Legal Text Processing, Discourse Structure, Rhetorical Structure Theory, Logical Structure, Paraphrase Identification, Discourse Unit, Text Similarity, Conditional Random Fields, Support Vector Machines.

Acknowledgments

My last six years at the Japan Advanced Institute of Science and Technology (JAIST) have been an incredible journey, where I have grown tremendously, both academically and personally. None of this would have been possible without the support of many people.

First and foremost, I would like to sincerely thank my great supervisor, Professor Akira Shimazu of the School of Information Science at JAIST, who brought me to the interesting world of natural language processing and supported me during my study. Without his guidance and encouragement, my work could not have been accomplished. Professor Shimazu taught me not only how to become a good researcher, but also how to become a kind person. I am really lucky to be one of his students.

Next, I thank my terrific Committee, consisting of Professor Takenobu Tokunaga from the Graduate School of Information Science and Engineering at the Tokyo Institute of Technology, Professor Satoshi Tojo, Professor Ho Tu Bao, and Associate Professor Kiyoaki Shirai at JAIST, for the time they spent reading my thesis and for their valuable comments. My thesis is improved very much through their comments.

I wish to express my deep thanks to the Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT) and the MEXT Scholarship Program. Without their financial support, I would not have come to Japan to study.

I would like to thank Associate Professor Kiyoaki Shirai of the School of Information Science at JAIST, who supported me during my study in the natural language processing laboratory (Shimazu-Shirai Lab). Professor Shirai also gave me valuable comments during the seminars of the lab.

I would like to sincerely thank Professor Kunihiko Hiraishi of the School of Information Science at JAIST, for his guidance and support during my subtheme research on dual decomposition and its application to part-of-speech tagging.

I also would like to thank Associate Professor Nguyen Le Minh of the School of Information Science at JAIST, who always listened to me, discussed with me, and supported me. I learned a lot from him, not only how to conduct good research, but also how to solve technical problems.

I also would like to thank Mr. Kenji Takano and Mr. Yoshiko Oyama for analyzing law sentences and building the Japanese National Pension Law corpus, a very important part of my research.

I wish to send my deep acknowledgments to "The 21st Century COE program Verifiable and Evolvable e-Society", "The Grant-in-Aid for Scientific Research, Education and Research Center for Trustworthy e-Society", "The JAIST Overseas Training Program for 3D Program Students", "The JAIST Foundation Research Grant", "The NII Research Grant", and "The NEC C&C Foundation Grant", for supporting me in my research and in attending international conferences.

I would like to express my deep appreciation to the English Language Education for Science, Technology & Engineering center (CELESTE) of JAIST for editing this manuscript and all JAIST staff for their kind and convenient procedures and services.

I am grateful to my colleagues in the natural language processing laboratory (Shimazu-Shirai Lab) for making JAIST an enjoyable place for my study and life.

Finally, I would like to give special thanks to my family for their sacrifice, love, and understanding.

Contents

Abstract **ii**

Acknowledgments **iv**

1 Introduction **1**

1.1 Background 1

1.2 Research Problems and Contributions 3

1.3 Thesis Outline 5

2 Background: Statistical Machine Learning Models **7**

2.1 Maximum Entropy Models 7

2.1.1 The Principle 7

2.1.2 The Models 7

2.2 Support Vector Machines 8

2.3 Conditional Random Fields 11

3 Learning Discourse Structures in the RST Framework **13**

3.1 Introduction 13

3.2 Related Work 14

3.2.1 Related Work on Discourse Segmentation 14

3.2.2 Related Work on Discourse Parsing 15

3.3 A Reranking Model for Discourse Segmentation using Subtree Features . . 16

3.3.1 Why Reranking? 16

3.3.2 Discriminative Reranking 16

3.3.3 Base Model 17

3.3.4 Subtree Features for Reranking 18

3.4 An Incremental Algorithm for Building Discourse Trees 20

3.5 Dual Decomposition for Building Discourse Trees 22

3.5.1 Dual Decomposition 22

3.5.2 A Parsing Algorithm using Dual Decomposition 23

3.6 Experiments 24

3.6.1 Data and Evaluation Methods 24

3.6.2 Experiments on Discourse Segmentation 25

3.6.3 Experiments on Discourse Parsing 28

3.7 Conclusions 29

4 Analyzing Logical Structures of Legal Texts 31
4.1 Introduction . . . 31
4.1.1 Logical Parts and Logical Structures of Legal Texts . . . 32
4.1.2 Motivation of This Work . . . 34
4.1.3 Overview of This Chapter . . . 35
4.2 Related Work . . . 36
4.2.1 Studies on Legal Text Processing . . . 36
4.2.2 Studies on Analyzing Logical Structures of Japanese Legal Texts . . . 38
4.3 Task Formulation . . . 39
4.3.1 Subtask 1: Recognition of Logical Parts . . . 40
4.3.2 Subtask 2: Recognition of Logical Structures . . . 41
4.4 Framework Architecture . . . 42
4.5 Multi-layer Sequence Learning for Logical Part Recognition . . . 42
4.6 ILP for Recognizing Logical Structures . . . 45
4.6.1 ILP Formulation . . . 45
4.6.2 Learning Binary Classifier . . . 47
4.7 Experiments . . . 49
4.7.1 Corpus . . . 49
4.7.2 Evaluation Methods . . . 50
4.7.3 Experiments on Subtask 1 . . . 52
4.7.4 Experiments on Subtask 2 . . . 55
4.7.5 Limitation & Improvement . . . 59
4.8 Conclusions . . . 63

5 Exploiting Discourse Information to Identify Paraphrases 65
5.1 Introduction . . . 65
5.2 Related Work . . . 66
5.3 Paraphrases and Discourse Units . . . 67
5.4 EDU-Based Similarity . . . 69
5.5 Experiments . . . 70
5.5.1 Data & Evaluation Method . . . 70
5.5.2 MT Metrics . . . 72
5.5.3 Experimental Results with a Single SVM Classifier . . . 73
5.5.4 Revision Learning & Voting . . . 74
5.5.5 Error Analysis . . . 76
5.6 Conclusions . . . 78

6 Conclusions and Future Work 80
6.1 Summary of the Thesis . . . 80
6.2 Future Directions . . . 81

A LSDemo: A Demonstration System for Analyzing Logical Structures of Paragraphs in Legal Articles 83
A.1 Requirements . . . 83
A.2 Graphical User Interface . . . 84

References 86

Publications **98**

List of Figures

1.1 A general framework for text structure analysis. 2
1.2 Illustration of our work in this thesis. 3

2.1 Large margin and small margin in SVMs. 9
2.2 Calculation of the margin in the SVM framework. 10
2.3 SVM framework in non-separable cases. 10
2.4 Graphical model of a CRF for sequence learning (n is the length of sequence). 11

3.1 A discourse tree [123]. 14
3.2 Examples of segmenting sentences into EDUs. 17
3.3 Partial lexicalized syntactic parse trees. 18
3.4 Subtree features. 19
3.5 An example sentence for illustrating two evaluation methods. 24
3.6 Some errors made by our model. 28

4.1 Examples of law sentences and logical parts (A: antecedent part; C: consequent part; T: topic part). 33
4.2 Four popular cases of legal paragraphs [130]. 34
4.3 An example of Individual type in the JNPL corpus (A means *Antecedent part*; C means *Consequent part*; T means *Topic part*. 34
4.4 Two cases of inputs and outputs of the task. 39
4.5 An example of *overlapping* and *embedded* relationships. 40
4.6 Framework architecture. 43
4.7 A law sentence with logical parts in three layers. 44
4.8 An example of labeling in the multi-layer model. 44
4.9 Examples of graphs and their cliques. 45
4.10 The structure of JNPL. 49
4.11 Relationship between a sentence and logical parts. 50
4.12 An annotated sentence in the JNPL corpus. (left: the original text; right: the translated text.) 51
4.13 An example of the first case. 56
4.14 An example of the second case. 58
4.15 List of cue phrases. 62

5.1 A paraphrase sentence pair in the PAN corpus [80]. 68
5.2 Another paraphrase sentence pair in the PAN corpus. 68
5.3 A revision learning model for the paraphrase identification task. 75

A.1 Graphical user interface of LSDemo. 84

A.2 A running example with a paragraph consisting of one sentence. 85
A.3 A running example with a paragraph containing multiple sentences. 85

List of Tables

3.1 Statistical information of the RST Discourse Treebank corpus 24
3.2 Performance when evaluating on B labels 26
3.3 Performance when evaluating on B and C labels 26
3.4 Top 30 subtree features with the highest weights 27
3.5 Top error words 28
3.6 Most frequent words that appear before error words 29
3.7 Experimental results of the tree building step (gold segmentation and gold parse trees) 29
3.8 Performance of the full system (our segmentation model and Stanford parse trees) 30

4.1 Features for learning binary classifier (T2: topic part in case 2; A: antecedent part; C: consequent part) 48
4.2 Statistics on logical parts of the JNPL corpus 50
4.3 Examples of evaluation method for Subtask 2 51
4.4 Examples of evaluation method for the whole system (PreP = Predicted logical parts, PreStr = Predicted logical structures, GP = Gold logical parts, GStr = Gold logical structures) 52
4.5 Experimental results of the baseline model 54
4.6 Experimental results for Subtask 1 on the JNLP corpus(W:Word; P: POS tag; B: Bunsetsu tag) 55
4.7 Experimental results in more details 55
4.8 Experiments on Subtask 2 (Gold-Input Setting) 58
4.9 Experiments on Subtask 2 (Whole-Task Setting) 59
4.10 Experimental results on Subtask 1 using Joint Decoding Algorithm (W:Word; P: POS tag; B: Bunsetsu tag). N = 1, 2, 3, 4, 5, 10, 20, 50, 100, and 1000. 60
4.11 Oracle results on Subtask 1 (the feature set consists of words and Bunsetsu tags) 61
4.12 Experimental results with context features 62
4.13 Experimental results with heuristic rules for post-processing 63

5.1 An example of computing sentence-based and EDU-based similarities . . . 71
5.2 PAN corpus for paraphrase identification 72
5.3 Experimental results on each individual MT metric 73
5.4 Experimental results on combined MT metrics 74
5.5 Experimental results on long and short sentences 74
5.6 Experimental results of the revision learning model and the voting model . 76
5.7 Statistic information of the experimental results on the test set 77

Chapter 1

Introduction

1.1 Background

Analyzing structures of texts is important to understand natural language, both general texts and texts in some specific domains such as the legal domain. In many natural language processing applications such as text summarization, machine translation, and paraphrase generation, systems should produce an output text that is not only informative but also coherent and readable. To do that, computers need knowledge about structures of texts or how information is structured in texts.

Research on text structure analysis has been a focus in natural language processing for a long time. Several linguistic theories of text structures have been proposed, including Scripts, the Centering theory, and the Rhetorical Structure Theory (RST).

1. **Scripts**
   The idea of Scripts [1] is that memory is organized as a collection of "scripts". A script describes a sequence of events commonly encountered in some particular situations. Abelson and Schank [1] present an example of script, the restaurant script: "*someone enters a restaurant, finds a table, moves to that table, then orders food, eats, pays money, and leaves*". Abelson and Schank [1] suggest that such a sequence of events is frequently encountered and therefore incorporated into our memory. Such knowledge helps us to recognize related and coherent texts in different situations. Abelson and Schank [1] also explain that the sentences below are not meaningful at all because they describe events that do not conform to a common situation:

   "*John was walking on the street. He thought of cabbages. He picked up a shoe horn.*"

2. **Centering theory**
   The idea of the Centering theory [48] is to use the property of entity repetition to describe local coherence in texts. According to this theory, adjacent sentences in a discourse often share entities and therefore focus and topic in the text are maintained. The Centering theory also states that some specific patterns of entity sharing are preferred to others in order to achieve coherence. In a coherent text, the more salient entities in a sentence are assumed to be more likely to appear in subsequent sentences. Centering proposes that such preferred coreference patterns make the text coherent for the reader.

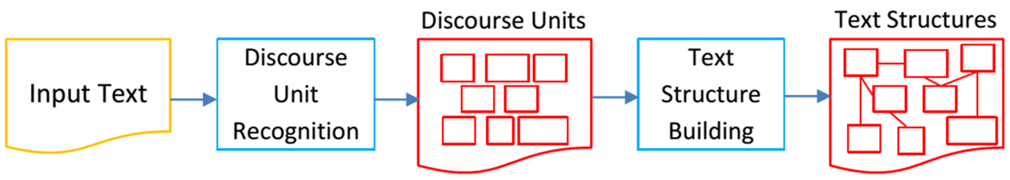


Figure 1.1: A general framework for text structure analysis.

3. **Rhetorical Structure Theory**
   The Rhetorical Structure Theory or RST [81] is based on theories of rhetorical or discourse relations in which clauses or sentences in coherent texts are connected by semantic relations such as cause, evidence, and contrast. The RST assumes that a text can be divided into several elementary discourse units (EDUs) which are connected by such semantic relations to form a discourse tree.

Discourse structures have been shown to have an important role in many natural language processing applications, including text summarization [79, 82], question answering [126], information presentation [10], dialogue generation [52], and paraphrase extraction [110]. In the legal domain, where legal texts have their own specific characteristics, recognizing logical structures in legal texts does not only help people in understanding legal documents, but also support other tasks in legal text processing [63].

Text structure analysis, therefore, is an important task in the field of computational linguistics. From the linguistic point of view, the task contributes to understanding how information is organized in written texts as well as how coherent texts should be. On the other hand, from the computational point of view, the task plays a key role in building robust natural language processing applications.

In this thesis, we study the structures of texts based on relations between discourse units. Regarding relations between discourse units, we focus on general semantic relations and on logical relations, which are appropriate in some cases such as laws. For general semantic relations, we study a model based on Rhetorical Structure Theory (RST). For logical relations, we study a model for legal paragraphs. Both models are based on the same framework (illustrated in Figure 1.1), which consists of two steps as follows:

1. **Discourse Unit Recognition**: Recognizing discourse units of texts, elementary discourse units (EDUs) in the case of RST discourse structures and logical parts in the case of logical structures of legal texts

2. **Text Structure Building**: Building structures of texts from the discourse units, RST discourse structures and logical structures of legal texts.

We also study how to utilize structures of texts for paraphrase computation. Paraphrases are phrases, sentences, or longer expressions with the same or very similar meanings. Paraphrases have been shown to play an important role in many natural language processing applications, including text summarization [9], question answering [39], machine translation [19], and plagiarism detection [139]. Specifically, we consider the problem of using RST discourse structures to identify paraphrases, which determines whether two given sentences have the same meaning. Figure 1.2 illustrates our work in this thesis.

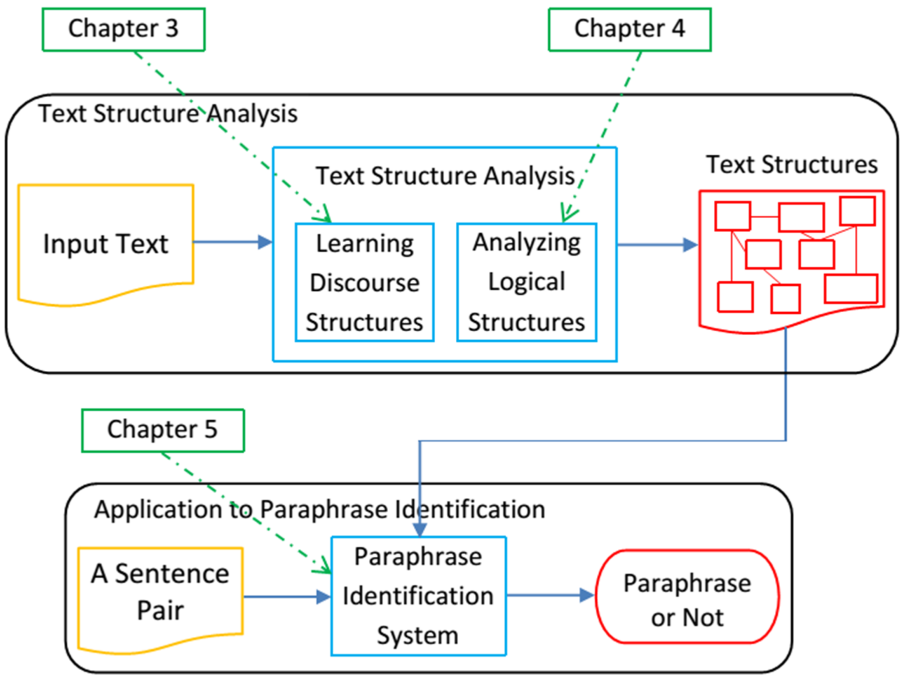


Figure 1.2: Illustration of our work in this thesis.

1.2 Research Problems and Contributions

This thesis focuses on the structural analysis of texts based on machine learning and its application to paraphrase identification. Statistical machine learning models have been widely applied to various fields in computer science, including computer vision, bioinformatics, chemical informatics, robotics, computer games, and natural language processing (NLP). They also have been claimed to have the potential to amplify every aspect of a working scientist's progress to understanding [89]. In the NLP field, statistical machine learning models have been applied successfully to various problems, ranging from fundamental tasks such as part-of-speech tagging [75, 109], chunking [67, 71, 119], named entity recognition [14, 118], syntactic and semantic parsing [97, 151], discourse parsing [54], and paraphrase identification [80] to applications such as machine translation [25, 101], text summarization [96], and question answering [41, 128].

We concentrate on three problems as follows:

1. **Learning Discourse Structures in the RST Framework**
   Rhetorical Structure Theory (RST) [81] is one of the most widely used theories of text structures. In the RST framework, a text is first divided into several elementary discourse units (EDUs). Consecutive EDUs are then put in relation with each other to build a discourse tree. This is a difficult but very important task in the field of natural language processing.

   Previous studies on learning RST discourse structures reveal two remarkable points. The first point is that the sets of rhetorical relations in different works are inconsistent. The second point is that the performance of a state-of-the-art discourse parsing system is too low when evaluating in the labeled score. Studies on applications of RST show that in many text analysis applications, only a few relations are enough [82, 153]. The purpose of our research is to build an unlabeled discourse parsing system, which produces a discourse structure tree without relation labels. The number of relations, types of relations, and the process of labeling relations for the discourse structure tree will be implemented in text analysis applications. Our contributions are as follows:

- We introduce a new model for segmenting texts into elementary discourse units. Unlike previous models, which only focus on boundaries of EDUs, our model employs subtree features in a discriminative re-ranking framework, which can capture long-distance non-local features which describe whole EDUs.
- We propose a new model for building discourse trees based on the dual decomposition technique, which allows us to integrate two base models with different characteristics to yield a more powerful model.
- Our system achieves state-of-the-art results on both the discourse segmentation task and the unlabeled discourse parsing task on the RST Discourse Treebank corpus.

2. **Analyzing Logical Structures of Legal Texts**
Analyzing logical structures of legal texts is an important task in Legal Engineering [62, 64]. The outputs of this task will be beneficial to people in understanding legal texts. This task is the preliminary step, which supports other tasks in legal text processing (translating legal articles into logical and formal representations, legal text summarization, legal text translation, question answering in legal domains, etc) and serves legal text verification, an important goal of Legal Engineering.

Analyzing law sentences individually is not sufficient to understand a legal document. In most cases, to understand a law sentence, we need to understand related sentences or its context. To our knowledge, however, no existing research addresses the task at the paragraph level. Analyzing logical structures of legal paragraphs is therefore very important in the research of Legal Engineering to achieve its goals: assisting people in understanding legal texts and making computers able to process legal texts automatically. Our original idea is to propose a novel learning framework that can integrate graphical model, linear programming, and machine learning for understanding legal texts. Our contributions are as follows:

- We introduce a new task for legal text processing, analyzing logical structures of paragraphs in legal articles.
- We propose a two-phase framework to solve the task: a multi-layer sequence learning model for recognizing logical parts and a graph-based model with integer linear programming for recognizing logical structures.
- We introduce an annotated corpus for the task, the Japanese National Pension Law corpus.
- We evaluate our framework on that corpus.

Our work shows promising results for further research on this interesting task.

3. **Exploiting Discourse Information to Identify Paraphrases**
   Previous studies have shown that discourse structures deliver important information for paraphrase computation. However, such studies only consider some special kinds of data, for which the discourse structures can be easily extracted. The goal of our research is to exploit discourse structures for computing paraphrases in general texts. Our contributions are as follows:

   - We show the relation between discourse units and paraphrasing, in which discourse units play an important role in paraphrasing. We also show that, in many cases, paraphrase sentences can be generated from the original sentences by performing several transformations
   - We propose a new method to compute text similarity based on elementary discourse units. Our method is general and not limited to any kind of text.
   - We apply the method to the task of paraphrase identification.
   - We achieve state-of-the-art results on the PAN corpus, a large corpus for detecting paraphrase. Experimental results show that discourse information is effective for the task of identifying paraphrases

   To the best of our knowledge, this is the first work that employs discourse units for computing similarity as well as for identifying paraphrases.

1.3 Thesis Outline

This dissertation is structured as follows. Chapter 2 gives a brief introduction to several statistical machine learning methods, including Maximum Entropy Models (MEMs), Support Vector Machines (SVMs), and Conditional Random Fields (CRFs).

Chapter 3 presents our study on learning discourse structures in the RST framework. We first describe related work on discourse segmentation and discourse parsing in the RST framework. We then introduce our discourse parsing system, including a reranking model for segmenting a text into elementary discourse units and a dual decomposition model for building discourse trees from discourse units. We also describe our experimental results on the RST Discourse Treebank corpus and compare them with the results of previous research.

In Chapter 4, we investigate the task of analyzing logical structures of legal paragraphs. We first describe related work on legal text processing. We then present how to formulate the task of analyzing logical structures of legal paragraphs. We next introduce our proposed two-phase framework for the task, which recognizes logical parts in the first phase and logical structures in the second phase. We present experimental results on an annotated corpus, the Japanese National Pension Law corpus. A preliminary study on exploiting heuristic rules in the post-processing step to improve the performance of the system is also described.

Chapter 5 presents our study on exploiting discourse information to identify paraphrases. We first describe related work on paraphrase identification. We next introduce our findings on the relation between discourse units and paraphrasing. We then present a new method, EDU-based similarity, for computing text similarity. We also describe our experiments on the PAN corpus to show that EDU-based similarity is effective for identifying paraphrases.

Finally, Chapter 6 summarizes this thesis and discusses future directions.

Chapter 2

Background: Statistical Machine Learning Models

In this chapter, we give a brief introduction to three common statistical machine learning models that will be employed in this thesis, including Maximum Entropy Models (Section 2.1), Support Vector Machines (Section 2.2), and Conditional Random Fields (Section 2.3).

2.1 Maximum Entropy Models

2.1.1 The Principle

The principle of Maximum Entropy states that equal probabilities must be assigned to each competing assertion if there is no positive reason for assigning them different probabilities.

Jaynes [58] summarizes the principle of Maximum Entropy as follows:

"*...in making inferences on the basic of partial information we must use that probability distribution which has maximum entropy subject to whatever is known. That is the only unbiased assignment we can make; to use any other would amount to arbitrary assumption of information which by hypothesis we do not have.*"

2.1.2 The Models

Maximum Entropy Models (MEMs) [12, 109] are a method of estimating the conditional probability $p(y|x)$ that a model outputs a label y given a context x:

$$p(y|x) = \frac{1}{Z(x)} exp(\sum_i \lambda_i f_i(x, y))$$

where $f_i(x, y)$ refers to a feature function; λ_i is a parameter of the model; and $Z(x)$ is a normalization factor. To capture statistic information, this method requires that the model accord with some constraints which have the form:

$$p(f) = \tilde{p}(f).$$

In this formula, f is a feature function (or feature for short), which takes a pair (x, y) as input and outputs a real value. Usually, f is a binary-value indicator function. $p(f)$

and $\tilde{p}(f)$ are the expected values of f with respect to the model $p(y|x)$ and the empirical distribution $\tilde{p}(x,y)$, respectively. They are defined as follows:

$$p(f) \equiv \sum_{x,y} \tilde{p}(x)p(y|x)f(x,y),$$

$$\tilde{p}(f) \equiv \sum_{x,y} \tilde{p}(x,y)f(x,y),$$

where $\tilde{p}(x)$ is the empirical distribution of x in the training samples.

Suppose that we have n feature functions $f_i(i = 1, 2, \ldots, n)$ and we want our model to accord with these statistics. Our model will belong to a subset Q of P (the set of all conditional probability distributions) defined by

$$Q \equiv \{p \in P | p(f_i) = \tilde{p}(f_i), i = 1, 2, \ldots, n\}.$$

The maximum entropy method chooses the model $p^* \in Q$ that maximizes the entropy function $H(p)$:

$$p^* = \text{argmax}_{p \in Q} H(p)$$

where the entropy function $H(p)$ is defined as follows:

$$H(p) \equiv -\sum_{x,y} \tilde{p}(x)p(y|x) \log p(y|x).$$

To solve the constrained optimization problem, we first convert the primal problem to a dual optimization problem using the method of Lagrange multipliers [12]. Then the solution of the dual optimization problem can be found by applying the improved iterative scaling method [12, 33] or LBFGS method [100].

Maximum entropy model has been applied successfully to many NLP task including POS tagging [109], chunking [67], syntactic and semantic dependency parsing [151], statistical machine translation [12, 40], and so on.

2.2 Support Vector Machines

Support Vector Machines (SVMs) are a statistical machine learning technique proposed by Vapnik et al. [15, 31, 91, 140]. SVMs have been demonstrated their performance on a number of problems in areas, including computer vision, handwriting recognition, pattern recognition, and statistical natural language processing. In the field of natural language processing, SVMs have been applied to text categorization [59], word sense disambiguation [77], text chunking [71], syntactic parsing [99], semantic parsing [97], discourse parsing [54], machine translation [150], topic classification [147], information extraction [13], sentiment analysis [107, 114], and so on, and achieved very good results.

We start with the binary classification task in linear cases, in which we want to find a hyperplane that separates positive and negative samples. Suppose that we have a set of n training samples:

$$S = \{(x_i, y_i)\}_{i=1}^{n}, x_i \in R^m, y_i \in \{+1, -1\},$$

where x_i is the feature vector and y_i is the class (or label) of the i^{th} sample. Our goal is to separate the positive and negative samples by a hyperplane in the form:

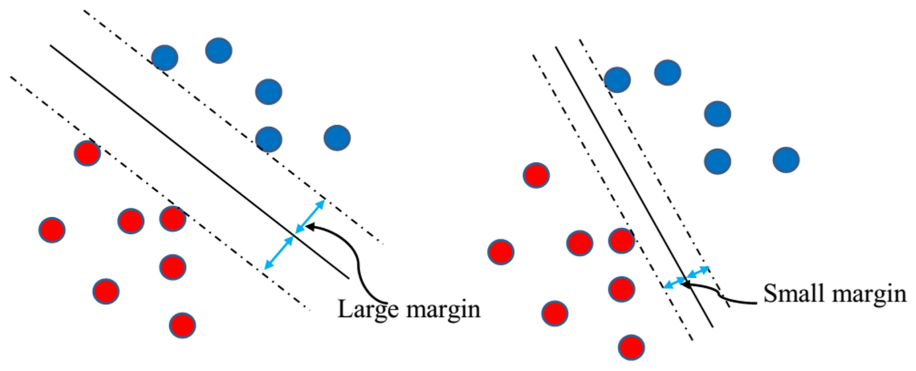


Figure 2.1: Large margin and small margin in SVMs.

$$w \cdot x + b = 0,$$

where $w \in R^m$ and $b \in R$ are parameters.

Among the set of all possible hyperplanes, SVMs will find an *optimal* hyperplane (correspond to find an optimal parameter set for w and b). In the SVM framework, the *optimal* hyperplane is the hyperplane with maximal margin between training samples and the hyperplane. Figure 2.1 illustrates this strategy. Solid lines show two possible hyperplanes (or candidates). Each candidate separates correctly the training samples into two classes. Two dashed lines parallel to a candidate indicate the boundaries in which the candidate can be moved without any misclassification.

Suppose that the training samples satisfy the following constraints:

$$w \cdot x_i + b \geq +1 \text{ for } y_i = +1,$$

$$w \cdot x_i + b \leq -1 \text{ for } y_i = -1.$$

These constraints can be combined into the following inequalities:

$$y_i(w \cdot x_i + b) - 1 \geq 0, \forall i = 1, 2, \ldots, n.$$

Figure 2.2 shows how to calculate the margin. We have, the perpendicular distance from the origin to the solid line ($w \cdot x + b = 0$) is $\frac{|b|}{\|w\|}$, where $\|w\|$ is the Euclidean norm of w. Similarly, the perpendicular distances from the origin to two dashed lines ($w \cdot x + b = 1$ and $w \cdot x + b = -1$) are $\frac{|b-1|}{\|w\|}$ and $\frac{|b+1|}{\|w\|}$.

Let d_+ and d_- be the distances between the solid lines and two dashed lines. We will have the margin M:

$$M = d_+ = d_- = \frac{1}{\|w\|}.$$

To maximize the margin M, we minimize $\|w\|$. The task now becomes solving the following optimization problem:

Minimize:

$$L(w) = \frac{1}{2}\left\|w\right\|^2$$

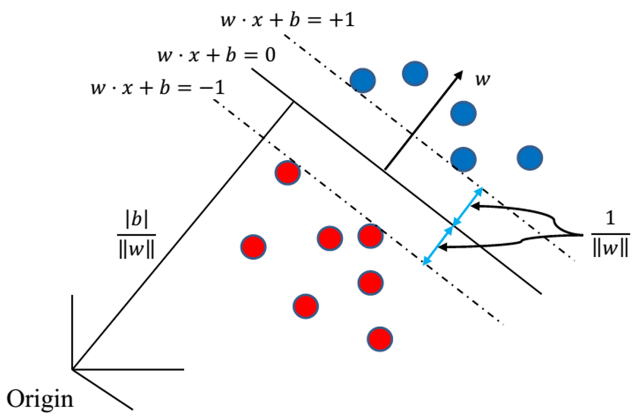


Figure 2.2: Calculation of the margin in the SVM framework.

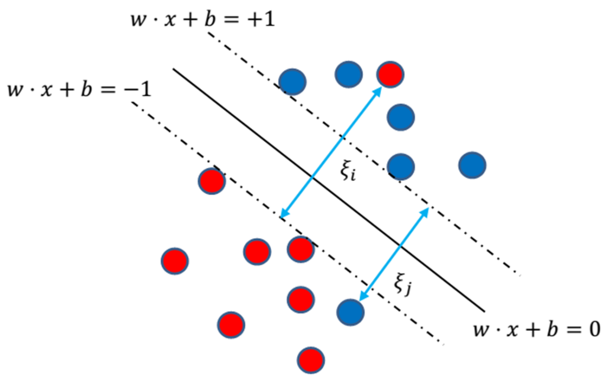


Figure 2.3: SVM framework in non-separable cases.

Subject to:

$$y_i(w \cdot x_i + b) - 1 \geq 0, \forall i = 1, 2, \ldots, n. \tag{2.1}$$

The training samples which lie on two dashed lines are called support vectors.

We now move to the non-separable cases, in which the training data are not linearly separable. In such cases, for any hyperplane $w \cdot x + b = 0$, there exists $x_i \in S$ such that:

$$y_i(w \cdot x_i + b) - 1 < 0.$$

Therefore, the constraints in Equation 2.1 cannot all hold simultaneously. We use a relaxed version of these constraints by introducing slack variables $\xi_i (1 \leq i \leq n)$ as follows: For each $i \in \{1, 2, \ldots, n\}$, there exists $\xi_i \geq 0$ such that:

$$y_i(w \cdot x_i + b) - 1 + \xi_i \geq 0.$$

The slack variable ξ_i measures the distance that the vector x_i violates the constraint $y_i(w \cdot x_i + b) - 1 \geq 0$. The SVM framework in the non-separable cases is illustrated in Figure 2.3.

We want to find a hyperplane that:

1. Limits the total amount of slack, i.e., $\sum_{i=1}^{m} \xi_i^p$, here $p \geq 1$ is a constant, and

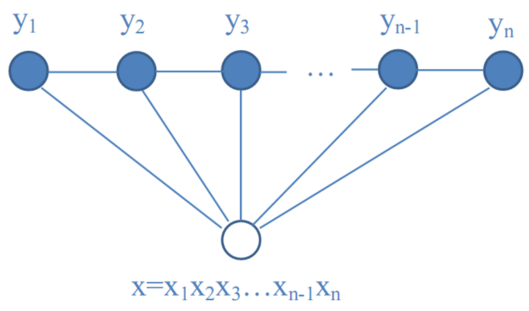


Figure 2.4: Graphical model of a CRF for sequence learning (n is the length of sequence).

2. Has a large margin.

The task now becomes solving the new optimization problem as follows:
Minimize:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i^p$$

Subject to:

$$y_i(w \cdot x_i + b) - 1 + \xi_i \geq 0 \text{ and } \xi_i \geq 0, \forall i = 1, 2, \dots, n.$$

Here, $C \geq 0$ is a parameter which determines the trade-off between the maximization of the margin and the minimization of the slack penalty.

2.3 Conditional Random Fields

Conditional Random Fields (CRFs) [75, 129] are undirected graphical models (see Figure 2.4), which define the probability of a label sequence y given an observation sequence x as a normalized product of potential functions. Each potential function has the following form:

$$exp(\sum_j \lambda_j t_j(y_{i-1}, y_i, x, i) + \sum_k \mu_k s_k(y_i, x, i))$$

where $t_j(y_{i-1}, y_i, x, i)$ is a transition feature function (or edge feature), which is defined on the entire observation sequence x and the labels at positions i and $i-1$ in the label sequence y; $s_k(y_i, x, i)$ is a state feature function (or node feature), which is defined on the entire observation sequence x and the label at position i in the label sequence y; and λ_j and μ_k are parameters of the model, which are estimated in the training process. Each feature function (edge feature and node feature) takes a real value.

The probability of a label sequence y given an observation sequence x then can be defined as follows:

$$p(y|x, \lambda, \mu) = \frac{1}{Z(x)} exp(\sum_j \lambda_j t_j(y_{i-1}, y_i, x, i) + \sum_k \mu_k s_k(y_i, x, i))$$

where $Z(x)$ is a normalization factor,

$$Z(x) = \sum_{y} exp(\sum_{j} \lambda_j t_j(y_{i-1}, y_i, x, i) + \sum_{k} \mu_k s_k(y_i, x, i)).$$

Training CRFs is commonly performed by maximizing the likelihood function with respect to the training data using advanced convex optimization techniques like L-BFGS [18]. And inference in CRFs, i.e., searching the most likely output label sequence of an input observation sequence, can be done by using Viterbi algorithm [45].

CRFs have been shown to be an efficient and powerful framework for sequence learning tasks. CRFs have all the advantages of Maximum Entropy Markov Models (MEMMs) [85] but do not suffer from the label bias problem [75]. CRFs have been applied successfully to many NLP tasks such as POS tagging, chunking, named entity recognition, syntax parsing, information retrieval, information extraction, analyzing logical structures of legal texts at the sentence level, and so on [4, 73, 75, 106, 119].

Chapter 3

Learning Discourse Structures in the RST Framework

This chapter presents our study on learning discourse structures in the RST framework, one of the most widely used theories of text structures. We first give an introduction to the discourse parsing task and motivation of our work (Section 3.1). We then describe related work on discourse parsing (Section 3.2). We next present our reranking model for discourse segmentation (Section 3.3) and two algorithms for building discourse trees, including an incremental algorithm (Section 3.4) and a dual decomposition algorithm (Section 3.5). Finally, we describe our experiments on the RST Discourse Treebank corpus (Section 3.6) and conclude this chapter (Section 3.7).

3.1 Introduction

Discourse structures have been shown to have an important role in many natural language processing applications, such as text summarization [79, 82], information presentation [10], question answering [126], and dialogue generation [52]. To produce such kinds of discourse structures, several attempts have been made to build discourse parsers in the framework of Rhetorical Structure Theory (RST) [81], one of the most widely used theories of text structures.

The discourse parsing task in the RST framework consists of two steps: *discourse segmentation* and *discourse tree building*. In the discourse segmentation step, an input text is divided into several elementary discourse units (EDUs). Each EDU may be a simple sentence or a clause in a complex sentence. In the tree building step, consecutive EDUs are put in relation with each other to create a discourse tree. Figure 3.1 shows an example of a discourse tree with three EDUs.

The quality of the discourse segmenter contributes a significant part to the overall accuracy of every discourse parsing system. If a text is wrongly segmented, no discourse parsing algorithm can build a correct discourse tree. Existing discourse segmenters usually exploit lexical and syntactic features to label each word in a sentence with one of two labels, *boundary* or *no-boundary*. The limitation of this approach is that it only focuses on the boundaries of EDUs. It cannot capture features that describe whole EDUs.

Recently, discriminative reranking has been used successfully in some NLP tasks such as part-of-speech (POS) tagging, chunking, and statistical parsing [29, 46, 57, 74]. The

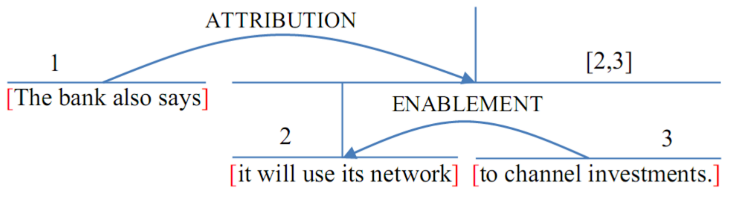


Figure 3.1: A discourse tree [123].

advantage of the reranking method is that it can exploit the output of a base model to learn. Based on that output, we can extract long-distance non-local features to rerank.

The first purpose of our study is to build a discourse segmentation system, which can capture non-local features describing whole EDUs. Our discourse segmenter exploits subtree features to rerank the N-best outputs of a base segmenter, which uses syntactic and lexical features in a CRF framework. Experimental results on the RST Discourse Treebank corpus show that our model outperforms existing discourse segmenters in both settings that use gold standard Penn Treebank parse trees [83] and Stanford parse trees [66].

Previous studies on discourse parsing reveal two remarkable points. The first point is that the sets of rhetorical relations in different works are inconsistent. For instance, Marcu [82] and Thanh et al. [135] use 15 relations and 14 relations respectively, while Sagae [116] and Hernault et al. [54] use 18 relations. We also note that in RST Discourse Treebank (RST-DT) [21], 78 rhetorical relations are used. The second point is that the performance of a state-of-the-art discourse parsing system is too low when evaluating in the labeled score (both structure and relations). HILDA [54], a state-of-the-art discourse parsing system, achieves only 47.3% in the labeled score on RST-DT. Studies on applications of RST show that in many text analysis applications, only a few relations are enough [82, 153]. Furthermore, from a machine learning perspective, working with a small set of relations can improve the performance of a discourse parsing system.

The second purpose of our study is to build an unlabeled discourse parsing system, which produces a discourse structure tree without relation labels. The number of relations, types of relations, and the process of labeling relations for the discourse structure tree will be implemented in text analysis applications, which use this discourse parser. Our discourse parsing system, UDRST, consists of a reranking-based segmentation model and a parsing model using dual decomposition. Our system achieves 77.3% in the unlabeled score on the standard test set of the RST Discourse Treebank corpus, which improves 5.0% compared to HILDA [54], a state-of-the-art discourse parsing system.

3.2 Related Work

3.2.1 Related Work on Discourse Segmentation

Several methods have been proposed to deal with the discourse segmentation task. Thanh et al. [134] present a rule-based discourse segmenter with two steps. In the first step, segmentation is done by using syntactic relations between words. The segmentation algorithm is based on some principles, which have been presented in [30] and [20], as follows:

1. *The clause that is attached to a noun phrase can be recognised as an embedded unit. If the clause is a subordinate clause, it must contain more than one word.*

2. *Coordinate clauses and coordinate sentences of a complex sentence are EDUs.*

3. *Coordinate clauses and coordinate elliptical clauses of verb phrases (VPs) are EDUs. Coordinate VPs that share a direct object with the main VP are not considered as a separate discourse segment.*

4. *Clausal complements of reported verbs and cognitive verbs are EDUs.*

The segmenter then uses cue phrases to correct the output of the first step.

Tofiloski et al. [137] describe another rule-based discourse segmenter. The core of this segmenter consists of 12 syntactic segmentation rules and some rules concerning a list of stop phrases, discourse cue phrases, and part-of-speech tags. They also use a list of phrasal discourse cues to insert boundaries not derivable from the parser's output.

Soricut and Marcu [123] introduce a statistical discourse segmenter, which is trained on RST-DT to label words with *boundary* or *no-boundary* labels. They use lexical and syntactic features to determine the probabilities of discourse boundaries $P(b_i|w_i, t)$, where w_i is the i^{th} word of the input sentence s, t is the syntactic parse tree of s, and $b_i \in \{boundary, no\text{-}boundary\}$. Given a syntactic parse tree t, their algorithm inserts a discourse boundary after each word w for which $P(boundary|w, t) > 0.5$.

Another statistical discourse segmenter using artificial neural networks is presented in Subba and Di Eugenio [125]. Like Soricut and Marcu [123], they formulate the discourse segmentation task as a binary classification problem of deciding whether a word is the *boundary* or *no-boundary* of EDUs. Their segmenter exploits a multilayer perceptron model with back-propagation algorithm and is also trained on RST-DT.

Hernault et al. [53] propose a sequential model for the discourse segmentation task, which considers the segmentation task as a sequence labeling problem rather than a classification problem. They exploit Conditional Random Fields (CRFs) [75] as the learning method and get state-of-the-art results on RST-DT.

In our work, like Hernault et al. [53], we also consider the discourse segmentation task as a sequence labeling problem. The final segmentation result is selected among the N-best outputs of a CRF-based model by using a reranking method with subtree features.

3.2.2 Related Work on Discourse Parsing

Several methods have been proposed to deal with the discourse parsing task. In this section, we present the most related studies, which describe a discourse parsing system in the RST framework.

Soricut and Marcu [123] present a sentence level discourse parser. Two probabilistic models are built to segment and to parse texts. Both models exploit syntactic and lexical information. For discourse segmentation, authors report an F-score of 84.7%. For building sentence level discourse trees, they achieve 70.5% in the unlabeled score, and 49.0% and 45.6% in the labeled score when using 18 labels and 110 labels respectively. However, this discourse parser only processes individual sentences.

Sagae [116] proposes a shift-reduced discourse parser. The parser includes a discourse segmenter based on a binary classifier trained on lexico-syntactic features and a parsing model which employs transition algorithms for dependency and constituent trees. Compared to Soricut and Marcu [123], the proposed parser is able to create text level discourse trees. The author reports an F-score of 86.7% for discourse segmentation and 44.5% in the labeled score for building text level discourse trees when using 18 labels.

Hernault et al. [54] describe HILDA, a discourse parser using Support Vector Machine (SVM) classification. The parser exploits following kinds of features: textual organization, lexical features, 'dominance sets' [123], and structural features. They use 18 relations like relations in the work of Sagae [116]. HILDA is considered as the first fully implemented text level discourse parser with state-of-the-art performance. It achieves 72.3% in the unlabeled score and 47.3% in the labeled score on RST-DT.

3.3 A Reranking Model for Discourse Segmentation using Subtree Features

3.3.1 Why Reranking?

Discriminative reranking with linear models has been used successfully in some NLP tasks such as POS tagging, chunking, and statistical parsing [29]. The advantage of the reranking method is that it can exploit the output of a base model to learn. Based on the output of a base model, we can extract long-distance non-local features, which are impossible in sequence learning markov models such as the Hidden markov model, the Maximum entropy markov model, and CRFs. In the discourse segmentation task, we use the reranking method to utilize subtree features extracted from the output of a base model which is learned using CRFs.

3.3.2 Discriminative Reranking

In the discriminative reranking method [29], first, a set of candidates is generated using a base model (GEN). GEN can be any model for the task. For example, in the POS tagging problem, GEN may be a model that generates all possible POS tags for a word based on a dictionary. Then, candidates are reranked using a linear score function:

$$score(y) = \Phi(y) \cdot W$$

where y is a candidate, $\Phi(y)$ is the feature vector of candidate y, and W is a parameter vector. The final output is the candidate with the highest score:

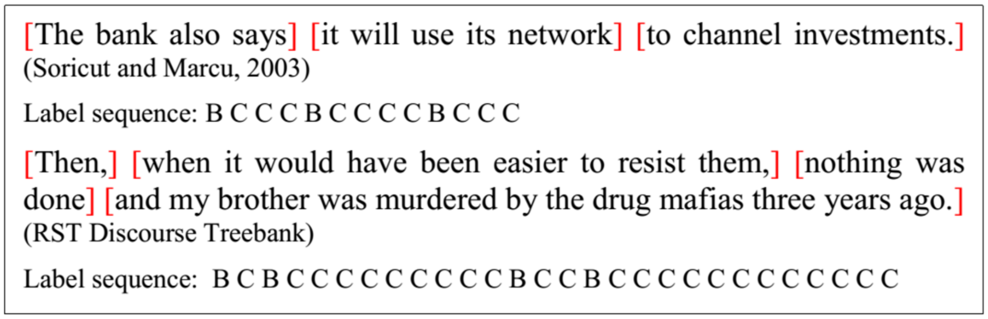
$$\begin{aligned} F(x) &= argmax_{y \in GEN(x)} score(y) \\ &= argmax_{y \in GEN(x)} \Phi(y) \cdot W. \end{aligned}$$

To learn the parameter W we use the average perceptron algorithm, which is presented as Algorithm 1.

In the next sections we will describe our base model and features that we use to rerank candidates.

Algorithm 1 Average perceptron algorithm for reranking [29]

```
1:  Inputs: Training set $\{(x^i, y^i) | x^i \in R^n, y^i \in C, \forall i = 1, 2, \ldots, m\}$
2:  Initialize: $W \leftarrow 0, W_{avg} \leftarrow 0$
3:  Define: $F(x) = argmax_{y \in GEN(x)} \Phi(y) \cdot W$
4:  for $t = 1, 2, \ldots, T$ do
5:    for $i = 1, 2, \ldots, m$ do
6:      $z^i \leftarrow F(x^i)$
7:      if $z^i \neq y^i$ then
8:        $W \leftarrow W + \Phi(y^i) - \Phi(z^i)$
9:      end if
10:     $W_{avg} \leftarrow W_{avg} + W$
11:   end for
12: end for
13: $W_{avg} \leftarrow W_{avg}/(mT)$
14: Output: Parameter vector $W_{avg}$.
```



[The bank also says] [it will use its network] [to channel investments.]
(Soricut and Marcu, 2003)

Label sequence: B C C C B C C C C B C C C

[Then,] [when it would have been easier to resist them,] [nothing was done] [and my brother was murdered by the drug mafias three years ago.]
(RST Discourse Treebank)

Label sequence: B C B C C C C C C C C C B C C B C C C C C C C C C C C C

Figure 3.2: Examples of segmenting sentences into EDUs.

3.3.3 Base Model

Similar to the work of Hernault et al. [53], our base model uses Conditional Random Fields[1] to learn a sequence labeling model. Each label is either *beginning* of EDU (B) or *continuation* of EDU (C). Soricut and Marcu [123] and Subba and Di Eugenio [125] use *boundary* labels, which are assigned to words at the end of EDUs. Like Hernault et al. [53], we use *beginning* labels, which are assigned to words at the beginning of EDUs. However, we can convert an output with *boundary*, *no-boundary* labels to an output with *beginning*, *continuation* labels and vice versa. Figure 3.2 shows two examples of segmenting a sentence into EDUs and their correct label sequences.

We use the following lexical and syntactic information as features:

- Words,
- POS tags,
- Nodes in parse trees, and

[1]We use the implementation of Kudo [70].

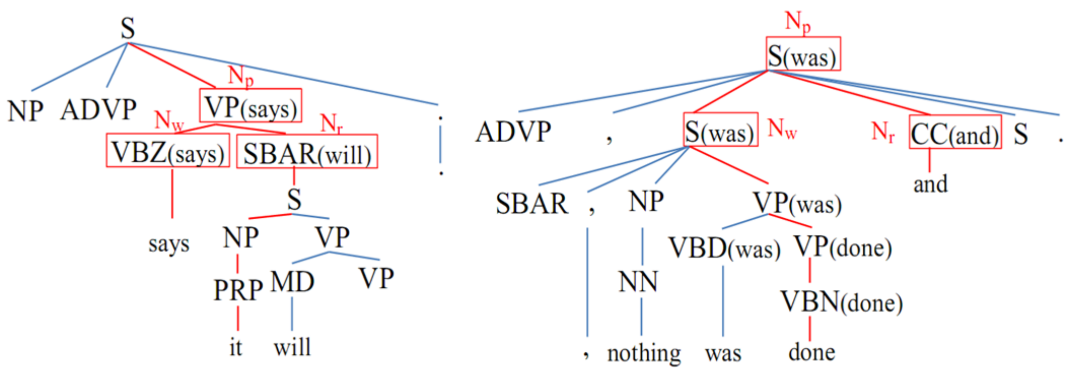


Figure 3.3: Partial lexicalized syntactic parse trees.

- Lexical heads and POS heads of the nodes in parse trees[2].

When extracting features for word w, let r be the word on the right-hand side of w and N_p be the deepest node that belongs to both paths from the root to w and r. N_w and N_r are child nodes of N_p that belong to two paths, respectively. Figure 3.3 shows two partial lexicalized syntactic parse trees. In the first tree, if $w = says$ then $r = it$, $N_p = VP(says)$, $N_w = VBZ(says)$, and $N_r = SBAR(will)$. We also consider the parent and the right-sibling of N_p if any. The final feature set for w consists of not only features extracted from w but also features extracted from two words on the left-hand side and two words on the right-hand side of w.

Our feature extraction method is different from the method in previous works [53, 123]. They define N_w as the highest ancestor of w that has lexical head w and has a right-sibling. Then N_p and N_r are defined as the parent and right-sibling of N_w. In the first example, our method gives the same results as the previous one. In the second example, however, there is no node with lexical head "*done*" and having a right-sibling. The previous method cannot extract N_w, N_p, and N_r in such cases. We also use some new features such as the head node and the right-sibling node of N_p.

3.3.4 Subtree Features for Reranking

We need to decide which kinds of subtrees are useful to represent a candidate, a way to segment the input sentence into EDUs. In our work, we consider two kinds of subtrees: *bound trees* and *splitting trees*.

The *bound tree* of an EDU, which spans from word u to word w, is a subtree which satisfies two conditions:

1. its root is the deepest node in the parse tree which belongs to both paths from the root of the parse tree to u and w, and

2. it only contains nodes in two those paths.

The *splitting tree* between two consecutive EDUs, from word u to word w and from word r to word v, is a subtree which is similar to a bound tree, but contains two paths

[2]Lexical heads are extracted using Collins' rules [28].

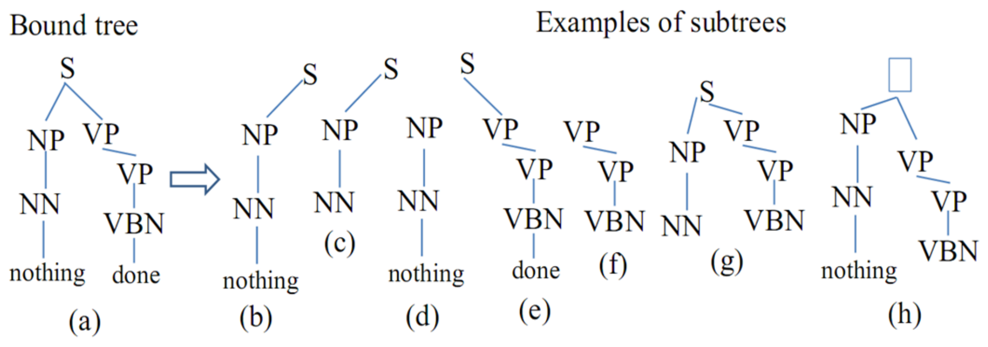


Figure 3.4: Subtree features.

from the root of the parse tree to w and r. Hence, a *splitting tree* between two consecutive EDUs is a *bound tree* that only covers two words: the last word of the first EDU and the first word of the second EDU. Bound trees will cover the whole EDUs, while splitting trees will concentrate on the boundaries of EDUs.

From a bound tree (similar to a splitting tree), we extract three kinds of subtrees: subtrees on the left path (*left tree*), subtrees on the right path (*right tree*), and subtrees consisting of a subtree on the left path and a subtree on the right path (*full tree*). In the third case, if both subtrees on the left and right paths do not contain the root node, we add a pseudo root node. Figure 3.4 shows the bound tree of EDU "*nothing was done*" of the second example in Figure 3.3, and some examples of extracted subtrees.

Each subtree feature is then represented by a string as follows:

- A left tree (or a right tree) is represented by concatenating its nodes with hyphens between nodes. For example, subtrees (b) and (e) in Figure 3.4 can be represented as follows:

  S-NP-NN-nothing, and

  S-VP-VP-VBN-done.

- A full tree is represented by concatenating its left tree and right tree with string ### in the middle. For example, subtrees (g) and (h) in Figure 3.4 can be represented as follows:

  S-NP-NN###S-VP-VP-VBN, and

  NP-NN-nothing###VP-VP-VBN.

The feature set of a candidate is the set of all subtrees extracted from bound trees of all EDUs and splitting trees between two consecutive EDUs.

Among two kinds of subtrees, splitting trees can be computed between any two adjacent words and therefore can be incorporated into the base model. However, if we do so, the feature space will be very large and contains a lot of noisy features. Because many words are not a boundary of any EDU, many subtrees extracted by this method will never become a *real* splitting tree (tree that splits two EDUs). Splitting trees extracted in the reranking model will focus on a small but compact and useful set of subtrees.

3.4 An Incremental Algorithm for Building Discourse Trees

There have been two major approaches building a discourse tree given a segmented text. The first approach uses a greedy strategy [54]. The method gradually combines two consecutive spans (EDUs or subtrees of EDUs), which are most probably connected by a rhetorical relation, until all EDUs are merged into a single discourse tree. The tree construction algorithm is presented as Algorithm 2, where l_i denotes the i^{th} element of list L. The algorithm needs a score function (used in lines 4,9, and 10), which evaluates how likely two consecutive spans should be connected. To calculate this score, we first learn a binary classifier *StructClassifier* that takes two consecutive spans as the input, and returns +1 in the case two spans should be connected and −1 otherwise. Then we define the score function as follows:

$$StructScore(l_i, l_{i+1}) = Prob(StructClassifier(l_i, l_{i+1}) = +1).$$

Algorithm 2 A greedy algorithm for building discourse trees [54].

```
1: Input: List of EDUs, $E = (e_1, e_2, \ldots, e_n)$
2: Initialize: $L \leftarrow E$
3: for $(l_i, l_{i+1})$ in $L$ do
4:     $Scores[i] \leftarrow StructScore(l_i, l_{i+1})$ [Calculate score]
5: end for
6: while $|L| > 1$ do
7:     $i \leftarrow argmax(Scores)$
8:     $NewSubTree \leftarrow CreatTree(l_i, l_{i+1})$ [Create a new subtree]
9:     $Scores[i-1] \leftarrow StructScore(l_{i-1}, NewSubTree)$ [Calculate score]
10:    $Scores[i+1] \leftarrow StructScore(NewSubTree, l_{i+2})$ [Calculate score]
11:    $delete(Scores[i])$      [Update $Scores$]
12:    $L \leftarrow [l_1, \ldots, l_{i-1}, NewSubTree, l_{i+2}, \ldots]$ [Update $L$]
13: end while
14: $FinalTree \leftarrow l_1$
15: Output: $FinalTree$.
```

Note that if we want to build a labeled tree, in addition to *StructClassifier*, we need a multi-class classifier *LabelClassifier* that also takes two consecutive spans as the input, and returns the most probable relation label holding between the two spans. In Algorithm 2, we use this classifier to find the relation label before creating a new subtree (line 8 in the algorithm).

The second approach for building discourse tress is based on a dynamic programming technique (CYK parsing) [123]. We maintain a two-dimension array $t[i][j]$ storing the most probable structure tree covering from the i^{th} EDU to the j^{th} EDU, and an array $Score[i][j]$ storing the score of $t[i][j]$. The final structure tree is $t[1][n]$, where n is the number of EDUs. $Score[i][j]$ and structure tree $t[i][j]$ can be calculated as follows:

$$Score[i][j] = max_{i \le k < j}(Score[i][k] + Score[k+1][j] + StructScore(t[i][k], t[k+1][j])) \quad (3.1)$$

and

$$t[i][j] \leftarrow CreateTree(t[i][k^*], t[k^* + 1][j]),$$

where k^* is the index that maximizes the score function in Equation (3.1).

Although the first method gives an approximate solution, it is more suitable in practice because it runs much faster than the second method ($O(n)$ compared to $O(n^3)$, where n is the number of EDUs). In fact, the second method is only used in sentence level discourse parsing [123], where the number of EDUs is small.

In the tree building step, the goal is to build a discourse tree given a text which has been segmented into EDUs. Usually, the text consists of several paragraphs, and each paragraph consists of some sentences. We note that EDUs within one sentence tend to be connected to make a subtree before connecting to EDUs in other sentences. The same thing also takes place at the paragraph level. EDUs within one paragraph tend to be connected to make a subtree before connecting to EDUs in other paragraphs. It is because of the coherence of a well-written text. A sentence expresses a statement, question, exclamation, request, command, or suggestion, and sentences in a paragraph should focus on a topic. Paragraphs help separate ideas and indicate the change of topics.

Motivated by this property, we propose an incremental algorithm (our first algorithm) for building a discourse tree. The algorithm is presented as Algorithm 3. When we build a subtree for a sentence, the input consists of EDUs in that sentence. When we build a subtree for a paragraph, the input consists of several subtrees, and each subtree corresponds to a sentence. When we build the final discourse tree for whole text, the input consists of several subtrees, and each subtree corresponds to a paragraph. In all three cases, the number of spans (EDUs or subtrees) in the input is very small in comparison with the total EDUs of whole text. So we can use a dynamic programing technique like the method presented in Soricut and Marcu [123] to solve it.

Algorithm 3 An incremental algorithm for building discourse trees.

```
1: Input: a text $T$
2: for each sentence $s$ in $T$ do
3:     Create a subtree for $s$
4: end for
5: for each paragraph $p$ in $T$ do
6:     Create a subtree for $p$ based on subtrees of sentences
7: end for
8: Create a discourse tree for $T$ based on subtrees of paragraphs
9: Output: Discourse tree
```

Compared to the greedy algorithm (Algorithm 2) presented in Hernault et al. [54], this algorithm has two advantages. It supports the coherence property we described before. The algorithm employs a dynamic programming technique, so it can find an extract solution in each step. However, the algorithm gives a hard constraint on the order in which EDUs are connected. So it makes the search process biased. Our solution is creating a parsing model by integrating two above algorithms. To achieve this, in the next section, we will present a dual decomposition algorithm for building discourse trees.

3.5 Dual Decomposition for Building Discourse Trees

3.5.1 Dual Decomposition

Dual decomposition is a method to solve complex optimization problems that can be decomposed into two or more subproblems, together with linear constraints that enforce the agreement on solutions of the subproblems [112]. The subproblems are chosen such that they can be solved efficiently. The constraints are incorporated using Lagrange multipliers, and an iterative algorithm is used to minimize the optimization problem.

We consider the following optimization problem:

$$argmax_{y \in Y, z \in Z}(f(y) + g(z))$$

subject to:

$$y(i) = z(i), \text{ for all } i \in \{1 \dots n\}.$$

Each constraint $y(i) = z(i)$ describes an agreement on the solutions of two subproblems. We introduce Lagrange multipliers $u(i)$, $i \in \{1 \dots n\}$, and assume that for any value $u(i) \in R$, we can efficiently solve:

$$argmax_{y \in Y}(f(y) + \sum_{i=1}^{n} u(i)y(i)), and$$

$$argmax_{z \in Z}(g(z) - \sum_{i=1}^{n} u(i)z(i)).$$

The dual decomposition algorithm can be expressed as Algorithm 4, where δ_k is the step size at the k^{th} iteration.

Algorithm 4 The dual decomposition algorithm [112].

```
 1: Initialize: $u^{(0)}(i) = 0$, for all $i \in \{1 \dots n\}$
 2: for $k = 1$ to $K$ do
 3:     $y^{(k)} \leftarrow argmax_{y \in Y}(f(y) + \sum_{i=1}^{n} u^{(k-1)}(i)y(i))$ [Subproblem 1]
 4:     $z^{(k)} \leftarrow argmax_{z \in Z}(g(z) - \sum_{i=1}^{n} u^{(k-1)}(i)z(i))$ [Subproblem 2]
 5:     if $y^{(k)}(i) = z^{(k)}(i)$ for all $i \in \{1 \dots n\}$ then
 6:         return $(y^{(k)}, z^{(k)})$
 7:     else
 8:         $u^{(k)}(i) \leftarrow u^{(k-1)}(i) - \delta_k(y^{(k)}(i) - z^{(k)}(i))$
 9:     end if
10: end for
11: return $(y^{(K)}, z^{(K)})$
```

As shown by Rush and Collins [112], in the cases that the solutions of two subproblems are agreement, the returned solution will be an optimal solution of the original problem.

Dual decomposition has been applied successfully to several NLP tasks such as parsing [113], dependency parsing [68], and coordination disambiguation [51].

3.5.2 A Parsing Algorithm using Dual Decomposition

Recall that the parsing problem is to find:

$$argmax_{y \in Y} h(y)$$

where Y is the space of all possible discourse trees, and $h(y)$ is a score function defined on Y. In our method, the score function consists of two factors $h(y) = f(y) + g(y)$, where $f(y)$ is the score returned by our base model1 using the incremental algorithm (Algorithm 3), and $g(y)$ is the score returned by our base model2 using the greedy algorithm (Algorithm 2).

For each discourse tree y, we define variables $y(i,j)$ as follows:

$$y(i,j) = \begin{cases} 1 & \text{if exists a subtree that covers from the } i^{th} \text{ EDU to the } j^{th} \text{ EDU} \\ 0 & \text{otherwise.} \end{cases}$$

The problem now becomes:

$$argmax_{y \in Y, z \in Z}(f(y) + g(z))$$

subjects to:

$$y(i,j) = z(i,j) \text{ for all } 1 \le i < j \le n,$$

where n is the number of EDUs and $Z = Y$.

We solve this problem by using dual decomposition. The proposed algorithm (our second algorithm) is presented as Algorithm 5, where $u(i,j)$ are Lagrange multipliers.

Algorithm 5 A dual decomposition algorithm for building discourse trees.

```
1: Initialize: $u^{(0)}(i,j) = 0$, for all $1 \le i < j \le n$.
2: for $k = 1$ to $K$ do
3:     $y^{(k)} \leftarrow argmax_{y \in Y}(f(y) + \sum_{1 \le i < j \le n} u^{(k-1)}(i,j)y(i,j))$ [Base Model1]
4:     $z^{(k)} \leftarrow argmax_{z \in Z}(g(z) - \sum_{1 \le i < j \le n} u^{(k-1)}(i,j)z(i,j))$ [Base Model2]
5:     if $y^{(k)}(i,j) = z^{(k)}(i,j)$ for all $1 \le i < j \le n$ then
6:         return $y^{(k)}$
7:     else
8:         $u^{(k)}(i,j) \leftarrow u^{(k-1)}(i,j) - \delta_k(y^{(k)}(i,j) - z^{(k)}(i,j))$
9:     end if
10: end for
11: return $y^{(K)}$
```

Note that when solving two subproblems (lines 3 and 4 in Algorithm 5) using two base algorithms (Algorithms 2 and 3), we need to modify score functions as follows:

$$Score[i][j] = max_{i \le k < j}(Score[i][k] + Score[k+1][j] + StructScore(t[i][k], t[k+1][j])) + \mathbf{u(i,j)}),$$

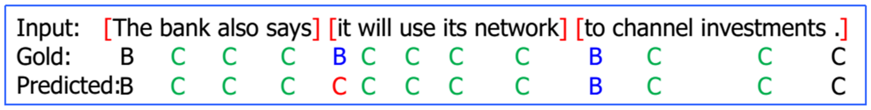
in Base Model1, and

$$Score[i][j] = max_{i \le k < j}(Score[i][k] + Score[k+1][j] + StructScore(t[i][k], t[k+1][j])) - \mathbf{u(i,j)}),$$

in Base Model2.

Table 3.1: Statistical information of the RST Discourse Treebank corpus

Dataset	Number of articles	Number of sentences
Training	347	6132
Test	38	991



Input:	[The	bank	also	says]	[it	will	use	its	network]	[to	channel	investments	.]
Gold:	B	C	C	C	B	C	C	C	C	B	C	C	C
Predicted:	B	C	C	C	C	C	C	C	C	B	C	C	C

Figure 3.5: An example sentence for illustrating two evaluation methods.

To learn the binary classifier $StructClassifier$, which is used to compute $StructScore$, like Hernault et al. [54], we use lexical and syntactic features including textual organization features, lexical features, 'dominance sets' [123], and structural features. We also employ Support Vector Machines[3] as the learning method.

3.6 Experiments

3.6.1 Data and Evaluation Methods

We tested our system on the RST Discourse Treebank (RST-DT) corpus [21]. Table 3.1 shows statistical information of RST-DT. This corpus consists of 385 articles from the Penn Treebank [83], which are divided into a Training set and a Test set. The Training set consists of 347 articles (6132 sentences), and the Test set consists of 38 articles (991 sentences).

For the discourse segmentation task, there are two evaluation methods that have been used in previous work. The first method measures only *beginning* labels (B labels) [123, 125]. The second method [53] measures both *beginning* and *continuation* labels (B and C labels)[4]. This method first calculates scores on B labels and scores on C labels, and then produces the average of them.

Figure 3.5 shows an example of a sentence in the sequence model and a predicted output. In this example, the output contains a mistake, which is the C label of the word *it*. Now we will compute precision, recall, and the F_1 score under the two evaluation schemes.

- The first evaluation method uses only B labels.

  $Precision = 1/1 = 100\%$, $Recall = 1/2 = 50.0\%$, $F_1 = 66.7\%$.

  Note that we do not count the first B label, which is a sentence boundary.

- The second method uses both B and C labels.

  For B labels: $Precision = 1/1 = 100\%$, $Recall = 1/2 = 50.0\%$, $F_1 = 66.7\%$.

  For C labels: $Precision = 9/10 = 90.0\%$, $Recall = 9/9 = 100\%$, $F_1 = 94.7\%$.

[3]In our experiments, we used Libsvm: http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[4]Neither evaluation method counts sentence boundaries.

Average: *Precision* = 95.0%, *Recall* = 75.0%, $F_1 = 80.7\%$.

Note that we do not count the last C label, which is a sentence boundary.

Due to the number of C labels being much higher than the number of B labels, the second evaluation method yields much higher results. In Hernault et al. [53], the authors compare their systems with previous work despite using different evaluation methods. Such comparisons are not valid. In our work, we measure the performance of the proposed model using both methods.

For the discourse parsing task, we measure the performance of the proposed system using the unlabeled score, which is the same as the unlabeled score described in previous work [54, 82, 123].

3.6.2 Experiments on Discourse Segmentation

We learned the base model on the Training set and tested on the Test set to get the N-best outputs to rerank. To learn parameters of the reranking model, we conducted 5-fold cross-validation tests on the Training set. In all experiments, we set N to 20. To choose the number of iterations, we used a development set, which is about 20 percent of the Training set.

Table 3.2 shows experimental results when evaluating only *beginning* (B) labels, in which SPADE is the work of Soricut and Marcu[123], NNDS is a segmenter that uses neural networks [125], and CRFSeg is a CRF-based segmenter [53]. When using gold parse trees, our base model got 92.5% in the F_1 score, which improves 1.3% compared to the state-of-the-art segmenter (CRFSeg). When using Stanford parse trees [66], our base model improved 1.7% compared to CRFSeg. It demonstrates the effectiveness of our feature extraction method in the base model. As expected, our reranking model got better results compared to the base model in both settings. The reranking model got 93.7% and 91.0% in two settings, which improves 2.5% and 2.0% compared to CRFSeg. Also note that, when using Stanford parse trees, our reranking model got competitive results with CRFSeg when using gold parse trees (91.0% compared to 91.2%).

Table 3.3 shows experimental results when evaluating on both *beginning* and *continuation* labels. Our models also outperformed CRFSeg in both settings, using gold parse trees and using Stanford parse trees (96.6% compared to 95.3% in the first setting, and 95.1% compared to 94.1% in the second setting).

Both evaluation methods have a weak point in that they do not measure the ability to find EDUs exactly. We suggest that the discourse segmentation task should be measured on EDUs rather than boundaries of EDUs. Under this evaluation scheme, our model achieved 90.0% and 86.2% when using gold parse trees and Stanford parse trees, respectively.

A direct comparison with systems described in [134] and [137] is not possible due to the difference of datasets. Thanh et al. [134] evaluated their system on only 8 texts of RST-DT with gold standard parse trees. They achieved 81.4% and 79.2% in the precision and recall scores, respectively. Tofiloski et al. [137] tested their system on only 3 texts of RST-DT and used different segmentation guidelines. They reported a precision of 82.0% and recall of 86.0% when using Stanford parse trees.

An important question is which subtree features were useful for the reranking model. This question can be answered by looking at the weights of subtree features (the parameter

Table 3.2: Performance when evaluating on B labels

Model	Trees	Precision(%)	Recall(%)	F_1(%)
SPADE	Penn	84.1	85.4	84.7
NNDS	Penn	85.5	86.6	86.0
CRFSeg	Penn	92.7	89.7	91.2
Base	Penn	92.5	92.5	92.5
Reranking	Penn	**93.1**	**94.2**	**93.7**
CRFSeg	Stanford	91.0	87.2	89.0
Base	Stanford	91.4	90.1	90.7
Reranking	Stanford	**91.5**	**90.4**	**91.0**
Human	-	98.5	98.2	98.3

Table 3.3: Performance when evaluating on B and C labels

Model	Trees	Precision(%)	Recall(%)	F_1(%)
CRFSeg	Penn	96.0	94.6	95.3
Base	Penn	96.0	96.0	96.0
Reranking	Penn	**96.3**	**96.9**	**96.6**
CRFSeg	Stanford	95.0	93.2	94.1
Base	Stanford	95.3	94.7	95.0
Reranking	Stanford	**95.4**	**94.9**	**95.1**

vector learned by the average perceptron algorithm). Table 3.4 shows 30 subtree features with the highest weights in absolute value. These features are thus useful for reranking candidates in the reranking model. We can see that most subtree features at the top are *splitting trees*, so *splitting trees* have a more important role than *bound trees* in our model. Among three types of subtrees (*left tree*, *right tree*, and *full tree*), *full tree* is the most important type. It is understandable because subtrees in this type convey much information; and therefore describe *splitting trees* and *bound trees* more precise than subtrees in other types.

Now we discuss the cases in which our model fails to segment discourses. Note that all errors belong to one of two types, *over-segmentation* type (i.e., words that are not EDU boundaries are mistaken for boundaries) and *miss-segmentation* type (i.e., words that are EDU boundaries are mistaken for not boundaries).

Tabel 3.5 shows 15 most frequent words for which our model usually makes a mistake and their percentage among all segmentation errors. Most errors are related to coordinating conjunctions and subordinators (*and*, *that*, *as*, *if*, *when*), personal pronouns (*he*, *it*, *they*), determiners (*the*, *a*), prepositions (*of*, *without*), punctuations (quotes and hyphens), and the word *to*.

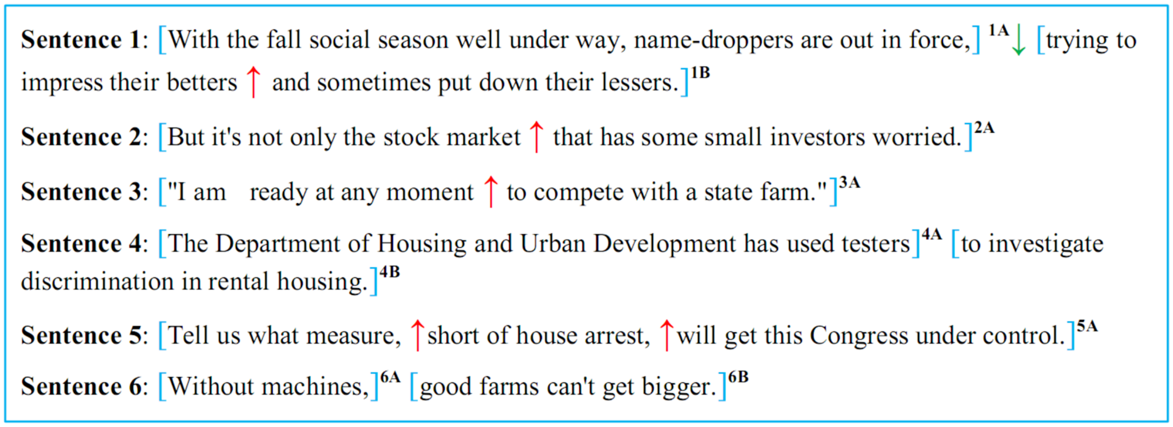
Figure 3.6 shows some errors made by our model. In these examples, gold (correct) EDU boundaries are marked by bracket squares ([]), while predicted boundaries made by our model are indicated by arrows (↓ or ↑). A down arrow (↓) shows a boundary which is predicted correctly, while an up arrow (↑) indicates an *over-segmentation* error. A boundary with no arrow means a *miss-segmentation* error. For example, in Sentence 1, we have a correct boundary and an *over-segmentation* error. Sentences 2 and 3 show two *over-segmentation* errors, and sentences 4 and 6 show two *miss-segmentation* errors.

Table 3.4: Top 30 subtree features with the highest weights

Type of tree	Type of subtree	Subtree feature	Weight
Splitting tree	Full tree	NP###NP-VP	23.0125
Splitting tree	Full tree	VP###S-VP	19.3044
Splitting tree	Full tree	NP###VBN	18.3862
Splitting tree	Right tree	VP	-18.3723
Splitting tree	Full tree	NP###SBAR	17.7119
Splitting tree	Full tree	NP###NP-SBAR	17.0678
Splitting tree	Full tree	NP###,	-16.6763
Splitting tree	Full tree	NP###VP	15.9934
Splitting tree	Left tree	NP-VP	15.2849
Splitting tree	Full tree	NP###NP	15.1657
Splitting tree	Right tree	SBAR	14.6778
Splitting tree	Full tree	NP###S-NP	14.4962
Splitting tree	Full tree	NP###S	13.1656
Bound tree	Full tree	S-PP###,	12.7428
Splitting tree	Full tree	NP###NP-VP-VBN	12.5210
Bound tree	Full tree	NP###NP	-12.4723
Bound tree	Full tree	VP###VP	-12.1918
Splitting tree	Full tree	NP-VP###S	12.1367
Splitting tree	Right tree	NP-VP	12.0929
Splitting tree	Full tree	NP-SBAR###VP	12.0858
Splitting tree	Full tree	NP-SBAR-S###VP	12.0858
Splitting tree	Full tree	VP###VP-VP	-12.0338
Bound tree	Full tree	VBG###.	11.9067
Bound tree	Right tree	:	11.8833
Bound tree	Full tree	VP###S	-11.7624
Bound tree	Full tree	S###VP	-11.7596
Bound tree	Full tree	“###”	11.5524
Bound tree	Full tree	S###,	11.5274
Splitting tree	Full tree	NP###VP-VBN	11.3342
Bound tree	Left tree	0	11.2878

Table 3.5: Top error words

Word	Percentage among all errors (%)
to	14.5
and	5.8
that	4.6
the	4.6
“	3.5
he	2.3
it	2.3
of	2.3
without	2.3
–	1.7
as	1.7
if	1.7
they	1.7
when	1.7
a	1.2



Sentence 1: [With the fall social season well under way, name-droppers are out in force,]^{1A}↓ [trying to impress their betters ↑ and sometimes put down their lessers.]^{1B}

Sentence 2: [But it's not only the stock market ↑ that has some small investors worried.]^{2A}

Sentence 3: ["I am ready at any moment ↑ to compete with a state farm."]^{3A}

Sentence 4: [The Department of Housing and Urban Development has used testers]^{4A} [to investigate discrimination in rental housing.]^{4B}

Sentence 5: [Tell us what measure, ↑short of house arrest, ↑will get this Congress under control.]^{5A}

Sentence 6: [Without machines,]^{6A} [good farms can't get bigger.]^{6B}

Figure 3.6: Some errors made by our model.

We also note that many errors occur right after punctuations (commas, quotes, hyphens, brackets, and so on). We analyzed statistics on words that appear before error words. Table 3.6 shows 10 most frequent words and their percentage among all errors. Overall, more than 35% errors occur right after punctuations.

3.6.3 Experiments on Discourse Parsing

We tested our system on the Test set of RST-DT in two settings. In the first setting, we used gold segmentation and Peen Treebank parse trees. The purpose of this setting is to test the performance of the proposed parsing model. In the second setting, we used segmentation produced by our discourse segmenter and Stanford parse trees. The purpose of this setting is to test the performance of the full system. In all experiments, the step size δ_k was chosen as the guidance in Rush and Collins [112], and the number of iterations

Table 3.6: Most frequent words that appear before error words

Word	Percentage among all errors (%)
,	24.9
“	5.2
–	2.3
time	1.7
)	1.2
assets	1.2
investors	1.2
month	1.2
plan	1.2
was	1.2

Table 3.7: Experimental results of the tree building step (gold segmentation and gold parse trees)

System	Algorithm	Precision(%)	Recall(%)	F_1(%)	Improvement(%)
HILDA	Greedy	83.0	83.0	83.0	-
UDRST	Incremental	84.3	84.3	84.3	1.3
	Dual	**84.6**	**84.6**	**84.6**	**1.6**

K was set to 10.

Table 3.7 shows experimental results of the tree building step in the unlabeled score. When using the incremental algorithm, UDRST achieved 84.3% in the F_1 score, which improves 1.3% compared to HILDA. As expected, UDRST with the dual decomposition algorithm got the better result than UDRST with the incremental algorithm (84.6% compared to 84.3%).

Table 3.8 shows the performance of the full system in the unlabeled score. UDRST outperformed HILDA in both algorithms, the incremental algorithm and the dual decomposition algorithm. It achieved 77.0% and 77.3% in two algorithms, which improve 4.7% and 5.0% compared to HILDA.

We do not compare our system to systems described in Sagae [116] and in Soricut and Marcu [123]. Sagae [116] does not report the performance of his system in the unlabeled score. Soricut and Marcu [123] evaluate their system only on sentence level discourse parsing. They achieves 70.5% in the unlabeled score.

3.7 Conclusions

In this chapter, we have presented our study on learning discourse structures in the RST framework. We first introduced a reranking model for the discourse segmentation task. Our model exploits subtree features to rerank the N-best outputs of a base model, which uses CRFs to learn. We then presented a novel model for unlabeled discourse parsing which employs dual decomposition. The basic idea of the parsing model is that EDUs in a sentence (a paragraph) tend to be connected to form a subtree before connecting to EDUs in other sentences (paragraphs). This idea is put into our model by integrating an incremental model and a greedy model using dual decomposition. Experiments on the

Table 3.8: Performance of the full system (our segmentation model and Stanford parse trees)

System	Algorithm	Precision(%)	Recall(%)	F_1(%)	Improvement(%)
HILDA	Greedy	73.0	71.7	72.3	-
UDRST	Incremental	77.2	76.7	77.0	4.7
	Dual	**77.5**	**77.0**	**77.3**	**5.0**

RST Discourse Treebank corpus show that our models achieved the best results on both the discourse segmentation task and the unlabeled discourse parsing task.

Chapter 4

Analyzing Logical Structures of Legal Texts

4.1 Introduction

In recent years, a new research field called Legal Engineering has been proposed in the 21st Century COE Program, Verifiable and Evolvable e-Society[62, 63, 64]. Legal Engineering serves to exam and verify whether a law has been established appropriately according to its purpose, whether a law contains contradictions, whether the law is consistent with related laws, and whether the law has been modified, added, and deleted consistently. There are two important goals of Legal Engineering. The first goal is to help experts make complete and consistent laws, and the other is to design an information system which works based on laws.

Legal Engineering regards that laws are a kind of software for our society. Specifically, laws such as pension law are specifications for information systems such as pension systems. To achieve a trustworthy society, laws need to be verified about their consistency and contradiction.

Legal texts have some specific characteristics that make them different from other daily-use documents. Legal texts are usually long and complicated. They are composed by experts who spent a lot of time to write and check carefully.

One of the most important characteristics is that legal texts usually have some specific structures at both sentence and paragraph levels. At the sentence level, a law sentence can roughly be divided into two logical parts: *requisite part* and *effectuation part* [4, 6, 132]. At the paragraph level, a paragraph usually contains a main sentence[1] and one or more subordinate sentences [130]. In this chapter, we will consider the task of analyzing logical structures of legal texts, in which we first recognize logical parts in legal articles and then group related logical parts into logical structures.

In the remainder of this section, we first explain about logical parts and logical structures[2]. We next describe the motivation of this work, and the overview of this chapter.

[1]Usually, the first sentence is the main sentence.

[2]The formal definitions of them will be presented in Section 4.3

4.1.1 Logical Parts and Logical Structures of Legal Texts

A logical part is a clause or phrase in law sentences that conveys a part of the meaning of legal texts. Each logical part contains a specific kind of information according to its type. Three main types of logical parts are *antecedent part*, *consequent part*, and *topic part*[3]. A logical part in *consequent* type describes a law provision; a logical part in *antecedent* type indicates cases (or the context) the law provision can be applied; and a logical part in *topic* type describes subjects related to the law provision.

A logical structure is a pair of two *high-level* logical parts, a *requisite part* and an *effectuation part* in the form:

$$\textit{requisite part} \Rightarrow \textit{effectuation part}.$$

Note that the implication mark is used to connect the *requisite part* and the *effectuation part*. It is not exactly the same as the logical implication.

Each *requisite part* or *effectuation part* consists of several logical parts. This is the reason why we call them *high-level* logical parts. In a simple case, the *requisite part* only consists of one *antecedent part*; and the *effectuation part* only consists of one *consequent part*. Now, we will consider main cases of logical structures of legal texts.

Single Sentence Case

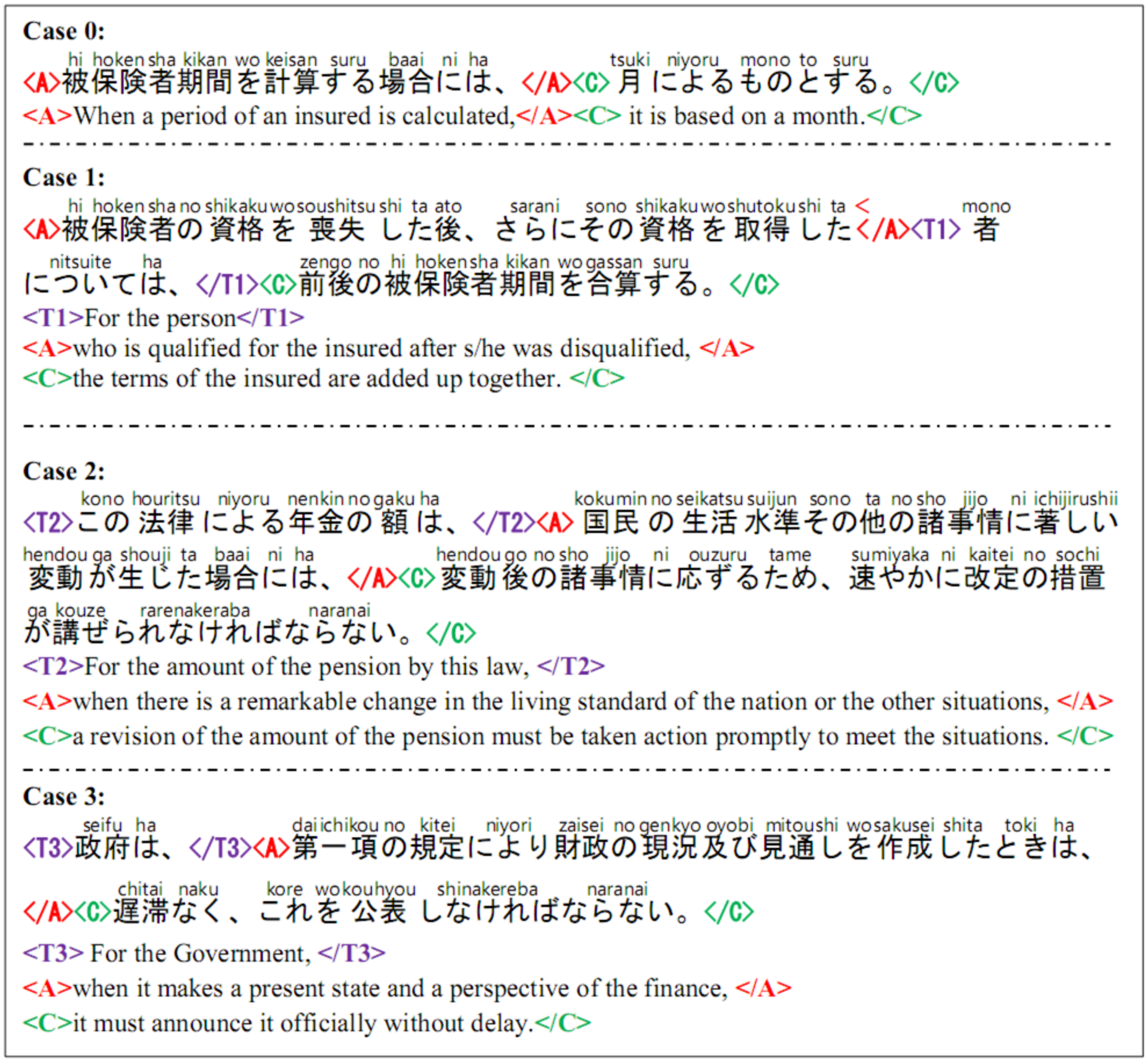
Figure 4.1 shows four popular cases of law sentences and their logical parts. Usually, a law sentence consists of a topic part (T), an antecedent part (A), and a consequent part (C). Four cases are divided depending on the relationships between the topic part and the antecedent part and the consequent part.

- Case 0: There is no topic part. The requisite part only consists of the antecedent part, and the effectuation part only consists of the consequent part.
- Case 1: The topic part depends on the antecedent part. The requisite part is composed from the topic part and the antecedent part, while the effectuation part only consists of the consequent part.
- Case 2: The topic part depends on the consequent part. The requisite part consists of the antecedent part, while the effectuation part is composed from the topic part and the consequent part.
- Case 3: The topic part depends on both the antecedent part and the consequent part. The requisite part is composed from the topic part and the antecedent part, while the effectuation part is composed from the topic part and the consequent part.

Multiple Sentences Case

Figure 4.2 shows four popular cases of legal paragraphs consisting of multiple sentences and their logical structures defined by Takano et al. [130]. Usually, a legal paragraph consists of a main sentence and a subordinate sentence. The main sentence is the first sentence in the paragraph, while the subordinate sentence can be the second sentence or an embedded sentence in parentheses within the main sentence.

[3]In our corpus, three main types of logical parts make up 82% of all types.



Case 0:

hi hoken sha kikan wo keisan suru baai ni ha tsuki niyoru mono to suru

<A>被保険者期間を計算する場合には、</A><C> 月 によるものとする。</C>

<A>When a period of an insured is calculated,</A><C> it is based on a month.</C>

Case 1:

hi hoken sha no shikaku wo soushitsu shi ta ato sarani sono shikaku wo shutoku shi ta < mono

<A>被保険者の 資格 を 喪失 した後、さらにその資格 を 取得 した</A><T1> 者

nitsuite ha zengo no hi hoken sha kikan wo gassan suru

については、</T1><C>前後の被保険者期間を合算する。</C>

<T1>For the person</T1>

<A>who is qualified for the insured after s/he was disqualified, </A>

<C>the terms of the insured are added up together. </C>

Case 2:

kono houritsu niyoru nenkin no gaku ha kokumin no seikatsu suijun sono ta no sho jijo ni ichijirushii

<T2>この 法律 による年金の 額 は、</T2><A> 国民 の 生活 水準その他の諸事情に著しい

hendou ga shouji ta baai ni ha hendou go no sho jijo ni ouzuru tame sumiyaka ni kaitei no sochi

変動 が生じた場合には、</A><C> 変動 後の諸事情に応ずるため、速やかに改定の措置

ga kouze rarenakeraba naranai

が講ぜられなければならない。</C>

<T2>For the amount of the pension by this law, </T2>

<A>when there is a remarkable change in the living standard of the nation or the other situations, </A>

<C>a revision of the amount of the pension must be taken action promptly to meet the situations. </C>

Case 3:

seifu ha dai ichikou no kitei niyori zaisei no genkyo oyobi mitoushi wo sakusei shita toki ha

<T3>政府は、</T3><A>第一項の規定により財政の現況及び見通しを作成したときは、

chitai naku kore wo kouhyou shinakereba naranai

</A><C>遅滞なく、これを 公表 しなければならない。</C>

<T3> For the Government, </T3>

<A>when it makes a present state and a perspective of the finance, </A>

<C>it must announce it officially without delay.</C>

Figure 4.1: Examples of law sentences and logical parts (A: antecedent part; C: consequent part; T: topic part).

- Case 1 (Individual type): The main and subordinate sentences are represented in two logical structures. The first logical structure corresponds to the main sentence. The second logical structure corresponds to the subordinate sentence and some part of the main sentence.

- Case 2 (Embedded type): The main and subordinate sentences are represented in one logical structure.

- Case 3 (Mixed type): The main sentence and some part of the subordinate sentence are represented in the first logical structure, and the subordinate and some part of the main sentence are represented in the second logical structure.

- Case 4 (Independent type): The main sentence is represented in the first logical structure and the subordinate sentence is represented in the second logical structure.

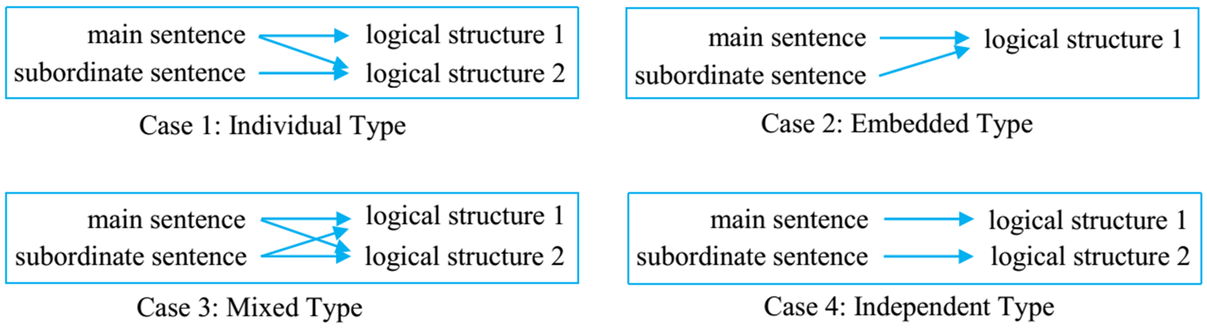
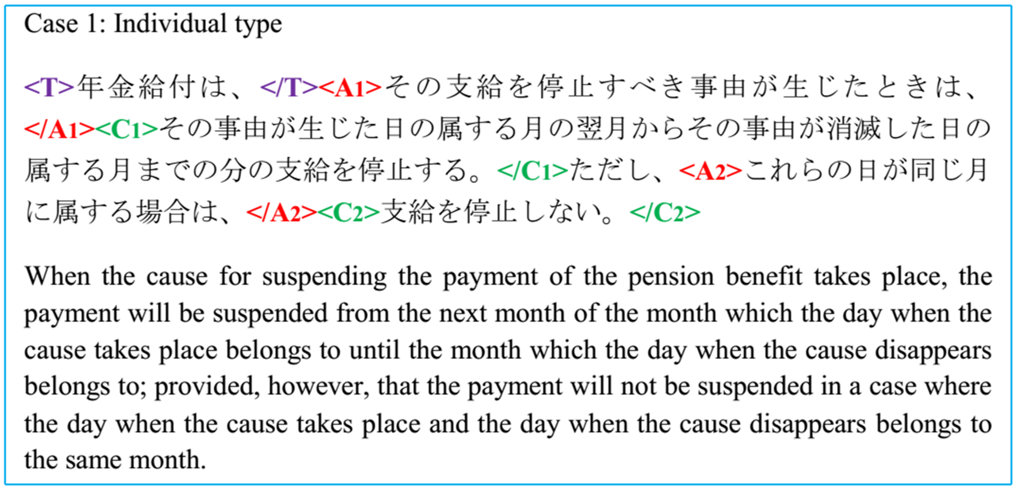


Figure 4.2: Four popular cases of legal paragraphs [130].


Case 1: Individual type

<T>年金給付は、</T><A1>その支給を停止すべき事由が生じたときは、</A1><C1>その事由が生じた日の属する月の翌月からその事由が消滅した日の属する月までの分の支給を停止する。</C1>ただし、<A2>これらの日が同じ月に属する場合は、</A2><C2>支給を停止しない。</C2>

When the cause for suspending the payment of the pension benefit takes place, the payment will be suspended from the next month of the month which the day when the cause takes place belongs to until the month which the day when the cause disappears belongs to; provided, however, that the payment will not be suspended in a case where the day when the cause takes place and the day when the cause disappears belongs to the same month.

Figure 4.3: An example of Individual type in the JNPL corpus (A means *Antecedent part*; C means *Consequent part*; T means *Topic part*.

Figure 4.3 shows an example of Individual type (Case 1) in the JNPL corpus. In this example, the main sentence consists of a topic part (T), an antecedent part (A1), and a consequent part (C1). The subordinate sentence consists of an antecedent part (A2) and a consequent part (C2). The first logical structure corresponds to the main sentence (means T, A1, and C1). The second logical structure corresponds to the subordinate sentence (means A2 and C2) and some part of the main sentence (means T).

In our work, we first recognize logical parts, then group logical parts into logical structures. We do not recognize the relations between logical parts. In this sense, a logical structure can be seen as a group of some logical parts.

4.1.2 Motivation of This Work

Analyzing logical structures of legal texts is an important task in Legal Engineering. The outputs of this task will be beneficial to people in understanding legal texts. They can help understanding:

1. What does a law sentence say? (The *consequent part* of a logical structure is the answer for this question.)

2. What cases the law sentence can be applied? (The *antecedent part* is the answer for this question.)

3. What subjects are related to the provision described in the law sentence? (The *topic part* is the answer for this question.)

This task is the preliminary step, which supports other tasks in legal text processing (translating legal articles into logical and formal representations, legal text summarization, legal text translation, question answering in legal domains, etc) and serves legal text verification, an important goal of Legal Engineering. For example, in the task of translating legal articles into logical and formal representations, we can do as follows:

1. Recognizing logical parts and logical structures in the legal articles

2. Translating each logical part into a formal representation

3. Combining formal representations of logical parts in the same logical structures into a single one.

Until now, most researches have focused on analyzing logical structures of legal texts at the sentence level. Analyzing law sentences individually is not enough to understand a legal document. In most cases, to understand a law sentence, we need to understand related sentences or its context. To the best of our knowledge, however, no existing research addresses the task at the paragraph level. Analyzing logical structures of legal paragraphs is therefore a next step in the research of Legal Engineering to achieve its goals: *helping people in understanding legal texts* and *making computers able to process legal texts automatically.*

4.1.3 Overview of This Chapter

In this chapter, we address the task of analyzing logical structures of legal articles at the paragraph level. We propose a two-phase framework to complete the task, in which we recognize logical parts in the first phase and logical structures in the second phase. We also describe experimental results on legal data. Our main contributions can be summarized in the following points.

1. Introducing a new task to legal text processing, analyzing logical structures of paragraphs in legal articles.

2. Presenting an annotated corpus for the task, the Japanese National Pension Law corpus.

3. Proposing a two-phase framework and providing solutions to solve the task.

4. Evaluating our framework on a real annotated corpus.

The rest of this chapter is organized as follows. We first describe related work in Section 4.2. We next present our task and its two subtasks: *recognition of logical parts* and *recognition of logical structures* in Section 4.3. Section 4.4 presents the overall architecture of our framework. In Section 4.5, we present our solution for the first subtask: multi-layer sequence learning model for recognizing logical parts. Section 4.6 describes our solution for the second subtask: integer linear programming for recognizing logical structures. Experimental results on legal articles are described in Section 4.7. This section also presents limitation and methods for improving the performance of our system. Finally, conclusions and directions for further research are presented in Section 4.8.

4.2 Related Work

This section presents related work and places our work in the scope of the legal text processing field. First, we present studies on legal text processing in general. Then, we focus on studies on analyzing logical structures of Japanese legal texts, which are closest to our work.

4.2.1 Studies on Legal Text Processing

Studies on legal text processing can be categorized into several topics including Legal Ontology Learning, Legal Information Extraction, Legal Semantic Annotation, Automatic Identification of Legal Terms, Legal Knowledge Modeling, Legal Argumentation, Legal Automatic Summarization, and Fundamental NLP Tasks for Legal Texts[4]. In the following, we describe a short summary of previous studies according to each topic[5].

- *Legal Ontology Learning*: Lame [76] presents a general method using NLP techniques for recognizing legal concepts and semantic relations among them, the main components of an ontology. Using this method, the author built an ontology of French laws, which is dedicated to information retrieval. Saias and Quaresma [117] present a methodology to automatically create an OWL (Ontology Web Language) ontology from a set of legal documents in five steps: 1) Definition of an initial top-level ontology; 2) Identification of concepts referred in the legal documents and extraction of its properties; 3) Identification of relations between the identified concepts; 4) Creation of an ontology using the identified concepts and relations; 5) Mergence of the created ontology with the initial ontology. Both Lame [76] and Saias and Quaresma [117] identify components of legal ontologies from the analysis of legal texts. The method of Saias and Quaresma [117], however, can exploit initial documents which are enriched with instances to build the final ontology. The authors claim that this process allows the definition of semantic web agents able to query the semantic content of these documents.

  A rule-based method for extracting and analyzing definitions from parsed texts is presented in Walter and Pinkal [145]. They evaluated this method on a corpus of about 6000 German court decisions and reported an experiment exploring the use of extraction results to improve the quality of text-based ontology learning. Völker et al. [143] describe the open-source software Text2Onto2, a framework for ontology learning from open-domain unstructured text.

[4]First seven topics are chosen from [141]. The last topic is added by ourselves.

[5]We do not have ambitions to present all the previous work. We only try to provide a full picture about research into legal text processing field.

- *Legal Information Extraction*: Walter [144] presents a rule-based method, which uses dependency parse trees, to extract definitions from German court decisions. McCarty [86] presents a method to compute semantic interpretations of legal texts from the output of a syntactic parser. The author introduced an initial legal corpus consisting of federal civil cases in the appellate courts in the United States. However neither experiment nor evaluation was described.

- *Legal Semantic Annotation*: Brighi et al. [16] present an approach for the automatic annotation of modificatory provisions of Italian laws. They adopted a rule-based algorithm to fill the semantic roles of the semantic frame associated with the modificatory provision. However this work is still in a prototypal stage.

  Spinosa et al. [124] present a system for the automatic consolidation of Italian legislative texts. The goal of the system is to be used as a support of an editorial consolidating activity. The proposed approach to consolidation is metadata-oriented (XML-based) and based on NLP techniques. The system was implemented and evaluated on Italian textual amendments.

- *Automatic Identification of Legal Terms*: Pala et al. [103] describe a project on identification of legal terms. The goal of this project is to build an electronic dictionary of Czech law terms. They also presented a legal database including approximate 50,000 Czech law documents.

- *Legal Knowledge Modeling*: Nakamura et al. [95] describe a rule-based system which translates legal texts into logical forms. Their logical formalization conforms to Davidsonian Style, which is suitable for languages allowing expressions with zero-pronouns such as Japanese. The system achieved 78% accuracy in terms of deriving predicates with bound variables.

- *Legal Argumentation*: Moens et al. [90] describe an investigation on the detection of arguments in legal texts. They considered the detection task as a classification problem and built a classifier using a set of annotated arguments. Various kinds of features were evaluated including lexical, syntactic, semantic, and discourse properties of texts.

  Wyner et al. [149] present recent approaches to automatic identification of legal arguments, which use Context Free Grammar, ontologies, and NLP techniques

- *Legal Automatic Summarization*: Grover et al. [49] present a method for automatic summarization of legal documents in two steps: 1) sentences in the legal documents are classified according to their rhetorical role; 2) sentences are selected to form a summary based on their rhetorical role.

- *Fundamental NLP Tasks for Legal Texts*: These studies investigate fundamental NLP tasks (such as morphology, syntactic parsing, chunking and so on) on legal domains. Pala et al. [102] explore the morphology of the Czech law texts on a corpus of approximate 50,000 Czech law documents including Constitution, acts, public notices, and court judgements. Venturi [141] describes an investigation on syntactic and lexical characteristics of legal language (Italian and English laws) with respect to ordinary language. According to the author, understanding these characteristics of specialized languages has practical importance in the development of domain-specific applications.

4.2.2 Studies on Analyzing Logical Structures of Japanese Legal Texts

Analyzing logical structures of legal texts can be considered as a subtopic of legal knowledge modeling, in which we try to model knowledge conveyed in legal documents.

There have been some studies analyzing logical structures of Japanese legal texts. Tanaka et al [132] describe the standard structure of legal provisions based on the principle of legal *condition-effect*. Tanaka [131] analyzes semantic functions of the legal-effect's restrictive part and its semantic restriction to the provision.

Muramatsu et al. [92] describe a tool that displays logical structure of legal sentence from tagged legal sentences. This tool consists of two functions: 1) tagging support function, which labels automatically for gross structure, and displays tag candidates for logical structure; 2) logical structure display function, which shows a logical structure of a legal sentence based on the tag information.

Recently, a new research field called Legal Engineering has been proposed in the 21st Century COE Program, Verifiable and Evolvable e-Society [62, 63, 64]. Several works have been conducted in this program. Bach et al. [6] present the RRE task[6], which analyzes logical structures of legal texts at the sentence level. In the RRE task, the goal is to recognize logical parts given an input law sentence. This task considers two types of sentences (*implication type*[7] and *equivalence type*) and seven kinds of logical parts (three kinds of *topic parts*, *antecedent part*, *consequent part*, *left equivalent part*, and *right equivalent part*).

Compared to the RRE task, which analyzes logical structures of legal texts at the sentence level, our task is more difficult in some points:

- In the RRE task, we only consider a single sentence. We assume that all logical parts in a sentence belong to the same logical structure. In this task, we consider multiple sentences. A logical structure consists of several logical parts in different sentences. A logical part also can belong to multiple logical structures.

- In this task, we also consider cases that a logical part contains other logical parts (embedded relationship is possible).

Several machine learning models have been proposed to deal with the RRE task [6], in which the task is modeled as a sequence learning problem. Experimental results showed that Conditional random files (CRFs) [75] can solve the task relatively well. They achieved nearly 90% in F_1 score on the Japanese National Pension Law corpus.

Bach et al. [5] describe an investigation on contributions of words to the RRE task. Authors presented a method to evaluate the importance of words in the task and found that words that have strong relations to the logical structure of law sentences are very important for machine learning models. Kimura et al. [65] focus on dealing with legal sentences including itemized and referential expressions. They presented a rule-based

[6]The task of Recognition of Requisite part and Effectuation part in law sentences.

[7]Most sentences (98.6%) belong to implication type.

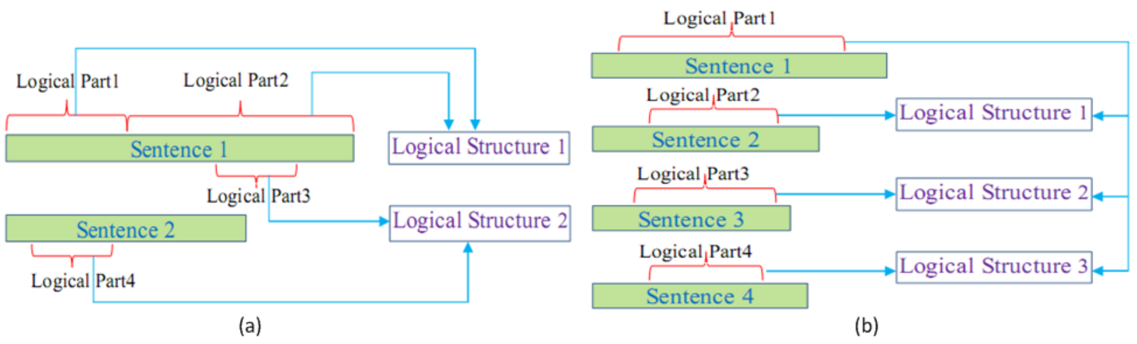


Figure 4.4: Two cases of inputs and outputs of the task.

method for substituting referent phrases. These works, however, only analyze logical structures of legal texts at the sentence level.

At the paragraph level, Takano et al. [130] classify a legal paragraph into one of six predefined categories: A, B, C, D, E, and F. Among six types, Type A, B, and C correspond to cases in which the main sentence is the first sentence, and subordinate sentences are other sentences. In paragraphs of Type D, E, and F, the main sentence is the first or the second sentence, and a subordinate sentence is an embedded sentence in parentheses within the main sentence.

4.3 Task Formulation

Analyzing logical structures of paragraphs in legal articles is the task of recognition of *logical structures* between *logical parts* in law sentences. A logical structure is formed from a pair of a *requisite part* and an *effectuation part*. These two parts are built from other kinds of logical parts such as *topic parts*, *antecedent parts*, *consequent parts*, and so on [4, 6][8]. Usually, consequent parts describe a law provision, antecedent parts describe cases in which the law provision can be applied, and topic parts describe subjects which are related to the law provision. In our work, a logical structure can be defined as a set of some related logical parts.

Figure 4.4 shows two cases of the inputs and outputs of the task. In the first case (a), the input is a paragraph of two sentences, and the outputs are four logical parts, which are grouped into two logical structures. In the second case (b), the input is a paragraph consisting of four sentences, and the outputs are four logical parts, which are grouped into three logical structures. We have two remarks:

1. A logical part may contain other logical parts. For example, in case (a), logical part 2 contains logical part 3.

2. A logical part can belong to multiple logical structures. For example, in case (b), logical part 1 belongs to three logical structures.

[8]We only recognize logical structures (a set of related logical parts). The task of translating legal articles into logical and formal representations is not covered in our work.

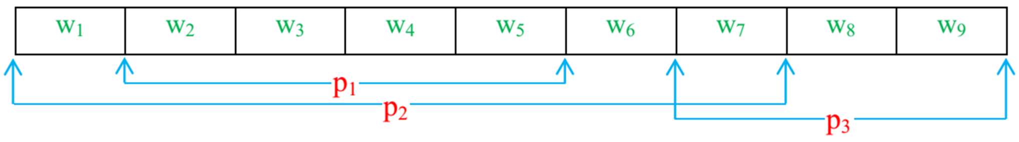


Figure 4.5: An example of *overlapping* and *embedded* relationships.

4.3.1 Subtask 1: Recognition of Logical Parts

Let s be a law sentence[9] in the space of law sentences S, then s can be represented by a sequence of words $s = [w_1 w_2 \ldots w_n]$. A legal paragraph x in the legal paragraph space X is a sequence of law sentences $x = [s_1 s_2 \ldots s_l]$, where $s_i \in S, \forall i = 1, 2, \ldots, l$. For each paragraph x, we denote a logical part p by a quad-tuple $p = (b, e, k, c)$ where b, e, and k are three integers which indicate *position of the beginning word*, *position of the end word*, and *sentence position* of p, and c is a logical part category in the set of predefined categories C. Formally, the set P of all possible logical parts defined in a paragraph x can be described as follows:

$$P = \{(b, e, k, c) | 1 \leq k \leq l, 1 \leq b \leq e \leq len(k), c \in C\}.$$

In the above definition, l is the number of sentences in the paragraph x, and $len(k)$ is the length of the k^{th} sentence.

In this subtask, we want to recognize some *non-overlapping* (but possibly *embedded*) logical parts in an input paragraph. A solution for this task is a subset $y \subseteq P$ which does not violate the *overlapping* relationship. We say that two logical parts p_1 and p_2 are *overlapping* if and only if they are in the same sentence ($k_1 = k_2$) and $b_1 < b_2 \leq e_1 < e_2$ or $b_2 < b_1 \leq e_2 < e_1$. We denote the *overlapping* relationship by $\sim$. We also say that p_1 is *embedded* in p_2 if and only if they are in the same sentence ($k_1 = k_2$) and $b_2 \leq b_1 \leq e_1 \leq e_2$, and denote the *embedded* relationship by $\prec$.

Figure 4.5 illustrates an example of *overlapping* and *embedded* relationships. In this example, the law sentence consists of nine words $w_1, \ldots, w_9$, and three logical parts p_1 (from 2 to 5), p_2 (from 1 to 7), and p_3 (from 7 to 9). Among three logical parts, p_2 and p_3 are *overlapping*, and p_1 is *embedded* in p_2.

Formally, the solution space can be described as follows:

$$Y = \{y \subseteq P | \forall u, v \in y, u \nsim v\}.$$

The learning problem in this subtask is to learn a function $R : X \rightarrow Y$ from a set of m training samples $\{(x^i, y^i) | x^i \in X, y^i \in Y, \forall i = 1, 2, \ldots, m\}$.

In our task, we consider the following types of logical parts:

1. An antecedent part is denoted by A
2. A consequent part is denoted by C
3. A topic part which depends on the antecedent part is denoted by T_1
4. A topic part which depends on the consequent part is denoted by T_2

[9] s may be a complete or non-complete sentence (clause or phrase).

5. A topic part which depends on both the antecedent part and the consequent part is denoted by T_3

6. The left part of an equivalent statement is denoted by EL

7. The right part of an equivalent statement is denoted by ER

8. An object part, whose meaning is defined differently in different cases, is denoted by Ob

9. An original replacement part, which will be replaced by other replacement parts (denoted by $RepR$) in specific cases, is denoted by $RepO$.

4.3.2 Subtask 2: Recognition of Logical Structures

In the second subtask, the goal is to recognize a set of logical structures given a set of logical parts. We recall that a logical structure is a set of some related logical parts.

Let $G =< V, E >$ be a complete undirected graph with the vertex set V and the edge set E. A real value function f is defined on E as follows:

$f : E \rightarrow R,\ e \in E \mapsto f(e) \in R.$

In this subtask, each vertex of the graph corresponds to a logical part, and a complete subgraph corresponds to a logical structure. The value on an edge connecting two vertices expresses the degree that the two vertices belong to one logical structure. The positive (negative) value means that two vertices are likely (not likely) to belong to one logical structure.

Let G_s be a complete subgraph of G, then $v(G_s)$ and $e(G_s)$ are the set of vertices and the set of edges of G_s, respectively. We define the total value of a subgraph as follows:

$$f(G_s) = f(e(G_s)) = \sum_{e \in e(G_s)} f(e).$$

Let Ω be the set of all complete subgraphs of G. The problem becomes determining a subset $\Psi \subseteq \Omega$ that satisfies the following constraints:

1. $\forall g \in \Psi, |v(g)| \geq 2$,

2. $\cup_{g \in \Psi} v(g) = V$,

3. $\forall g_1, g_2 \in \Psi | v(g_1) \subseteq v(g_2) \Rightarrow v(g_1) = v(g_2)$,

4. $\forall g \in \Psi, \cup_{h \in \Psi, h \neq g} v(h) \neq V$, and

5. $\sum_{g \in \Psi} f(g) \rightarrow$ maximize.

Constraint 1), *minimal constraint*, says that each logical structure must contain at least two logical parts. There is a case that a logical structure contains only a consequent part. Due to the characteristics of Japanese law sentences, however, our corpus does not contain such cases. A logical structure which contains a consequent part will also contain a topic part or an antecedent part or both of them. So a logical structure contains at least two logical parts. Constraint 2), *complete constraint*, says that each logical part must belong to at least one logical structure. Constraint 3), *maximal constraint*, says that we cannot have two different logical structures such that the set of logical parts in one logical structure contains the set of logical parts in the other logical structure. Constraint 4), *significant constraint*, says that if we remove any logical structure from the solution, Constraint 2) will be violated. Although Constraint 3) is guaranteed by Constraint 4), we introduce it because of its importance.

4.4 Framework Architecture

Figure 4.6 shows the framework architecture of our whole system. In learning process, the goal is to learn a multi-layer sequence model and a binary classifier. First, information about logical parts and logical structures is extracted from annotated Japanese National Pension Law (JNPL) corpus[10]. Then we conduct preprocessing using CaboCha tool [72]. The output of this process is then inputted into two feature extraction modules. Features extracted from sequence feature extraction module are used to learn multi-layer sequence model. To learn this model, we exploited Conditional random fields with *CRF++* tool [70]. Features extracted from relation feature extraction module are inputted into a machine learning model to learn the binary classifier. We present experimental results with two learning methods: Maximum Entropy model with *maxent* tool [138] and Support vector machines with *LIBSVM* tool [26, 56].

In testing process, the goal is to produce logical parts and logical structures given a new legal paragraph. First we conduct preprocessing on the input legal paragraph. It is then inputted into the sequence feature extraction module and multi-layer sequence learning model to produce logical parts. Logical parts are inputted into the relation feature extraction module and binary classifier to generate a weighted graph. From the weighted graph, an ILP formulation is then created. The solution of this ILP formulation produces a subgraph, from which logical structures are extracted using Bron-Kerbosch algorithm.

4.5 Multi-layer Sequence Learning for Logical Part Recognition

First, we give some related notions. Let s be a law sentence, and P be the set of logical parts of s, $P = \{p_1, p_2, \ldots, p_m\}$. $Layer^1(s)$ (outermost layer) is defined as a set of logical parts in P, which are not embedded in any other part. $Layer^i(s)$ is defined as a set of logical parts in $P \setminus \cup_{k=1}^{i-1} Layer^k(s)$, which are not embedded in any other part in $P \setminus \cup_{k=1}^{i-1} Layer^k(s)$. Formally, we have:

$$Layer^1(s) = \{p | p \in P, p \not\prec q, \forall q \in P, q \neq p\}.$$

$$Layer^i(s) = \{p | p \in Q^i, p \not\prec q, \forall q \in Q^i, q \neq p\},$$

where

$$Q^i = P \setminus \cup_{k=1}^{i-1} Layer^k(s)$$

[10]We will introduce JNPL corpus in Section 4.7.

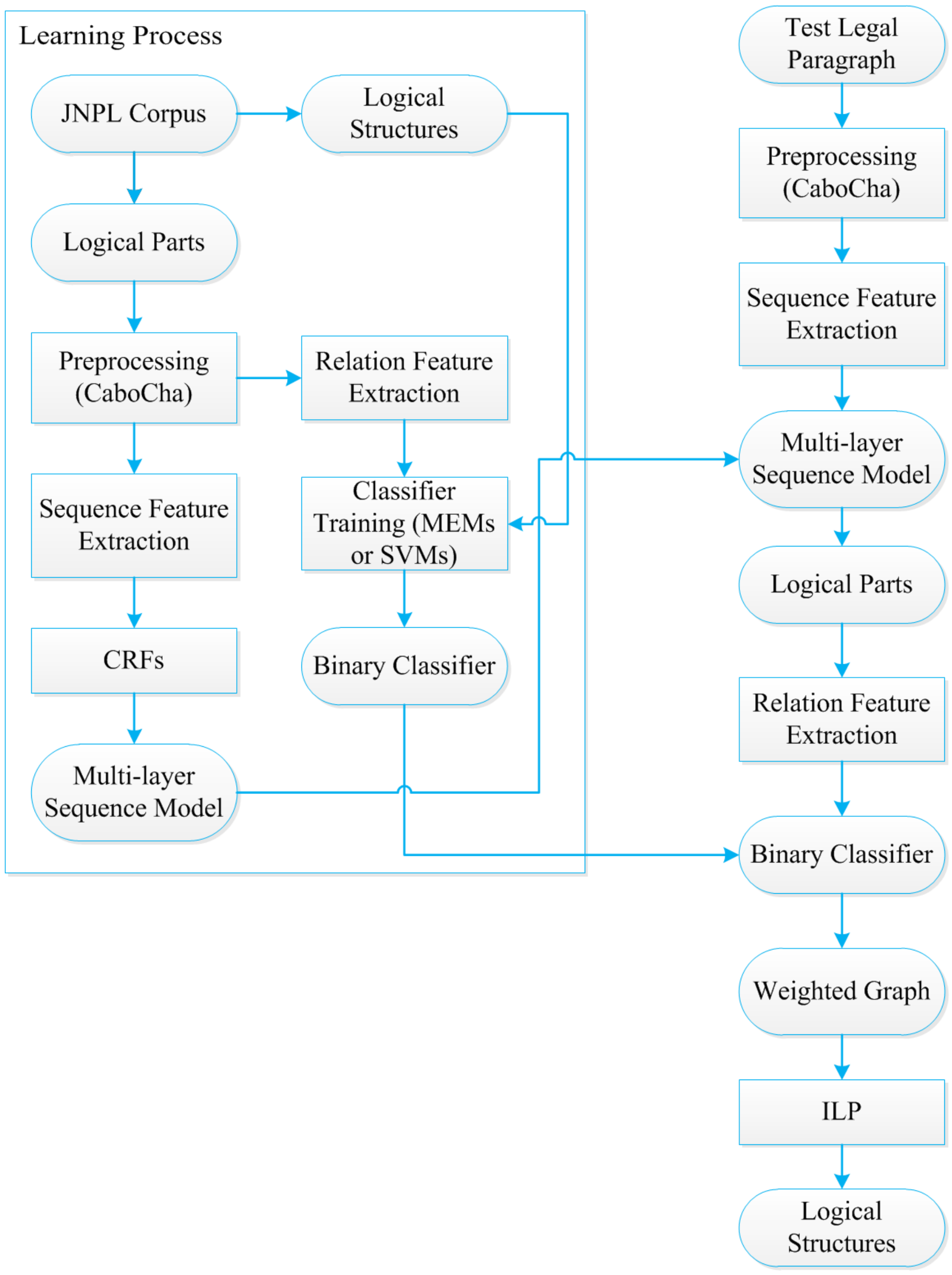


Figure 4.6: Framework architecture.

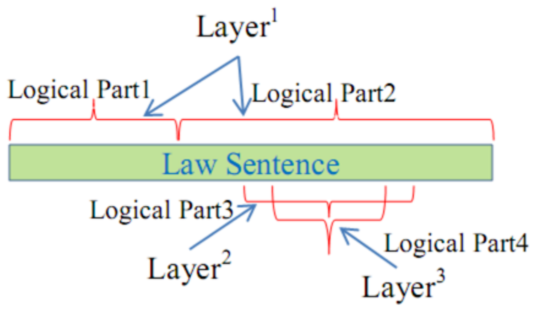
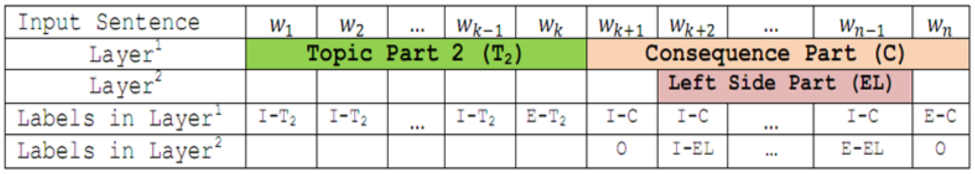


Figure 4.7: A law sentence with logical parts in three layers.



Input Sentence	w_1	w_2	...	w_{k-1}	w_k	w_{k+1}	w_{k+2}	...	w_{n-1}	w_n
Layer^1	**Topic Part 2 (T_2)**					**Consequence Part (C)**				
Layer^2							**Left Side Part (EL)**			
Labels in Layer^1	I-T_2	I-T_2	...	I-T_2	E-T_2	I-C	I-C	...	I-C	E-C
Labels in Layer^2						O	I-EL	...	E-EL	O

Figure 4.8: An example of labeling in the multi-layer model.

Figure 4.7 illustrates a law sentence with four logical parts in three layers: Part 1 and Part 2 in $Layer^1$, Part 3 in $Layer^2$, and Part 4 in $Layer^3$.

Let K be the number of layers in a law sentence s, our model will recognize logical parts in K steps. In the k^{th} step we recognize logical parts in $Layer^k$. In each layer, we model the recognition problem as a sequence labeling task in which each word is an element. Logical parts in $Layer^{i-1}$ will be used as input sequence in the i^{th} step (in the first step, we use original sentence as input).

Figure 4.8 gives an example of labeling for an input sentence. The sentence consists of three logical parts in two layers. In our model, we use IOE tag setting: the last element of a part is tagged with E, the other elements of a part are tagged with I, and an element not included in any part is tagged with O. According to [6], IOE tag setting is the most suitable setting for recognizing logical parts in Japanese law sentences. There are two reasons which may explain why the IOE setting is suitable for recognizing logical parts. First, in Japanese, important words usually occur at the end of a phrase, and important Bunsetsu usually occur at the end of a sentence. A Bunsetsu always depends on another Bunsetsu which stands to its right. Second, a logical part tends to finish at a punctuation mark (comma or period). So a punctuation mark has a high probability of being the last element of a logical part.

Let K^* be the maximum number of layers in all law sentences in training data. We learn K^* models, in which the k^{th} model is learned from logical parts in the $Layer^k$ of training data, using Conditional random fields [70, 75]. In the testing phase, we first apply the first model to the input law sentence, and then apply the i^{th} model to the predicted logical parts in $Layer^{i-1}$.

In our multi-layer sequence learning model, we choose CRFs as the learning method. There are some reasons why we choose CRFs. The first reason comes from the nature of the first subtask. This subtask is modeled as a multi-layer sequence learning problem, in which we learn a sequence learning model for each layer, and CRFs is a suitable

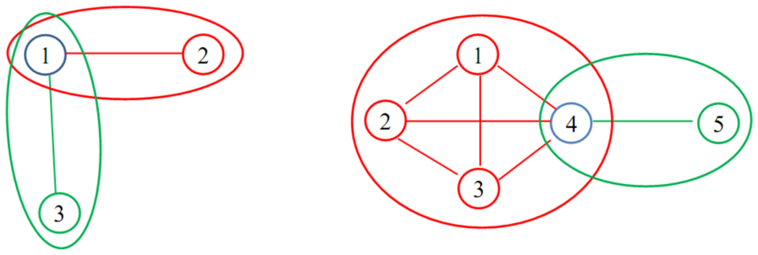


Figure 4.9: Examples of graphs and their cliques.

framework for sequence learning tasks. The second reason comes from the advantages of CRFs. CRFs is a discriminative method, it has all the advantages of Maximum Entropy Markov models (MEMMs) [85] but does not suffer from the label bias problem [75]. The last reason is that CRFs has been applied successfully to many NLP tasks such as POS tagging, chunking, named entity recognition, syntax parsing, information retrieval, information extraction, analyzing logical structures of legal texts at the sentence level, and so on [6, 73, 75, 106, 119].

4.6 ILP for Recognizing Logical Structures

This section describes our method for recognizing logical structures using integer linear programming (ILP).

4.6.1 ILP Formulation

Suppose that G' is a subgraph of G such that G' contains all the vertices of G and the degree of each vertex in G' is greater than zero, then the set of all the maximal complete subgraphs (or cliques) of G' will satisfy the *minimal*, *complete*, and *maximal* constraints, and also the *significant* constraint in most cases. We also note that, a set of cliques that satisfies all these four constraints will form a subgraph that has two properties like properties of G'.

Let Λ be the set of all such subgraphs G' of G, the subtask now consists of two steps:

1. Finding $G' = argmax_{G' \in \Lambda} f(G')$, and

2. Finding all cliques of G'.

Each clique found in the second step will correspond to a logical structure. Figure 4.9 illustrates two examples of graphs and their cliques. The graph on the left hand side consists of three nodes 1, 2, and 3, and two cliques {1, 2} and {1, 3}. The graph on the right hand side consists of five nodes 1, 2, 3, 4, and 5, and two cliques {1, 2, 3, 4} and {4, 5}.

Recently, some researches have shown that integer linear programming (ILP) formulations is an effective way to solve many NLP problems such as semantic role labeling [108], coreference resolution [36], summarization [27], dependency parsing [84], and so on. The advantage of ILP formulations is that we can incorporate non-local features or global constraints easily, which are difficult in traditional algorithms. Although solving an ILP is NP-hard in general, some fast algorithms and tools[11] are now available. So we can apply ILP to many NLP problems [84].

In this work, we exploit ILP to solve the first step. Let N be the number of vertices of G, we introduce a set of integer variables $\{x_{ij}\}_{1 \le i < j \le N}$. For a subgraph G' of G, the values of $\{x_{ij}\}$ are set as follows:

$$x_{ij} = \begin{cases} 1 & \text{if } (i,j) \in e(G'), \\ 0 & \text{otherwise.} \end{cases}$$

ILP formulations for the first step can be described:

Maximize:

$$\sum_{1 \le i < j \le N} f(i,j) * x_{ij}$$

Subject to:

$$Integer : \{x_{ij}\}_{1 \le i < j \le N}.$$

$$0 \le x_{ij} \le 1, (1 \le i < j \le N).$$

$$\sum_{i=1}^{j-1} x_{ij} + \sum_{k=j+1}^{N} x_{jk} \ge 1, (1 \le j \le N).$$

The last constraint guarantees that there is at least one edge connecting to each vertex in G'.

The second step, finding all cliques of an undirected graph, is a famous problem in graph theory. Many algorithms have been proposed to solve this problem efficiently. In this work, we exploit the Bron-Kerbosch algorithm, a backtracking algorithm. The main idea of the Bron-Kerbosch algorithm is using a branch-and-bound technique to stop searching on branches that cannot lead to a clique [17]. Although Bron-Kerbosch is a famous algorithm, we present it here for clarity.

First, we describe three sets, which play an important role in the algorithm[12].

1. The set *compsub*: contains the nodes already defined as a part of the clique.
2. The set *candidates*: contains all the nodes which serve as an extension to *compsub* (*candidates* is the set of nodes connected to all the nodes in *compsub*).
3. The set *not*: contains all the nodes that have already processed at an earlier stage, which lead to an extension of *compsub*, and are now excluded.

The pseudo code of the Bron-Kerbosch algorithm is presented as Algorithm 6. If there is an element in *not* connecting to all nodes in *candidates*, we cannot get a maximal clique from the present *compsub* (because we always miss that element in *not*). So the algorithm will terminate as early as possible.

The remaining problem is how to define the value function f. Our solution is that, first we learn a binary classifier C. This classifier takes a pair of logical parts as the input, and

[11]We used *lp-solve* from `http://lpsolve.sourceforge.net/`

[12]We use notations exactly the same as presented in Bron and Kerbosch [17].

Algorithm 6 Bron-Kerbosch Algorithm [17]

```
Input: An undirected graph $G =< V, E >$.
Output: Set of all cliques $C$.
Initialize: $compsub := \emptyset$, $candidates := V$, $not := \emptyset$, $C := \emptyset$;
Do the following recursive procedure until candidates is empty or there is an element
in not connecting to all nodes in candidates:

    1. Select a candidate $v$ in candidates

    2. Add $v$ to compsub:
       $compsub := compsub \cup \{v\}$;

    3. Compute new candidates and new not for the next recursion step
       $NewCandidates := \{u \in candidates | e(u, v) = 1\}$;
       $NewNot := \{u \in not | e(u, v) = 1\}$;

    4. Start the next recursion step (from (1)) with NewCandidates and NewNot

    5. Back from recursive with old sets of candidates and not, and make $v$ as processed
       (add $v$ to not): $not := not \cup \{v\}$;

If candidates and not are both empty, we have a clique which is a subgraph with the
set of nodes compsub
(add compsub to $C$): $C := C \cup \{compsub\}$;
```

outputs +1 if two logical parts belong to one logical structure, otherwise it will output −1. Then, we define the value function f for two logical parts p_i and p_j (correspond to node i and node j in the graph) as follows:

$$f(p_i, p_j) = Prob(C(p_i, p_j) = +1) - 0.5.$$

Function f will receive a value from −0.5 to +0.5.

4.6.2 Learning Binary Classifier

This section presents machine learning models and features that we use to learn the binary classifier C. Many classification methods have been proposed including traditional methods, such as k-NN [32], decision tree [115], naive Bayes [88], and more recent advanced models, like Maximum Entropy Models (MEMs) [12] and Support Vector Machines (SVMs) [140]. All of them can be used in our framework. Among these, we chose two classification methods to complete our framework: MEMs and SVMs. Both of them have been applied successfully to many NLP tasks. While SVMs is chosen because it is a very powerful method, MEMs is another good choice. It does not only performs better than SVMs in some particular cases, but also is very fast in both training and inference.

Features for Learning Binary Classifier

With a pair of logical parts, we extracted the following features (and combinations of them):

- Categories of two parts.
- Layers of two parts.
- The positions of the sentences that contain two parts (the first sentence or not).
- Categories of other parts in the input paragraph.

Table 4.1 shows features in details. Note that, in line number 22, we consider other logical parts in the input paragraph. Each such logical part will correspond to one feature. It is similar to line numbers 23, 24, and 25.

Table 4.1: Features for learning binary classifier (T2: topic part in case 2; A: antecedent part; C: consequent part)

#	Feature	Example
1	Category of the first logical part	A
2	Category of the second logical part	C
3	Categories of two logical parts	A-C
4	Layer of the first logical part	1
5	Layer of the second logical part	2
6	Layers of two logical parts	1-2
7	ID of the sentence (sentID) containing the first logical part	1
8	ID of the sentence (sentID) containing the second logical part	1
9	IDs of the sentences containing two logical parts	1-1
10	Category and layer of the first logical part	A-1
11	Category and layer of the second logical part	C-2
12	Categories and layers of two logical parts	A-C-1-2
13	Category and sentID of the first logical part	A-1
14	Category and sentID of the second logical part	C-1
15	Categories and sentIDs of two logical parts	A-C-1-1
16	Layer and sentID of the first logical part	1-1
17	Layer and sentID of the second logical part	2-1
18	Layers and sentIDs of two logical parts	1-2-1-1
19	Category, layer, and sentID of the first logical part	A-1-1
20	Category, layer, and sentID of the second logical part	C-2-1
21	Categories, layers, and sentIDs of two logical parts	A-C-1-2-1-1
22	Categories of other logical parts in the input	T2
23	Categories of the 1st logical part and other logical parts in the input	A-T2
24	Categories of the 2nd logical part and other logical parts in the input	C-T2
25	Categories of two logical parts and other logical parts in the input	A-C-T2

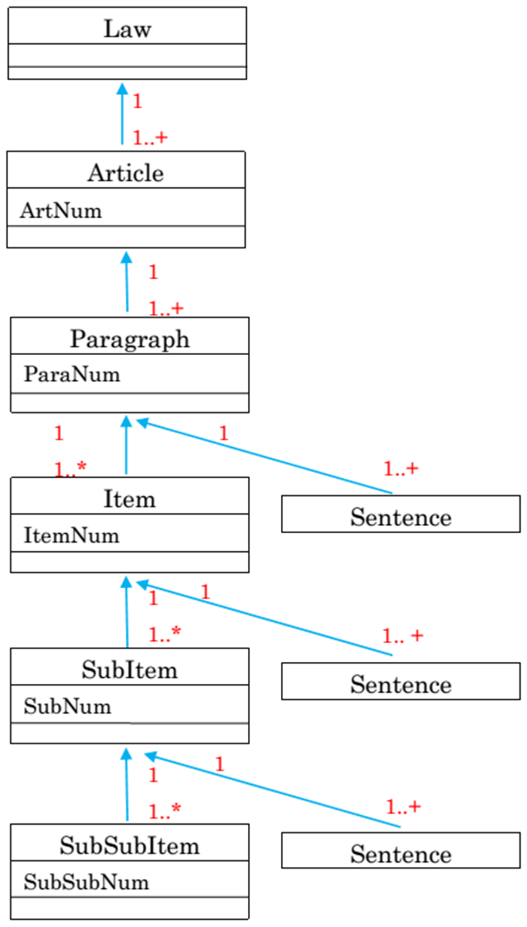


Figure 4.10: The structure of JNPL.

4.7 Experiments

4.7.1 Corpus

We have built a corpus, Japanese National Pension Law (JNPL) corpus, which consists of 83 legal articles[13] of Japanese national pension law. The structure of JNPL is shown in Figure 4.10. The law consists of articles, articles consist of paragraphs, and paragraphs contain sentences. A sentence may belong to items, subitems, or subsubitems of a paragraph.

Figure 4.11 shows the relationship between a law sentence and logical parts. A law sentence may contain some logical parts, and a logical part may be embedded in another one.

In our corpus, a logical part is annotated with information about its *type* (kind of part) and *formula-id* (logical parts with the same id will belong to one logical structure). An example of annotated sentence in the JNPL corpus is shown in Figure 4.12.

We employed two people[14] in a data-making company, who analyzed and annotated our corpus. The corpus consists of 83 legal articles, which contain 119 paragraphs with

[13]We annotate a main part of JNPL due to the resource limitation.
[14]A student and a graduated student of a law school.

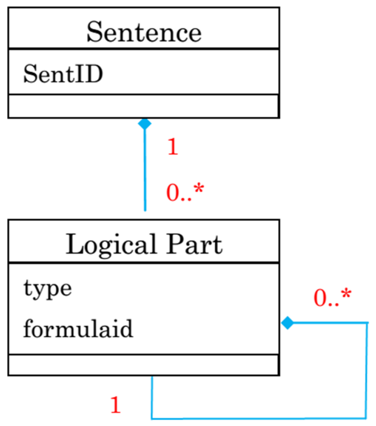


Figure 4.11: Relationship between a sentence and logical parts.

Table 4.2: Statistics on logical parts of the JNPL corpus

Logical Part	C	A	T_1	T_2	T_3	EL	ER	Ob	RepO	RepR
Number	248	286	0	114	12	55	57	9	12	14

426 sentences. On average, each paragraph consists of 3.6 sentences. The total number of logical parts is 807, and the number of logical structures is 351. On average, each paragraph consists of 6.8 logical parts and 3 logical structures.

We focus on paragraphs in Type A, B, and C defined in [130]. In those types, the first sentence of each paragraph is the main sentence, which usually contains more logical parts than other sentences. The other sentences often have a few logical parts, and in most cases these logical parts only appear in one layer. The first sentences usually contain logical parts in two layers.

Table 4.2 shows some statistics on the number of logical parts of each type. Main types of parts are A(35.4%), C(30.7%), T_2(14.1%), ER(7.1%), and EL(6.8%). Five main types of parts make up more than 94% of all types.

4.7.2 Evaluation Methods

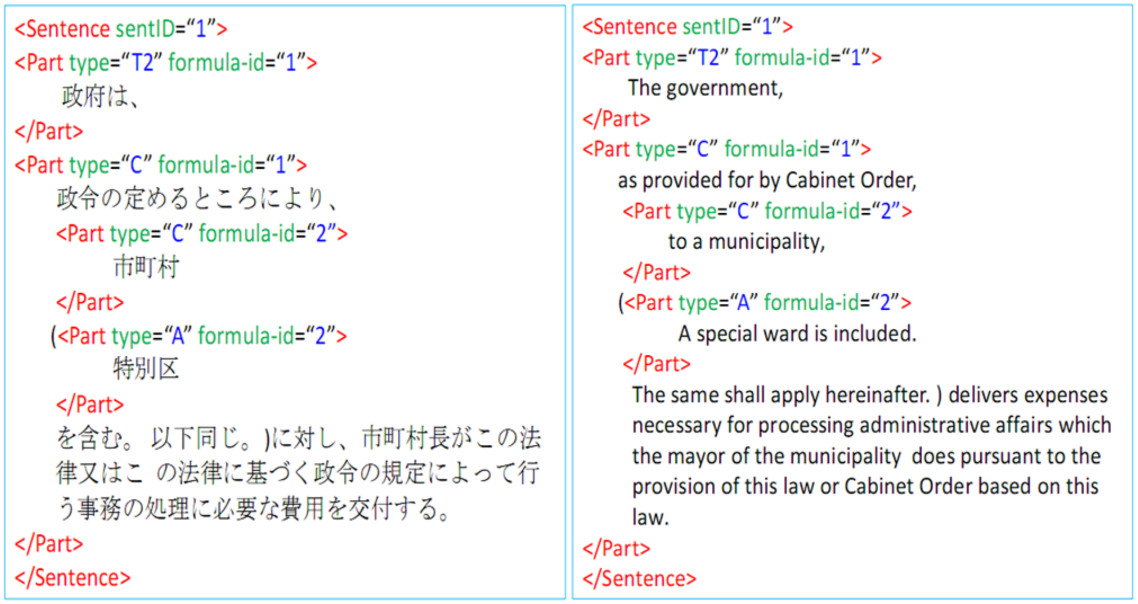
We divided the JNLP corpus into 10 sets, and conducted 10-fold cross-validation tests for all experiments in this chapter. For the first subtask, we evaluated the performance of our system by precision, recall, and F_1 scores as follows:

$$precision = \frac{number\ of\ correct\ logical\ parts}{number\ of\ predicted\ logical\ parts},$$

$$recall = \frac{number\ of\ correct\ logical\ parts}{number\ of\ actual\ logical\ parts},$$

$$F_1 = \frac{2 * precision * recall}{precision + recall}.$$

For the second subtask, we used MUC precision, recall, and F_1 scores as described in [142]. We summarize them here for clarity.



```
<Sentence sentID="1">
<Part type="T2" formula-id="1">
    政府は、
</Part>
<Part type="C" formula-id="1">
    政令の定めるところにより、
    <Part type="C" formula-id="2">
        市町村
    </Part>
    (<Part type="A" formula-id="2">
        特別区
    </Part>
    を含む。 以下同じ。)に対し、市町村長がこの法
    律又はこ の法律に基づく政令の規定によって行
    う事務の処理に必要な費用を交付する。
</Part>
</Sentence>
```

```
<Sentence sentID="1">
<Part type="T2" formula-id="1">
    The government,
</Part>
<Part type="C" formula-id="1">
    as provided for by Cabinet Order,
    <Part type="C" formula-id="2">
        to a municipality,
    </Part>
    (<Part type="A" formula-id="2">
        A special ward is included.
    </Part>
    The same shall apply hereinafter. ) delivers expenses
    necessary for processing administrative affairs which
    the mayor of the municipality  does pursuant to the
    provision of this law or Cabinet Order based on this
    law.
</Part>
</Sentence>
```

Figure 4.12: An annotated sentence in the JNPL corpus. (left: the original text; right: the translated text.)

Table 4.3: Examples of evaluation method for Subtask 2

Input	Predict	Gold	Recall	Precision	F_1
1,2,3,4,5	{1,2,3} {1,4} {2,5}	{1,2,3,4} {1,5}	$\frac{(4-2)+(2-2)}{(4-1)+(2-1)} = 0.50$	$\frac{(3-1)+(2-1)+(2-2)}{(3-1)+(2-1)+(2-1)} = 0.75$	0.60
1,2,3,4,5	{1,2,3} {1,4} {2,5}	{1,2,3} {1,5} {2,4}	$\frac{(3-1)+(2-2)+(2-2)}{(3-1)+(2-1)+(2-1)} = 0.50$	$\frac{(3-1)+(2-2)+(2-2)}{(3-1)+(2-1)+(2-1)} = 0.50$	0.50

Let $P_1, P_2, \dots, P_n$ be n predicted logical structures, and $G_1, G_2, \dots, G_m$ be the correct answers or gold logical structures. To calculate the recall, for each gold logical structure $G_i (i = 1, 2, \dots, m)$, let $k(G_i)$ be the smallest number such that there exist $k(G_i)$ predicted structures $P_1^i, P_2^i, \dots, P_{k(G_i)}^i$ which satisfy $G_i \subseteq \cup_{j=1}^{k(G_i)} P_j^i$:

$$recall = \frac{\sum_{i=1}^{m} (|G_i| - k(G_i))}{\sum_{i=1}^{m} (|G_i| - 1)}.$$

To calculate the precision, we switch the roles of predicted structures and gold structures. Finally, F_1 score is computed in a similar manner as in the first subtask.

Table 4.3 shows two examples of the evaluation method for Subtask 2. In two examples, we have five input logical parts numbered 1, 2, 3, 4, and 5, and the system predicts three logical structures {1, 2, 3}, {1, 4}, and {2, 5}. In the first case, the correct answer (gold) consists of two logical structures {1, 2, 3, 4} and {1, 5}, while in the second case, the correct answer consists of three logical structures {1, 2, 3}, {1, 5}, and {2, 4}.

To score the whole system, due to the predicted logical parts may differ from the correct logical parts, we need to modify the MUC scores. Let $P_1, P_2, \dots, P_n$ be n predicted logical structures, and $G_1, G_2, \dots, G_m$ be the gold logical structures. For each gold logical

Table 4.4: Examples of evaluation method for the whole system (PreP = Predicted logical parts, PreStr = Predicted logical structures, GP = Gold logical parts, GStr = Gold logical structures)

PreP	PreStr	GP	GStr	Recall	Precision	F_1
1,2,3	{1,2} {1,3}	1,2,3,4	{1,2,4} {3,4}	$\frac{(3-1-1)+(2-1-1)}{(3-1)+(2-1)}$=0.33	$\frac{(2-0-1)+(2-0-2)}{(2-1)+(2-1)} = 0.50$	0.40
1,2,3	{1,3} {2,3}	1,2,4	{1,4} {2,4}	$\frac{(2-1-1)+(2-1-1)}{(2-1)+(2-1)}$=0.00	$\frac{(2-1-1)+(2-1-1)}{(2-1)+(2-1)} = 0.00$	0.00

structure $G_i(i = 1, 2, \ldots, m)$, let D_i be the set of logical parts in G_i which are not included in the set of predicted logical parts. $D_i = \{p \in G_i | p \notin \cup_{j=1}^{n} P_j\}$. Let $k(G_i)$ be the smallest number such that there exist $k(G_i)$ predicted structures $P_1^i, P_2^i, \ldots, P_{k(G_i)}^i$ which satisfy $G_i \subseteq (\cup_{j=1}^{k(G_i)} P_j^i) \cup D_i$.

$$recall = \frac{\sum_{i=1}^{m} (|G_i| - |D_i| - k(G_i))}{\sum_{i=1}^{m} (|G_i| - 1)}.$$

To calculate the precision, we switch the roles of predicted structures and gold structures.

Two examples of the evaluation method for the whole system are shown in Table 4.4. In the first case, the set of predicted logical parts (output predicted by our system in Subtask 1) consists of three logical parts 1, 2, and 3, while the correct logical parts are 1, 2, 3, and 4. Logical part 4 is not included in the set of predicted logical parts. Therefore, when calculating recall, we need to subtract 1 from each factor in the numerator. In the second case, the set of predicted logical parts is unchanged, while the correct logical parts are 1, 2, and 4. The set of predicted logical parts does not contain logical part 4, and the set of gold logical parts does not contain logical part 3. Hence, we need to subtract 1 from each factor in the numerator when calculating both recall and precision.

4.7.3 Experiments on Subtask 1

Baseline: Filter-Ranking Perceptron Algorithm

We chose the Filter-Ranking (FR) Perceptron algorithm proposed by Carreras et al.,[22, 23] as our baseline model because of its effectiveness on phrase recognition problems, especially on problems that accept the *embedded* relationship[15]. We use FR-perceptron algorithm to recognize logical parts in law sentences one by one in an input paragraph.

The idea of the FR-perceptron algorithm is to build a recognition model with two components. The first component, operating at word level, is a *filtering* function F, which identifies a set of candidate logical parts for an input law sentence s, $F(s) \subseteq P$ (P is the set of all possible logical parts). The second one, operating at part level, is a *score* function which produces a real value score for a logical part. The recognizer will use the score function to search an optimal coherent subset from the candidate set $F(s)$.

$$R(s) = argmax_{y \subseteq F(s) | y \in Y} \sum_{u \in y} score(u, s, y).$$

[15]We re-implement the FR-perceptron algorithm by ourselves.

The *filtering* component F is only used to reduce the search space. Instead of searching in the space P, the R function only searches in a subset $F(s)$ of P. The setting for function F is a *begin-end* classification for each logical part category: a word is considered as *c-begin* if it is likely to begin a *category-c* logical part, and as *c-end* if it is likely to end a *category-c* logical part. Each pair of *c-begin* word w_b and *c-end* word w_e forms a logical part candidate (b, e, k, c), where k is the index of the sentence s in its paragraph. Suppose that h_b^c and h_e^c are begin and end classification functions for each category c, the filtering function F can be described as follows:

$$F(s) = \{(b, e, k, c) | h_b^c(w_b, s, k) = +1 \wedge h_e^c(w_e, s, k) = +1\}.$$

FR-perceptron algorithm uses linear functions for the begin and end classification functions in the filtering component, and the score function in the recognition component. To learn parameter vectors for these functions, a perceptron-like algorithm was introduced [22, 23]. The main advantage of FR-perceptron algorithm is that it can learn parameters for both classification functions and score function simultaneously. Moreover, using dynamic programming for inference is a guarantee for finding the best solution.

For *begin/end* predictors, we got features of words, POS tags, and Bunsetsu tags. In Japanese, a sentence is divided into some chunks called Bunsetsu. Each Bunsetsu includes one or more content words (noun, verb, adjective, etc) and may include some function words (case-marker, punctuation, etc) [93]. The Bunsetsu tag of a word indicates whether the word appears at the beginning of a Bunsetsu or not. We obtained following features in a window size 2:

- $f[-2]$, $f[-1]$, $f[0]$, $f[+1]$, $f[+2]$ (for words, POS tags, and Bunsetsu tags),
- $f[-2]f[-1]$, $f[-1]f[0]$, $f[0]f[+1]$, $f[+1]f[+2]$, $f[-2]f[-1]f[0]$, $f[-1]f[0]f[+1]$, $f[0]f[+1]f[+2]$ (for words and POS tags).

For example, if f is word feature then $f[0]$ is the current word, $f[-1]$ is the preceding word, and $f[-1]f[0]$ is the co-occurrence of them. Moreover, with *begin* predictor, we use a feature for checking whether this position is the beginning of the sentence or not. Similarly, with *end* predictor, we use a feature for checking whether this position is the end of the sentence or not.

With each logical part candidate, we extract following kinds of features:

1. Length of the logical part,
2. Internal structure: this feature is the concatenation of the top logical parts, punctuation marks, parenthesis, and quotes inside the candidate. An example about internal structure may be $(A+, +C+.)$ (plus is used to concatenate items). This means that the candidate consists of an antecedent part, a comma, a consequent part, and a period at the end.
3. Uni-gram of words and part-of-speech tags,
4. Bi-gram of words and part-of-speech tags, and
5. Tri-gram of words and part-of-speech tags.

Table 4.5: Experimental results of the baseline model

Logical Part	Prec(%)	Recall(%)	F_1(%)
C	**80.10**	**61.69**	**69.70**
EL	82.35	50.91	62.92
ER	66.67	21.05	32.00
Ob	0.00	0.00	0.00
A	**79.10**	**55.59**	**65.30**
RepO	100	16.67	28.57
RepR	0.00	0.00	0.00
T_2	**82.14**	**60.53**	**69.70**
T_3	50.00	8.33	14.29
Overall	**79.70**	**52.54**	**63.33**

Exeperimental results of the baseline model are shown in Table 4.5. The baseline model achieved pretty good results in three main types of parts: C(69.70%), A(65.30%), and T_2(69.70%). In all types of parts, the precision score was better than the recall score (80.10% and 61.69% in type C; 82.35% and 50.91% in type EL; 66.67% and 21.05% in type ER; 79.19% and 55.59% in type A; 100% and 16.67% in type RepO; 82.14% and 60.53% in type T_2; and 50.00% and 8.33% in type T_3). Overall, the model achieved 79.70% in the precision score, but only 52.54% in the recall score. These results led to a low F_1 score (63.33%).

Experimental Results of Multi-Layer Sequence Learning Model

We focused on paragraphs in Type A, B, and C defined in [130], where the first sentences of paragraphs usually contain logical parts in two layers, and other sentences contain logical parts in one layer. We divided sentences into two groups. The first group consists of the first sentences in paragraphs, and the second group consists of other sentences. We set the number of layers k to 2 for sentences in the first group, and to 1 for sentences in the second group. To learn sequence labeling models, we used CRFs [70, 75].

Experimental results on the JNPL corpus are described in Table 4.6. We conducted experiments with four feature sets: words; words and POS tags; words and Bunsetsu tags; and words, POS tags, and Bunsetsu tags. To extract features from source sentences, we used the CaboCha tool [72], a Japanese morphological and syntactic analyzer. The best model (the model used word and Bunsetsu tag features) achieved 74.37%in the F_1 score. It improves 11.04% in the F_1 score (30.11% in error rate) compared with the baseline model.

Table 4.7 shows experimental results of our best model in more detail. Our model got good results on most main parts: C(78.98%), A(80.42%), and T_2(82.14%). The model got low results on the other types of parts. It is understandable because three types of logical parts C, A, and T_2 make up more than 80%, while six other types only make up 20% of all types. Similar to the baseline model, the precision score was better than the recall score in all types of parts. However, the recall score was improved significantly (69.76% compared with 52.52%).

The high precision in some types of logical parts (RepO and RepR) does not mean that the system recognized well these types. Due to the number of logical parts of these

Table 4.6: Experimental results for Subtask 1 on the JNLP corpus(W:Word; P: POS tag; B: Bunsetsu tag)

Model	Prec(%)	Recall(%)	F_1(%)	Improvement(%)
Baseline	**79.70**	52.54	63.33	-
W	79.18	69.27	73.89	10.56
W+P	77.62	68.77	72.93	9.60
W+B	79.63	**69.76**	**74.37**	**11.04**
W+P+B	77.89	69.39	73.39	10.06

Table 4.7: Experimental results in more details

Logical Part	Prec(%)	Recall(%)	F_1(%)
C	**83.41**	**75.00**	**78.98**
EL	76.74	60.00	67.35
ER	41.94	22.81	29.55
Ob	0.00	0.00	0.00
A	**80.42**	**80.42**	**80.42**
RepO	100	16.67	28.57
RepR	100	28.57	44.44
T_2	**83.64**	**80.70**	**82.14**
T_3	60.00	25.00	35.29
Overall	**79.63**	**69.76**	**74.37**

types is very small, in the test phase the system only predicted a few logical parts of such types. In the case it predicted correctly, the precision was very high but the recall was very low.

4.7.4 Experiments on Subtask 2

Baseline: a Heuristic Algorithm

Our baseline is a heuristic algorithm to solve this subtask on graphs. This is an approximate algorithm which satisfies the *minimal*, *complete*, *maximal*, and *significant* constraints (described in Section 4.3). The main idea of our algorithm is picking up as many positive edges as possible, and as few negative edges as possible. We consider two cases:

1. There is no positive value edge on the input graph, and
2. There are some positive value edges on the input graph.

The heuristic algorithm in the first case is presented as Algorithm 7. In this case, because all the edges have negative values, we build logical structures with as few logical parts as possible. In this case, each logical structure contains exactly two logical parts. So we gradually choose two nodes in the graph with the maximum value on the edge connecting them.

An example of the first case is illustrated in Figure 4.13. The maximum value on an edge is -0.1, so the first logical structure will contain node 1 and node 3. The second

Algorithm 7 Heuristic Algorithm in the First Case

```
1:  Input: An undirected graph $G =< V, E >$.
2:  Output: Set of logical structures $L$.
3:  Initialize: $L := \emptyset$, $V_1 := V$;
4:  while $V_1 \neq \emptyset$ do
5:      if $|V_1| \geq 2$ then
6:          $(u, v) := argmax_{u \neq v \in V_1} f(u, v)$;
7:      else
8:          Let $v$ be the element in $V_1$.
9:          $u := argmax_{u \in V} f(u, v)$;
10:     end if
11:     Add $\{u, v\}$ to $L$: $L := L \cup \{\{u, v\}\}$;
12:     Update $V_1$: $V_1 := V_1 \setminus \{u, v\}$;
13: end while
14: Return $L$;
```

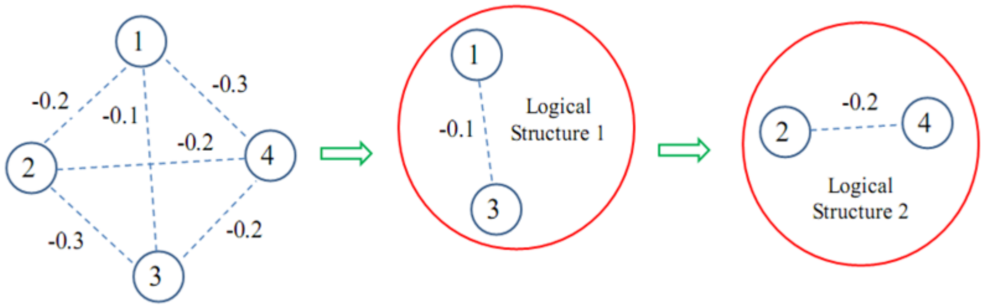


Figure 4.13: An example of the first case.

logical structure contains node 2 and node 4[16].

The main idea of the heuristic algorithm in the second case is described as Algorithm 8. Note that, $e(G)$ represents the set of edges and $v(G)$ represents the set of nodes in a graph G. In this case, we first consider the subgraph which only contains non-negative value edges. In this subgraph, we repeatedly build logical structures with as many logical parts as possible. After building successfully a logical structure, we remove all the nodes and the edges constituting the logical structure. When we have no positive edge, we will build logical structures with exactly two logical parts.

An example of the second case is illustrated in Figure 4.14. First, we consider the subgraph with positive edges (graph G_1 in the algorithm). This subgraph consists of five nodes $\{1, 2, 3, 4, 5\}$ and four edges $\{(1, 2), (1, 3), (2, 3), (2, 4)\}$. First, we have a logical structure with three nodes $\{1, 2, 3\}$. We remove these nodes and the positive edges connecting to these nodes. We have two nodes $\{4, 5\}$ with no positive edges. Now we build logical structures with exactly two nodes.

The set V' consists of nodes that do no appear in the logical parts. In this case, $V' = \{4, 5\}$. We divide V' into two sets V_1 and V_2. The set V_1 contains nodes that have one positive edge connecting to them, and the set V_2 contains nodes that all edges

[16]If the number of nodes is odd, the final logical structure will consist of the final node and another node, so that the edge connecting them has the maximal value.

Algorithm 8 Heuristic Algorithm in the Second Case

```
1: **Input:** An undirected graph $G =< V, E >$.
2: **Output:** Set of logical structures $L$.
3: **Initialize**: $L := \emptyset$;
4: Let $G_1 =< V, E^+ >$ be the subgraph of $G$ that only contains edges with non-negative
   values.
5: while $e(G_1) \neq \emptyset$ do
6:     Let $g$ be the complete subgraph of $G_1$ that maximizes $f(g)$.
7:     Add $g$ to $L$: $L := L \cup \{g\}$;
8:     Remove $g$ and edges connecting to a vertex in $g$ from $G_1$;
9: end while
10: Let $V'$ be the set of nodes which do not appear in any logical part in $L$:
11: $V' := \{v \in V | v \notin \bigcup_{g \in L} v(g)\}$;
12: Let $V_1$ be the set of nodes in $V'$ that have at least one non-negative edge connecting
    to them:
13: $V_1 := \{v \in V' | \exists u \in G, f(u, v) \geq 0\}$;
14: Let $V_2$ be the set of nodes in $V'$ such that all edges connecting to them have negative
    scores:
15: $V_2 := V' \backslash V_1$;
16: while $V_1 \neq \emptyset$ do
17:     $(u, v) := argmax_{u \in V, v \in V_1} f(u, v)$;
18:     Add $\{u, v\}$ to $L$: $L := L \cup \{\{u, v\}\}$;
19:     Update $V_1$: $V_1 := V_1 \backslash \{u, v\}$;
20: end while
21: while $V_2 \neq \emptyset$ do
22:     if $|V_2| \geq 2$ then
23:         $(u, v) := argmax_{u \neq v \in V_2} f(u, v)$;
24:     else
25:         Let $v$ be the element in $V_2$.
26:         $u := argmax_{u \in V} f(u, v)$;
27:     end if
28:     Add $\{u, v\}$ to $L$: $L := L \cup \{\{u, v\}\}$;
29:     Update $V_2$: $V_2 := V_2 \backslash \{u, v\}$;
30: end while
31: Return $L$;
```

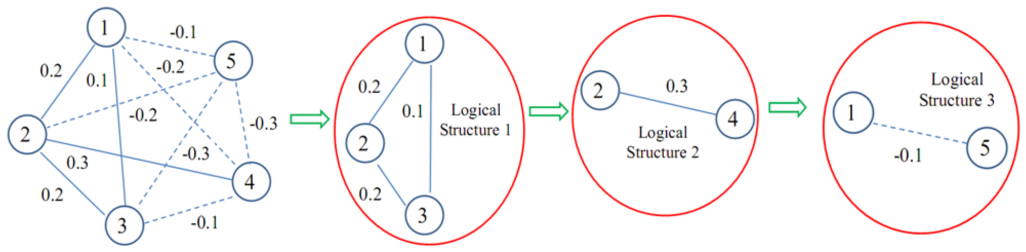


Figure 4.14: An example of the second case.

Table 4.8: Experiments on Subtask 2 (Gold-Input Setting)

Model	Precision(%)	Recall(%)	F_1(%)	Improvement(%)
MEMs				
Heuristic	**81.24**	71.19	75.89	-
ILP	76.56	**82.87**	**79.59**	**3.70**
SVMs (RBF Kernel)				
Heuristic	83.04	68.15	74.86	-
ILP	**78.97**	**81.22**	**80.08**	**5.22**

connecting to them have negative scores. In this case, $V_1 = \{4\}$ and $V_2 = \{5\}$. We consider node 4. Among edges connecting to node 4, the edge $(2,4)$ has maximal value. So we have the second logical structure with two nodes $\{2,4\}$. Next, we consider node 5, and we have the third logical structure with two nodes $\{1,5\}$.

Experimental Results

We conducted experiments on this subtask in two settings. In the first setting (gold-input setting), we used annotated logical parts as the inputs to the system. The purpose of this experiment is to evaluate the performance of the graph-based method on Subtask 2. We used MUC precision, recall, and F_1 scores to evaluate results in this setting. In the second setting (whole-task setting), predicted logical parts outputted by the Subtask 1 were used as the inputs to the system. The purpose of this experiment is to evaluate the performance of our framework on the whole task. We used a modification version of MUC precision, recall, and F_1 scores (presented in Section 4.7.2) to evaluate results in this setting.

Table 4.8 shows experimental results on the second subtask in the first setting (gold-input setting). The proposed method using ILP formulation outperformed the baseline (heuristic) algorithm in both experiments with MEMs and SVMs. When using MEMs as the classification method, our proposed model achieved 79.59% in the F_1 score while the baseline model only achieved 75.89% (improved 3.70% in the F_1 score, 15.35% in error rate). When using SVMs with the RBF kernel as classification method, the proposed model achieved 80.08% in the F_1 score comparing with 74.86% of the baseline model (improved 5.22% in the F_1 score, 20.76% in error rate).

Table 4.9 shows experimental results on the second subtask in the second setting (whole-task setting). Once again, the proposed model using ILP formulation outperformed the baseline model in both experiments with MEMs and SVMs. When using

Table 4.9: Experiments on Subtask 2 (Whole-Task Setting)

Model	Precision(%)	Recall(%)	F_1(%)	Improvement(%)
MEMs				
Heuristic	54.88	47.84	51.12	-
ILP	**57.51**	**54.06**	**55.73**	**4.61**
SVMs (RBF Kernel)				
Heuristic	55.81	46.07	50.47	-
ILP	**62.73**	**54.57**	**58.36**	**7.89**

MEMs as the classification method, our proposed model achieved 55.73% in the F_1 score while the baseline model only achieved 51.12% (improved 4.61% in the F_1 score, 9.43% in error rate). When using SVMs with the RBF kernel as classification method, the proposed model achieved 58.36% in the F_1 score comparing with 50.47% of the baseline model (improved 7.89% in the F_1 score, 15.93% in error rate).

4.7.5 Limitation & Improvement

This section discusses about our framework, the limitation of the system, and introduces simple methods to improve the performance of the system.

A Joint Decoding Algorithm for Logical Part Recognition

We proposed a pipeline framework, where the output of the first subtask will be used as the input of the second subtask. Specifically, logical parts outputted from the first subtask are inputted to the second subtask to build logical structures. The first step will affect to the second step, and therefore contribute significantly to the quality of the whole system. Experimental results showed that, our system achieved 80.08% in F_1 score when using correct logical parts, while only achieved 58.36% in F_1 score when using logical parts recognized in the first subtask. Hence, in this section we focus on the first subtask, *Recognition of logical parts*.

We take into account the question: How to improve the first subtask, and therefore improve the quality of the whole system?

In our multi-layer sequence learning model, logical parts recognized in the first layer will be used as the input sequence to recognize logical parts in the second layer. The limitation of this method is that, logical parts in different layers are recognized independently. When we recognize logical parts in the second layer, we can access information about logical parts in the first layer. When recognizing logical parts in the first layer, however, we have no information from logical parts in the second layer. Information is transmitted only in one direction.

A possible solution for this problem is follows. When recognizing logical parts in the first layer, we store the N-best outputs (we call them *candidates*) and use them to recognize logical parts in the second layer. The candidate that produces the best result (considering outputs in both layers) will be selected. By doing this, we can indirectly access information from logical parts in the second layer when recognizing logical parts in the first layer. A simple version of this method is the joint decoding algorithm for multi-layer sequence learning model, which is presented as Algorithm 9. In this algorithm, the

Table 4.10: Experimental results on Subtask 1 using Joint Decoding Algorithm (W:Word; P: POS tag; B: Bunsetsu tag). N = 1, 2, 3, 4, 5, 10, 20, 50, 100, and 1000.

Features	N	Precision(%)	Recall(%)	F_1(%)	Improvement(%)
W	1	**79.18**	69.27	**73.89**	-
	> 1	79.07	69.27	73.84	-0.05
W+P	1	77.62	68.77	72.93	-
	> 1	**77.98**	**69.76**	**73.64**	**+0.71**
W+B	1	79.63	69.76	74.37	-
	2	**80.39**	**70.63**	**75.20**	**+0.83**
	> 2	80.23	70.38	74.98	+0.61
W+P+B	1	**77.89**	69.39	73.39	-
	> 1	77.62	**69.64**	**73.42**	**+0.03**

condition to select the best candidate is maximizing the joint probability, which is the product of probabilities outputted by recognition models when recognizing logical parts in two layers.

Algorithm 9 Joint Decoding Algorithm for Multi-layer Sequence Learning Model

```
1: Input: A legal paragraph $p$ and a number of candidates $N$.
2: Output: Set of logical parts $L$.
3: Initialize: $L := \emptyset$.
4: for each sentence $s$ in paragraph $p$ do
5:    Recognize logical parts in the first layer of $s$.
6:    Store the N-best outputs (candidates) and their probabilities.
7:    for each candidate $c$ in the N-best candidates do
8:       Recognize logical parts in the second layer of $c$.
9:       Calculate the joint probability of $c$ (the product of probabilities in two layers).
10:   end for
11:   Select candidate $c^*$ with the maximum joint probability.
12:   Add logical parts in $c^*$ (in both layers) to $L$.
13: end for
14: Return $L$.
```

We conducted experiments on Subtask 1 using the joint decoding algorithm with different values of N (1, 2, 3, 4, 5, 10, 20, 50, 100, and 1000). Experimental results are shown in Table 4.10. When N equals to 1, it becomes the separately decoding algorithm. When N equals to 2, the joint decoding algorithm improved a little bit (0.71% when using word and POS tag features, 0.83% when using word and Bunsetsu tag features, and 0.03% when using word, POS tag, and Bunsetsu tag features). However, there is no difference when we continue to increase the value of N.

A natural question is that whether or not the list of the N-best outputs (the N-best list) contains the correct output? If the answer is *No*, we cannot find the correct result by this method. To answer this question, we conducted experiments in oracle setting. In this setting, if the N-best list contains the correct output, we assume that the system can find the output. Otherwise, the system returns the output with the highest probability.

Experimental results in oracle setting are presented in Table 4.11. When we used

Table 4.11: Oracle results on Subtask 1 (the feature set consists of words and Bunsetsu tags)

N	Precision(%)	Recall(%)	F_1(%)	Improvement(%)
1	79.63	69.76	74.37	-
10	83.26	73.36	78.00	3.63
50	83.50	73.36	78.10	3.73
100	84.20	73.98	78.76	4.39
1000	84.81	74.72	**79.45**	**5.08**
10000	84.81	74.72	**79.45**	**5.08**

1000-best list, we got 79.45% in F_1 score (improved 5.08% compared with 1-best list). However, the result was the same for 10000-best list. From experimental results, we have some conclusions as follows:

- The upper bound of the multi-layer sequence learning model is about 80% in F_1 score.
- In most cases, the 1000-best list will contain the correct answer. If the correct answer is out of the 1000-best list, we have very little chance to find it.

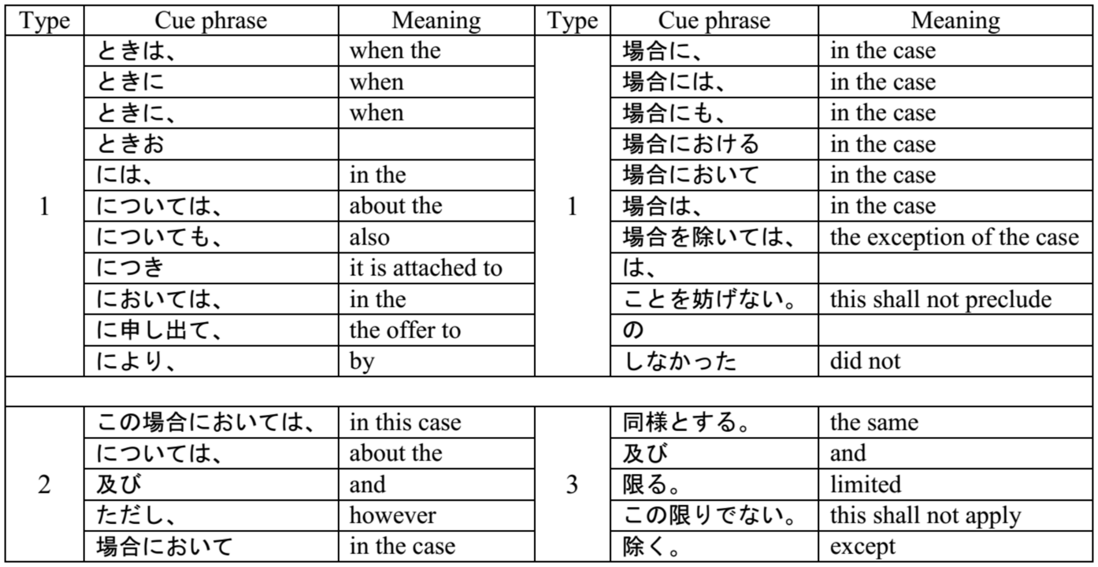
Cue Phrases as Context Features

In our framework, to assign weights for the edges connecting two logical parts, we learned a binary classifier that predicts whether the two logical parts belong to the same logical structure or not. The features used in that classifier includes categories and layers of the two logical parts, the positions of the sentences that contain two logical parts, categories of other logical parts in the input paragraph, and the combination of them.

In this section, we investigate the problem of using context features, the lexical features extracted from words inside or surrounding logical parts, to improve the performance of the recognition system. Based on our analysis of legal texts, we build a list of phrases which usually appear together with logical parts, and use them as context features for learning the binary classifier. Figure 4.15 shows the list of cue phrases and their meanings in English. These cue phrases can be divided into three types based on the positions they appear (corresponding to three types of context features):

1. Cue phrases that appear at the end of a logical part (Type 1),
2. Cue phrases that appear right before a logical part (Type 2), and
3. Cue phrases that appear right after a logical part (Type 3).

We conducted experiments with context features in two settings: using gold logical parts and using our recognized logical parts (full system). Experimental results are shown in Table 4.12. We can see that using context features improved the performance of the system in both settings (2.23% and 0.73% in the F_1 scores, respectively).



Type	Cue phrase	Meaning	Type	Cue phrase	Meaning
1	ときは、	when the	1	場合に、	in the case
	ときに	when		場合には、	in the case
	ときに、	when		場合にも、	in the case
	ときお			場合における	in the case
	には、	in the		場合において	in the case
	については、	about the		場合は、	in the case
	についても、	also		場合を除いては、	the exception of the case
	につき	it is attached to		は、	
	においては、	in the		ことを妨げない。	this shall not preclude
	に申し出て、	the offer to		の	
	により、	by		しなかった	did not
2	この場合においては、	in this case	3	同様とする。	the same
	については、	about the		及び	and
	及び	and		限る。	limited
	ただし、	however		この限りでない。	this shall not apply
	場合において	in the case		除く。	except

Figure 4.15: List of cue phrases.

Table 4.12: Experimental results with context features

Setting 1: Gold logical parts			
System	Precision(%)	Recall(%)	F_1(%)
Previous model (ILP+SVMs)	78.97	81.22	80.08
+Context features	**80.79**	**83.88**	**82.31(+2.23)**
Setting 2: Full system			
System	Precision(%)	Recall(%)	F_1(%)
Previous model (ILP+SVMs)	62.73	54.57	58.36
+Context features	**63.57**	**55.20**	**59.09(+0.73)**

Heuristic Rules for Post-Processing

Our framework for analyzing logical structures is totally based on statistical machine learning models. By analyzing the output of our system, we saw that in some cases, statistical models produced unreasonable results. However, these results could be corrected by applying some heuristic rules.

In this section, we investigate the problem of using heuristic rules to improve the performance of the system through a post-processing step. The main purpose is to show that using heuristic rules combined with statistical models can get better results than using only statistical models. We illustrate this claim by showing experiments with two simple heuristic rules as bellow. The problem of building such heuristic rules (automatically or manually) is still an open question for further research.

Heuristic rule 1: *If two logical structures have the same requisite part, so they should be combined into one logical structure.*

For example, logical structure $A \Rightarrow T$ and logical structure $A \Rightarrow C$ should be combined into a new logical structure whose requisite part consists of A and effectuation part consists of both T and C.

In this example, A means antecedent part, T means topic part, and C means conse-

Table 4.13: Experimental results with heuristic rules for post-processing

Setting 1: Gold logical parts			
System	Precision(%)	Recall(%)	F_1(%)
Previous model (ILP+SVMs)	78.97	81.22	80.08
+Context features	**80.79**	83.88	82.31(+2.23)
+Post-Processing	80.77	**86.17**	**83.38(+3.30)**

Setting 2: Full system			
System	Precision(%)	Recall(%)	F_1(%)
Previous model (ILP+SVMs)	62.73	54.57	58.36
+Context features	63.57	55.20	59.09(+0.73)
+Post-Processing	**63.68**	**56.98**	**60.14(+1.78)**

quent part.

Heuristic rule 2: *If one logical structure has all logical parts in A type, and another logical structure has logical parts in both A type and another type, then the logical structure with all logical parts in A type should be split into new logical structures, each logical structure contains one logical part in A type.*

For example, suppose that we have a logical structure consisting of A_1 and A_2 and logical structure $A_3 \Rightarrow C$, then the first logical structure should be split into two new logical structures as follows:

1. $A_1 \Rightarrow C$, and
2. $A_2 \Rightarrow C$

In this example, A_1, A_2, and A_3 are antecedent parts, and C means consequent part.

We also conducted experiments with heuristic rules for post-processing in two settings: using gold logical parts and using our recognized logical parts (full system). Experimental results are shown in Table 4.13. We can see that using heuristic rules through a post-processing step improved the performance of the system in both settings. We achieved 83.38% in the F_1 score when using gold logical parts and 60.14% in the F_1 score for the whole recognition task.

4.8 Conclusions

We have introduced the task of analyzing logical structures of paragraphs in legal articles, a new task which has been studied in research on Legal Engineering. There are two main goals of this task: 1) to help people in understanding legal documents; 2) to support other tasks in legal text processing, and therefore make computers be able to process legal texts automatically. We presented the Japanese National Pension Law corpus, an annotated corpus of real legal articles for the task. We also described a two-phase framework to complete the task. In the first phase, we presented a multi-layer sequence learning model for recognizing logical parts that uses CRFs as the learning method. In the second phase, we proposed a graph-based method for recognizing logical structures and exploited ILP formulation to solve the optimization problem on the graph. To learn weights of graphs, we chose MEMs and SVMs as our learning methods.

We conducted 10-fold cross-validation tests on the Japanese National Pension Law corpus. Experimental results showed that our proposed model outperformed a baseline model in both subtasks: *recognition of logical parts* and *recognition of logical structures.* For the first subtask, our model achieved 74.37% in the F_1 score, compared with 63.33% of the baseline model. For the second subtask, when using gold logical parts as the input, our model achieved 83.38% in the F_1 score, compared with 75.89% of the baseline model. When using the output of the first subtask as the input, our model achieved 60.14% in the F_1 score, compared with 51.12% of the baseline model. Our results provide a baseline for further research on this interesting task.

In future work, we will continue to focus on the first subtask: *Recognition of logical parts.* A direction is to investigate methods that can train recognition models in different layers simultaneously. The advantage of such joint learning models is that they can exploit information of logical parts in several layers at the same time.

Another direction is to study how to formulate the problem of *Analyzing Logical Structures of Paragraphs in Legal Articles* as a single task, in which we can recognize logical parts and logical structures simultaneously.

We also investigate how to exploit the results of this task to support other tasks in legal text processing.

Chapter 5

Exploiting Discourse Information to Identify Paraphrases

In this chapter, we present our study on exploiting discourse information to identify paraphrases. We first introduce the paraphrase identification task and our motivation in Section 5.1. We then present related work on paraphrase identification in Section 5.2. Section 5.3 describes the relation between paraphrases and discourse units. Section 5.4 presents our method for computing text similarity based on elementary discourse units, EDU-based similarity. Experiments on the paraphrase identification task are described in Section 5.5. Finally, Section 5.6 concludes the chapter and discusses future work.

5.1 Introduction

Paraphrase identification is the task of determining whether two sentences have essentially the same meaning. This task has been shown to play an important role in many natural language processing applications, including text summarization [9], question answering [39], machine translation [19], natural language generation [47], and plagiarism detection [139]. For example, detecting paraphrase sentences would help a text summarization system to avoid adding redundant information.

Paraphrase identification is not an easy task. Considering the two following sentence pairs, the first sentence pair is a paraphrase although the two sentences only share a few words, while the second one is not a paraphrase even though the two sentences contain almost all the same words.

- "*That would indeed be a great blessing.*" and
  "*The Lord had indeed fulfilled his hopes, and answered his prayers.*"
- "*Peter usually goes to the cinema with his girlfriend.*" and
  "*Peter never goes to the cinema with his girlfriend.*"

Although the paraphrase identification task is defined in the term of semantics, it is usually modeled as a binary classification problem, which can be solved by training a statistical classifier. Many methods have been proposed for identifying paraphrases. These methods usually employ the similarity between two sentences as features, which are computed based on words [42, 69, 87], n-grams [34, 69], syntactic parse trees [34, 111, 122], WordNet [69, 87], and MT metrics, the automated metrics for evaluation of translation quality [80].

Recently, several studies have shown that discourse structures deliver important information for paraphrase computation. For example, to extract paraphrases, Deléger and Zweigenbaum [35] match similar paragraphs in comparable texts. Regneri and Wang [110] extend the distributional hypothesis that entities are similar if they share similar contexts at the discourse level. According to them, sentences that play the same role in a certain discourse and have a similar discourse context can be paraphrases, even if a semantic similarity model does not consider them very similar. Using this assumption, Regneri and Wang [110] introduce a method for collecting paraphrases based on the sequential event order in discourse. However, both Deléger and Zweigenbaum [35] and Regneri and Wang [110] only consider some special kinds of data, where the discourse structures can be easily extracted.

Complete discourse structures such as discourse trees in the RST Discourse Treebank (RST-DT) [21] are difficult to extract though they can be very useful for paraphrase computation [110]. In order to produce such complete discourse structures for a text, we first segment the text into several elementary discourse units (EDUs). Each EDU may be a simple sentence or a clause in a complex sentence. Consecutive EDUs are then put in relation with each other to create a discourse tree [81]. Existing full automatic discourse parsing systems are neither robust nor very precise [8, 61, 110]. In recent years, however, several discourse segmenters with high performance have been introduced [7, 53, 60]. The discourse segmenter[1] described in Bach et al. [7] gives 91.0% in the F_1 score on the RST-DT corpus when using Stanford parse trees [66].

In this chapter, we present a new method to compute the similarity between two sentences based on elementary discourse units (EDU-based similarity). We first segment two sentences into several EDUs using a discourse segmenter, which is trained on the RST-DT corpus. These EDUs are then employed for computing the similarity between two sentences. The key idea is that for each EDU in one sentence, we try to find the most *similar* EDU in the other sentence and compute the similarity between them. We show how our method can be applied to the paraphrase identification task. Experimental results on the PAN corpus [80] show that our method is effective for this task. To our knowledge, this is the first work that employs discourse units for computing similarity as well as for identifying paraphrases.

5.2 Related Work

There have been many studies on the paraphrase identification task. Finch et al. [43] use some MT metrics, including BLEU [104], NIST [38], WER [98], and PER [78] as features for a SVM classifier. Wan et al. [146] combine BLEU features with some others extracted from dependency relations and tree edit-distance, and also take SVMs as the learning method to train a binary classifier. Mihalcea et al. [87] use pointwise mutual information, latent semantic analysis, and WordNet to compute an arbitrary text-to-text similarity metric. Kozareva and Montoyo [69] employ features based on longest common subsequence (LSC), skip n-grams, and WordNet. They use a meta-classifier composed of SVMs, k-nearest neighbor, and maximum entropy models. Rus et al. [111] adapt a graph-

[1]Note that this is our own segmenter, which has been presented in Chapter 3.

based approach (originally developed for recognizing textual entailment) for paraphrase identification. Fernando and Stevenson [42] build a matrix of word similarities between all pairs of words in both sentences. Das and Smith [34] introduce a probabilistic model which incorporates both syntax and lexical semantics using quasi-synchronous dependency grammars for identifying paraphrases. Socher et al. [122] describe a joint model that uses the features extracted from both single words and phrases in the parse trees of the two sentences.

Most recently, Madnani et al. [80] present an investigation of the impact of MT metrics on the paraphrase identification task. They examine 8 different MT metrics, including BLEU [104], NIST [38], TER [120], TERP [121], METEOR [37], SEPIA [50], BADGER [105], and MAXSIM [24], and show that a system using nothing but some MT metrics can achieve state-of-the-art results on this task.

Compared with previous work, our work makes some contributions to the advancement of paraphrase identification as follows:

- We present the first work on relations between discourse units and paraphrasing, in which discourse units play an important role in paraphrasing. We also show that many paraphrase sentences can be generated from the original sentences by conducting several transformations on discourse units.

- We present EDU-based similarity, a new method for computing the similarity between two sentences based on elementary discourse units. As shown in the experimental section, our method is effective for identifying paraphrases.

Discourse structures have only marginally been considered for paraphrase computation. Regneri and Wang [110] introduce a method for collecting paraphrases using discourse information on a special type of data, TV show episodes. With such kind of data, they assume that discourse structures can be achieved by taking sentence sequences of recaps. Our work employs the recent advances in discourse segmentation. Hernault et al. [53] introduce a sequence model for segmenting texts into discourse units using Conditional Random Fields. Joty et al. [60] present a discourse segmentation system exploiting a Logistic Regression model with l_2 regularization. They learn the model parameters using the L-BFGS algorithm [152]. Bach et al. [7] introduce a reranking model for discourse segmentation using subtree features. These segmenters achieve relatively high results on the RST-DT corpus (89.0% in Hernault et al. [53], 90.1% in Joty et al. [60], and 91.0% in Bach et al. [7]).

Unlike previous studies that exploit discourse structure of specific data for paraphrase computation, our work introduce a general method that is not limited to any specific kind of text. To our knowledge, this is the first work that exploits discourse units for computing text similarity as well as for identifying paraphrases.

5.3 Paraphrases and Discourse Units

In this section, we describe the relation between paraphrases and discourse units. We will show that discourse units are blocks which play an important role in paraphrasing.

Figure 5.1 shows an example of a paraphrase sentence pair. In this example, the first sentence can be divided into three elementary discourse units (EDUs), 1A, 1B, and

[Or his needful holiday has come,]1A [and he is staying at a friend's house,]1B [or is thrown into new intercourse at some health-resort.]1C

[Or need a holiday has come,]2A [and he stayed in the house of a friend]2B [or disposed of in a new relationship to a health resort.]2C

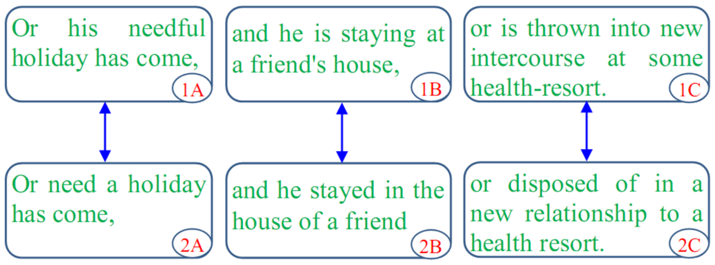

Or his needful holiday has come, 1A
and he is staying at a friend's house, 1B
or is thrown into new intercourse at some health-resort. 1C
Or need a holiday has come, 2A
and he stayed in the house of a friend 2B
or disposed of in a new relationship to a health resort. 2C

Figure 5.1: A paraphrase sentence pair in the PAN corpus [80].

[Age of consent legislation,]3A [as applied to the question of social vice,]3B [is one thing,]3C [and consent as applied to the question of slavery , quite another thing.]3D

[When applied to social vices,]4A [age of consent legislation is one thing,]4B [when the legislation is applied to slavery,]4C [a totally different and epidemic problem exists.]4D

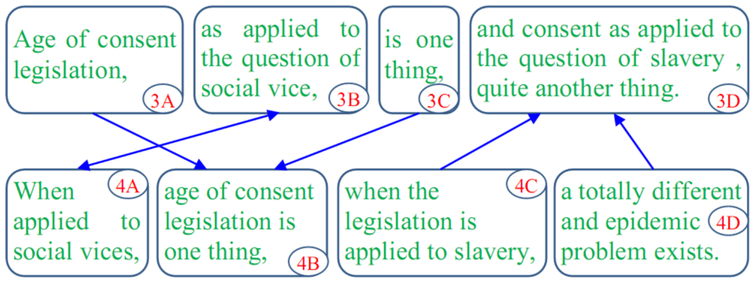

Age of consent legislation, 3A
as applied to the question of social vice, 3B
is one thing, 3C
and consent as applied to the question of slavery , quite another thing. 3D
When applied to social vices, 4A
age of consent legislation is one thing, 4B
when the legislation is applied to slavery, 4C
a totally different and epidemic problem exists. 4D

Figure 5.2: Another paraphrase sentence pair in the PAN corpus.

1C, and the second sentence can also be segmented into three EDUs, 2A, 2B, and 2C. Comparing these six EDUs, we can see that they make three aligned pairs of paraphrases: 1A with 2A, 1B with 2B, and 1C with 2C. Therefore, if we consider the first sentence is the original sentence, the second sentence can be created by paraphrasing each discourse unit in the original sentence.

Figure 5.2 shows a more complex case. The first sentence consists of four EDUs, 3A, 3B, 3C, and 3D; and the second sentence includes four EDUs, 4A, 4B, 4C, and 4D. In this case, if we consider the first sentence is the original one, we have some remarks:

- The discourse unit 4A is a paraphrase of the discourse unit 3B,
- The unit 4B is a paraphrase of the combination of two units, 3A and 3C, and
- The combination of two units 4C and 4D is a paraphrase of the unit 3D.

By analyzing paraphrase sentences, we found that discourse units are very important to paraphrasing. In many cases, a paraphrase sentence can be created by applying the following operations to the original sentence:

1. Reordering two discourse units,
2. Combining two discourse units into one unit,
3. Dividing one discourse unit into two units, and
4. Paraphrasing a discourse unit.

An example of Operation 1 and Operation 2 is the case of units 3A, 3B, and 3C in Figure 5.2 (reordering 3A and 3B, and then combining 3A and 3C). Unit 3D illustrates an example for Operation 3. The last operation is the most important operation, which is applied to almost all of discourse units.

5.4 EDU-Based Similarity

Motivated by the analysis of the relation between paraphrases and discourse units, we propose a method to compute the similarity between two sentences. In this section, we assume that each sentence can be represented as a sequence of elementary discourse units (EDUs). The method of segmenting sentences into EDUs was presented in Chapter 3.

First, we present the notion of *ordered similarity functions.* Given two arbitrary texts t_1 and t_2, an ordered similarity function $Sim_{ordered}(t_1, t_2)$ will return a score in real number, which measures how t_1 is similar to t_2. Note that in this function, the roles of t_1 and t_2 are different, in which t_2 can be seen as a *gold* standard and we want to evaluate t_1 based on t_2. Examples of ordered similarity functions are MT metrics, which evaluate how a hypothesis text (t_1) is similar to a reference text (t_2).

Given an ordered similarity function $Sim_{ordered}$, we can define the similarity between two arbitrary texts t_1 and t_2 as follows:

$$Sim(t_1, t_2) = \frac{Sim_{ordered}(t_1, t_2) + Sim_{ordered}(t_2, t_1)}{2}. \tag{5.1}$$

Let (s_1, s_2) be a sentence pair, then s_1 and s_2 can be represented as sequences of elementary discourse units: $s_1 = (e_1, e_2, \ldots, e_m)$ and $s_2 = (f_1, f_2, \ldots, f_n)$, where m and n are the numbers of discourse units in s_1 and s_2, respectively. We define an ordered similarity function between s_1 and s_2 as follows:

$$Sim_{ordered}(s_1, s_2) = \sum_{i=1}^{m} Imp(e_i, s_1) * Sim_{ordered}(e_i, s_2) \tag{5.2}$$

where $Imp(e_i, s_1)$ is the importance of the discourse unit e_i in the sentence s_1, and $Sim_{ordered}(e_i, s_2)$ is the ordered similarity between the discourse unit e_i and the sentence s_2.

In this work, we simply assume that all words contribute equally to the meaning of the sentence. Therefore, the importance function can be computed as follows:

$$Imp(e_i, s_1) = \frac{|e_i|}{|s_1|} \tag{5.3}$$

where $|e_i|$ and $|s_1|$ are the lengths (in words) of the discourse unit e_i and the sentence s_1, respectively.

The ordered similarity $Sim_{ordered}(e_i, s_2)$ is computed based on the discourse unit f_j in the sentence s_2, which is the most similar to e_i:

$$Sim_{ordered}(e_i, s_2) = Max_{j=1}^{n} Sim_{ordered}(e_i, f_j). \tag{5.4}$$

Substituting (5.3) and (5.4) into (5.2) we have:

$$Sim_{ordered}(s_1, s_2) = \sum_{i=1}^{m} \frac{|e_i|}{|s_1|} Max_{j=1}^{n} Sim_{ordered}(e_i, f_j). \tag{5.5}$$

Finally, from (5.5) and (5.1) we have the formula for computing the similarity between two sentences based on their discourse units (EDU-based similarity), where the ordered similarity between two units is computed directly using the definition of the ordered similarity function, as follows:

$$\begin{aligned} Sim(s_1, s_2) &= \frac{Sim_{ordered}(s_1, s_2) + Sim_{ordered}(s_2, s_1)}{2} \\ &= \frac{1}{2} * \sum_{i=1}^{m} \frac{|e_i|}{|s_1|} * Max_{j=1}^{n} Sim_{ordered}(e_i, f_j) \\ &+ \frac{1}{2} * \sum_{j=1}^{n} \frac{|f_j|}{|s_2|} * Max_{i=1}^{m} Sim_{ordered}(f_j, e_i). \end{aligned} \tag{5.6}$$

We now present an example of computing the EDU-based similarity between two sentences in Figure 5.1 using the BLEU score. Table 5.1 shows the basic information of the calculation step by step. Line 1 and line 2 present two tokenized sentences and their lengths in words. Lines 3 through 5 compute the similarity between two sentences directly based on sentences. By using this method, the similarity is 0.5332. Elementary discourse units of two sentences are shown in lines 6 through 11. The computation of EDU-based similarity is described in lines 12 through 20. By using this method, the similarity is 0.5369, which is slightly higher than the similarity computed directly using sentences.

5.5 Experiments

This section describes our experiments on the paraphrase identification task using EDU-based similarities as features for statistical classifiers. Like the work of Madnani et al. [80], we employed MT metrics as the ordered similarity functions. However, we computed the MT metrics based on EDUs in addition to MT metrics based on sentences. In all experiments, parse trees were obtained by using the Stanford parser [66].

5.5.1 Data & Evaluation Method

We conducted experiments on the PAN corpus[2], a corpus for paraphrase identification task created from a plagiarism detection corpus [80]. Table 5.2 shows the statistics on the corpus. The corpus includes a training set of 10,000 sentence pairs and a test set of

[2]The corpus can be downloaded at the address: http://bit.ly/mt-para.

Table 5.1: An example of computing sentence-based and EDU-based similarities

Line	Computation		
1	s_1: Or his needful holiday has come , and he is staying at a friend 's house , or is thrown into new intercourse at some health-resort .		Length=27
2	s_2: Or need a holiday has come , and he stayed in the house of a friend , or disposed of in a new relationship to a health resort .		Length=29
	Sentence-based Similarity		
3	BLEU(s_1, s_2) = **0.5333**		
4	BLEU(s_2, s_1) = **0.5330**		
5	$Sim(s_1, s_2) = \frac{BLEU(s_1,s_2)+BLEU(s_2,s_1)}{2} = \mathbf{0.5332}$		
	Discourse Units		
6	e_1: Or his needful holiday has come ,		Length=7
7	e_2: and he is staying at a friend 's house ,		Length=10
8	e_3: or is thrown into new intercourse at some health-resort .		Length=10
9	f_1: Or need a holiday has come ,		Length=7
10	f_2: and he stayed in the house of a friend ,		Length= 10
11	f_3: or disposed of in a new relationship to a health resort .		Length=12
	EDU-based Similarity		
12	BLEU(e_1, f_1) = **0.7143**	BLEU$(e_1, f_2) = 0.0931$	BLEU$(e_1, f_3) = 0.0699$
13	BLEU$(e_2, f_1) = 0.1818$	BLEU(e_2, f_2) = **0.5455**	BLEU$(e_2, f_3) = 0.0830$
14	BLEU$(e_3, f_1) = 0.0833$	BLEU$(e_3, f_2) = 0$	BLEU(e_3, f_3) = **0.4167**
15	EDU_BLEU$(s_1, s_2) = \frac{7}{27} * 0.7143 + \frac{10}{27} * 0.5455 + \frac{10}{27} * 0.4167 = \mathbf{0.5416}$		
16	BLEU(f_1, e_1) = **0.7143**	BLEU$(f_1, e_2) = 0.1613$	BLEU$(f_1, e_3) = 0.0699$
17	BLEU$(f_2, e_1) = 0.1000$	BLEU(f_2, e_2) = **0.5429**	BLEU$(f_2, e_3) = 0$
18	BLEU$(f_3, e_1) = 0.0833$	BLEU$(f_3, e_2) = 0.0833$	BLEU(f_3, e_3) = **0.4167**
19	EDU_BLEU$(s_2, s_1) = \frac{7}{29} * 0.7143 + \frac{10}{29} * 0.5429 + \frac{12}{29} * 0.4167 = \mathbf{0.5321}$		
20	EDU_Sim$(s_1, s_2) = \frac{EDU_BLEU(s_1,s_2)+EDU_BLEU(s_2,s_1)}{2} = \mathbf{0.5369}$		

Table 5.2: PAN corpus for paraphrase identification

	Training Set	Test Set
Number of sentence pairs	10,000	3,000
Number of EDUs per sentence	4.31	4.33
Number of words per sentence	40.07	41.12

$3,000$ sentence pairs[3]. On average, each sentence contains about 4.3 discourse units, and about 40.1 words in the training set, 41.1 words in the test set. We chose this corpus for the following reasons. First, it is a large corpus for detecting paraphrases. Second, it contains many long sentences. Our method computes similarities based on discourse units. It is suitable for long sentences with several EDUs. Lastly, according to Madnani et al. [80], the PAN corpus contains many realistic examples of paraphrases.

We evaluated the performance of our paraphrase identification system, which exploited the EDU-based similarities as features, by accuracy and the F_1 score. The accuracy was the percentage of correct predictions over all the test set, while the F_1 score was computed only based on the paraphrase sentence pairs[4].

5.5.2 MT Metrics

We investigated our method with six different MT metrics (six types of ordered similarity functions). These metrics have been shown to be effective for the task of paraphrase identification [80].

1. BLEU [104] is the most commonly used MT metric. It computes the amount of n-gram overlap between a hypothesis text (the output of a translation system) and a reference text.

2. NIST [38] is a variant of BLEU using the arithmetic mean of n-gram overlaps. Both BLEU and NIST use exact matching. They have no concept of synonymy or paraphrasing.

3. TER [120] computes the number of edits needed to "fix" the hypothesis text so that it matches the reference text.

4. TERP [121] or TER-Plus is an extension of TER, that utilizes phrasal substitutions, stemming, synonyms, and other improvements.

5. METEOR [37] is based on the harmonic mean of unigram precision and recall. It also incorporates stemming, synonymy, and paraphrase.

6. BADGER [105], a language independent metric, computes a compression distance between two sentences using the Burrows Wheeler Transformation (BWT).

Among the six MT metrics, TER and TERP compute a translation error rate between a hypothesis text and a reference text. Therefore, the smaller these MT metrics are, the

[3]The training set and the test set were divided exactly the same as in the work of Madnani et al. [80].

[4]If we consider each sentence pair as an instance with label +1 for *paraphrase* and label -1 for *non-paraphrase*, the reported F_1 score was the F_1 score on label +1.

Table 5.3: Experimental results on each individual MT metric

	Sentence-based similarities		+ EDU-based similarities	
MT Metric	Accuracy(%)	F$_1$(%)	Accuracy(%)	F$_1$(%)
BLEU(1-4)	89.0	88.4	89.6(+0.6)	89.1(+0.7)
NIST(1-5)	84.6	82.7	87.6(+3.0)	86.8(+4.1)
TER	88.2	87.3	88.5(+0.3)	87.7(+0.4)
TERP	91.0	90.6	91.1(+0.1)	90.8(+0.2)
METEOR	90.0	89.6	89.8(-0.2)	89.4(-0.2)
BADGER	88.1	87.8	88.2(+0.1)	87.8(-)

more similar the two texts are. When using these metrics in computing EDU-based similarities, we replaced the *max* function in Equation (5.6) by a *min* function.

$$\begin{aligned} Sim(s_1, s_2) &= \frac{Sim_{ordered}(s_1, s_2) + Sim_{ordered}(s_2, s_1)}{2} \\ &= \frac{1}{2} * \sum_{i=1}^{m} \frac{|e_i|}{|s_1|} * Min_{j=1}^{n} Sim_{ordered}(e_i, f_j) \\ &+ \frac{1}{2} * \sum_{j=1}^{n} \frac{|f_j|}{|s_2|} * Min_{i=1}^{m} Sim_{ordered}(f_j, e_i). \end{aligned}$$

5.5.3 Experimental Results with a Single SVM Classifier

In all experiments presented in this section, we chose SVMs [140] as the learning method to train a single binary classifier[5]. SVMs have been demonstrated their performance on a number of problems in areas, including computer vision, handwriting recognition, pattern recognition, and statistical natural language processing. In the field of natural language processing, SVMs have been applied to many tasks, including machine translation [150], topic classification [147], information extraction [13], sentiment analysis [107, 114], discourse parsing [54], and achieved very good results. In fact, SVMs have been also exploited successfully to identify paraphrases [43, 80, 87, 146]

First, we investigated each individual MT metric. To see the contributions of EDU-based similarities, we conducted experiments in two settings. In the first setting, we directly applied the MT metric to pairs of sentences to get the similarities (sentence-based similarities). In the second setting, we computed EDU-based similarities in addition to the sentence-based similarities. Like Madnani et al. [80], in our experiments, we used BLEU1 through BLEU4 as 4 different features and NIST1 through NIST5 as 5 different features[6]. Table 5.3 shows experimental results in two settings on the PAN corpus. We can see that, adding EDU-based similarities improved the performance of the paraphrase identification system with most of the MT metrics, especially with NIST(3.0%), BLEU (0.6%), and TER (0.3%).

Table 5.4 shows experimental results with multiple MT metrics on the PAN corpus. With each MT metric, we computed the similarities in both methods, based directly on

[5]We conducted experiments on LIBSVM tool [26] with the RBF kernel.

[6]BLEUn and NISTn use n-grams.

Table 5.4: Experimental results on combined MT metrics

MT Metrics	Accuracy(%)	F_1(%)
BLEU	89.6	89.1
BLEU+NIST	91.2	90.9
BLEU+NIST+TER	91.8	91.6
BLEU+NIST+TER+TERP	**93.1**	**93.0**
Madnani-4	91.5	91.2
Madnani-6	92.3	92.1

Table 5.5: Experimental results on long and short sentences

Subset	#sent pairs	#EDUs/sent	#words/sent	Acc.(%)	F_1(%)
Subset1	1317	6.5	56.6	**96.6**	**94.8**
Subset2	1683	2.6	27.2	90.4	92.3

sentences and based on discourse units. We gradually added MT metrics one by one to the system. After adding the TERP metric, we achieved 93.1% accuracy and 93.0% in the F_1 score. Adding two more MT metrics, METEOR and BADGER, the performance was not improved.

Two last rows of Table 5.4 shows the results of Madnani et al. [80] when using 4 MT metrics, including BLEU, NIST, TER, and TERP (Madnani-4) and when using all 6 MT metrics (Madnani-6)[7]. Compared with the best previous results, our method improves 0.8% accuracy and 0.9% in the F_1 score. It yields a 10.4% error rate reduction. Note that we used the same training and test datasets as the datasets in previous work [80].

We also investigated our method on long and short sentences. We divided sentence pairs in the test set into two subsets: Subset1 (long sentences) contains sentence pairs that both sentences have at least 4 discourse units[8], and Subset2 (short sentences) contains the other sentence pairs. Table 5.5 shows the information and experimental results on two subsets. Subset1 consists of 1,317 sentence pairs (on average, 6.5 EDUs and 56.6 words per sentence), while Subset2 consists of 1,683 sentence pairs (on average, 2.6 EDUs and 27.2 words per sentence). We can see that, our method was effective for the long sentences, which we achieved 96.6% accuracy and 94.8% in the F_1 score compared with 90.4% accuracy and 92.3% in the F_1 score of the short sentences.

5.5.4 Revision Learning & Voting

We presented experiments that combine several MT metrics into a single SVM classifier. In this section, we investigate the combination of several SVM classifiers (ensemble models) building on individual MT metrics, for the paraphrase identification task. We present experiments with a revision learning model and a maximal voting model. Revision learning and voting are popular and simple, but also powerful methods to produce ensemble models. They have been shown to be effective in a number of NLP problems, including word sense disambiguation [44, 55], part-of-speech tagging [94], word alignment [148],

[7]Madnani et al. [80] show that adding more MT metrics does not improve the performance of the paraphrase identification system.

[8]Number 4 was chosen because on average each sentence contains about 4 EDUs (see Table 5.2).

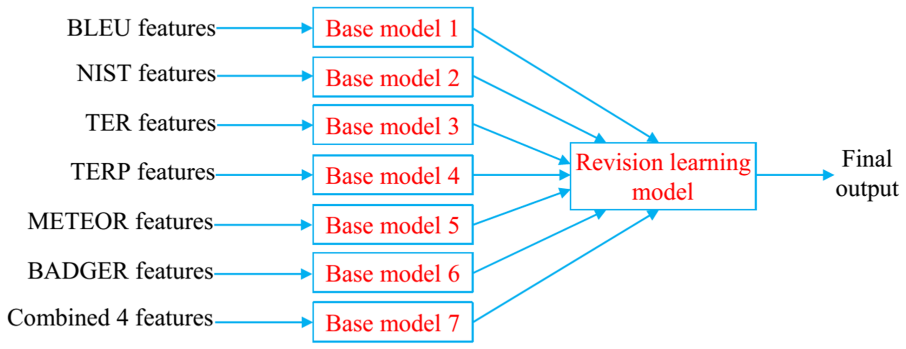


Figure 5.3: A revision learning model for the paraphrase identification task.

dependency parsing [3, 127], named entity translation [133], and information extraction [136].

The main idea of two models can be expressed as follows.

1. We first build several classifiers (base models) to identify paraphrases using normal features (MT metrics).

2. We then build a final classifier (revision model or voting model) to judge paraphrase relation based on the outputs of the base models in the first step.

In our experiments, we built seven base models using SVMs. The first six models employed six MT metrics (BLEU, NIST, TER, TERP, METOER, and BADGER) as features, respectively. The last model used the best combination of MT metrics, including BLEU, NIST, TER, and TERP. For each MT metric, we computed two types of similarities, sentence-based similarity and EDU-based similarity.

Our revision learning model is illustrated in Figure 5.3. The revision learning model was also trained by using SVMs with features as the probabilities that base models judge the sentence pair is a paraphrase or not. Each base model contributes two features (probability of paraphrase and probability of non-paraphrase) that yield totally fourteen features. To create training data for the revision model, we used a development set, which is about 20% of the training set.

Algorithm 10 describes our voting model. The voting model first picks the model that produces the highest probability among seven base models. If that probability is higher than a threshold[9] (confident score), the output of that model is selected as the output of the voting model, otherwise the voting model selects the output of the best base model (the seventh base model with combined MT metrics) as output. The intuitive meaning is that if none of the base models gives a confident result, the best base model is a reasonable choice.

Table 5.6 shows experimental results of the revision learning model and the voting model on the PAN corpus. Our revision learning model achieved 93.2% accuracy and 93.1% in the F_1 score, which slightly improved the best base model with combined MT metrics. The voting model achieved the best results with 93.4% accuracy and 93.3% in the F_1 score, which improved 0.3% (both accuracy and in the F_1 score) compared with

[9]The threshold was set by using a development set, which is about 20% of the training set. It was 0.95 in our experiments.

Algorithm 10 A voting algorithm for the paraphrase identification task.

```
1: **Input**:
     • A sentence pair
     • Seven base models
     • A threshold $T$
2: **Output**: Yes (in the case of paraphrase), No (in the case of non-paraphrase)
3: Predict label for the sentence pair using the base models
4: Select the base model (called $BM$) producing the highest probability ($prob$)
5: if $prob \geq T$ then
6:     Return the output of $BM$
7: else
8:     Return the output of the best base model (using combined MT metrics)
9: end if
```

Table 5.6: Experimental results of the revision learning model and the voting model

Model	Accuracy(%)	F_1(%)
Madnani et al. [80]	92.3	92.1
The best base model (combined MT metric)	93.1(+0.8)	93.0(+0.9)
Revision learning	93.2(+0.9)	93.1(+1.0)
Voting	**93.4(+1.1)**	**93.3(+1.2)**

the best base model, and 1.1% accuracy and 1.2% in the F_1 score compared with the previous work of Madnani et al. [80].

5.5.5 Error Analysis

This section identifies the cause of the errors that our method made on the test data of the PAN corpus, which includes 3,000 sentence pairs. Firstly, we wanted to know the statistic information of the experimental results on the test data. We considered the following questions:

1. With each sentence pair in the test set, how many models among seven base models produced a correct output?
2. How many sentence pairs were predicted correctly by at least one base model? And therefore, how many sentence pairs were unable to be predicted correctly by base models?

Table 5.7 shows statistic information of the experimental results on the test set. Among 3,000 sentence pairs, 2,344 sentence pairs (78.1%) were predicted correctly by all seven base models, 226 sentence pairs (7.5%) were predicted correctly by six base models, and only 89 sentence pairs (3.0%) were unable to be predicted correctly by base models. There are 2,911 sentence pairs (97%) that that were predicted correctly by at least one base model. The upper bound of our method can therefore be considered as 97%.

Table 5.7: Statistic information of the experimental results on the test set

#Base model(s) predicted correctly	#Sentence pairs	Percentage
7	2344	78.1
6	226	7.5
5	89	3.0
4	85	2.8
3	42	1.4
2	66	2.2
1	59	2.0
0	**89**	**3.0**

Paraphrases (predicted as non-paraphrases)

This section shows three main types of errors in which paraphrase sentence pairs were predicted as non-paraphrases (or false negative):

1. ***Complex Sentential Paraphrases***
These sentence pairs are real world paraphrases, where the paraphrase sentences are produced by making several complex transformations and using a lot of new words. Considering two following sentence pairs, in the first case the paraphrase sentence is even totally rewritten.

    - *““Sukey will be good to him,” said Mrs. Lawton, in tones more gentle than usual.”* and
    “Was it her imagination, or did Mrs. Lawton’s eyes look shifty?”
    - *“A rich man named Fintan was childless, for his wife was barren for many years.”* and
    “Wealthy fellow, Fintan, had an infertile wife, so their marriage was a childless one.”

2. ***Idioms***
These sentence pairs use idioms that make the meaning very difficult to understand and therefore difficult to judge the paraphrase relation. Below is such an example.

    - *“Such an artist, by the very nature of his endeavors, must needs stand above all public-clapper -clawing, pro or con.”* and
    “A true artist must never try to please patrons, clients, or colleagues but must work on his own inspiration and stand apart from the public’s praise or contempt.”

3. ***Typographical and Spelling Errors***
The PAN corpus includes sentence pairs containing typos and spelling errors that make the system cannot judge correctly. Below is such an example.

    - *“If I could only git him to move I’d be happier jest ter foller him.”* and
    “But still, I would follow him if he ever chose to move on.”

Non-paraphrases (predicted as paraphrases)

This section shows two main types of errors in which non-paraphrase sentence pairs were predicted as paraphrases (or false positive):

1. ***Misleading Lexical Overlap***
   These sentence pairs consist of two sentences which have large lexical overlap. They share a lot of words and contain only a few different words. However, these few different words make the meaning change. Here are some examples.

   - *"For catching doves, and other current game, they had ingenious little traps."* and
     "For catching doves, and other small game, they had ingenious romantic journeyings."
   - *"Drawn by Boudier, from a photograph by M. de Morgan."* and
     "Drawn by Boudier, from a photograph by M. Binder."

2. ***Containing***
   These sentence pairs consist of two sentences in which one of them contains the other one but has additional parts. Here is such an example.

   - *"His dinner had been put back half an hour!"* and
     "The end of all things was at hand; his dinner had been put back half an hour!"

5.6 Conclusions

In this chapter, we have showed that discourse information, specifically elementary discourse units, can be exploited to identify paraphrases. We proposed a new method to compute the similarity between two sentences based on elementary discourse units, EDU-based similarity. This method was motivated by the analysis of the relation between paraphrases and discourse units. By analyzing examples of paraphrases, we found that discourse units play an important role in paraphrasing. We applied EDU-based similarity to the task of paraphrase identification. Experimental results on the PAN corpus showed the effectiveness of the proposed method. To the best of our knowledge, this is the first work to employ discourse units for computing similarity as well as for identifying paraphrases. Although our method is proposed for computing the similarity between two sentences, it can be also used to compute the similarity between two arbitrary texts.

Our work will be beneficial to building many natural language processing applications, including paraphrase generation, text summarization, question answering, and machine translation. A general method for paraphrase generation is using machine translation systems to generate a set of paraphrase sentence pairs. After that, a paraphrase identification system is exploited to select the best paraphrases. Paraphrase generation can be used to rewrite the question in a question answering system, or to generate reference sentences for evaluating translation quality. Paraphrase identification can also help a text summarization system avoid adding redundant information to a summary. In a multi-document summarization system, similar sentences in different documents are grouped into a cluster. Important sentences in clusters will be extracted to form a summary. Each cluster may contain several sentences which convey the same information. A paraphrase identification system can be exploited to remove such redundant sentences. Because our method exploits RST discourse structures, it is not limited to any specific kind of text. It is also language-independent and requires only MT metric tools.

There are several ways to extend the current work. First, we would like to investigate our method on other datasets for the paraphrase identification task as well as to other related tasks such as recognizing textual entailment [11] and semantic textual similarity [2]. Second, we plan to exploit the roles of discourse units to improve the method of computing text similarity. In this work, the importance of discourse units is calculated based on only their length (i.e., number of words). However, the importance of a discourse unit also depends on its role in the text and the relations with other units. Exploiting the relations between discourse units for computing similarity may be an interesting direction for further research.

Chapter 6

Conclusions and Future Work

6.1 Summary of the Thesis

In this thesis, we have presented a general framework for the structural analysis of texts. Our framework consists of two steps, i.e., recognizing discourse units of texts and building structures of texts from the discourse units. Under this framework, we proposed a model for learning RST discourse structures of general texts and a model for analyzing logical structures of legal paragraphs. We also described our study on applications of text structures to paraphrase identification. We proposed a new method for computing text similarity based on elementary discourse units and conducted experiments to show that our method is effective to identify paraphrases. We have made the following contributions in the previous chapters:

1. We proposed a new system for unlabeled discourse parsing in the RST framework (Chapter 3), which consists of a model for segmenting texts into elementary discourse units using the reranking method and a model for building discourse trees using dual decomposition. Unlike previous discourse segmenters, which only focus on the boundary of discourse units, our segmenter can capture the whole discourse units by exploiting subtree features. For the discourse tree building step, our model exploits dual decomposition to combine the advantages of two base models which use a heuristic algorithm and an incremental algorithm.

2. We then proposed a two-phase framework for analyzing logical structures of legal paragraphs (Chapter 4), which consists of a multi-layer sequence model for recognizing logical parts and a graph-based model for recognizing logical structures. We introduced an integer linear programming formulation for recognizing logical structures on graphs. We also presented the Japanese National Pension Law corpus, an annotated corpus for learning logical structures of legal paragraphs.

3. We finally proposed a new method for computing text similarity, EDU-based similarity (Chapter 5), which exploits elementary discourse units to compute text similarity. Our method is language-independent and not limited to any kind of text. We also conducted experiments to show that discourse information is an important source for identifying paraphrases.

6.2 Future Directions

In future work, we plan to pursue the following directions:

1. **Studying applications of RST discourse structures to natural language processing applications**.
   In Chapter 3, we presented a discourse parsing system in the RST framework. In future work, we intend to study applications of RST discourse structures to natural language processing applications, such as machine translation, text summarization, and question answering.

2. **Studying joint models for analyzing logical structures of legal texts**.
   In Chapter 4, we presented a two-phase framework for analyzing logical structures of legal paragraphs, in which we recognized logical parts in the first phase and recognized logical structures in the second phase. In this pipeline framework, the output of the first phase will be used as the input of the second phase, which leads to the problem of error propagation. Joint models for recognizing logical parts and logical structures simultaneously may be a solution for such problem.

3. **Exploiting discourse relations to compute text similarity**.
   EDU-based similarity, the method for computing text similarity presented in Chapter 5 calculates the importance of discourse units based on only their length (i.e., the number of words). However, the importance of a discourse unit also depends on its role in the text and the relations with other units. Exploiting the relations between discourse units for computing text similarity may be an interesting direction for further research.

4. **Studying legal text paraphrasing using logical structures**.
   Legal text paraphrasing is an important task whose aim is to produce a new representation for legal documents that could be easier to understand. We intend to study machine learning models for paraphrasing legal texts based on their logical structures.

5. **Studying less supervised models with natural annotations for analyzing structures of different kinds of texts**.
   We plan to continue and extend the current research on analyzing the structures of texts and exploiting the structures of texts to solve other natural language processing tasks.

   Our future research, however, will aim to deal with various kinds of texts and try to loosen the dependence on annotated data. Nowadays, with the appearance of the Internet and personal computers, the Web becomes one of the most important vehicles to convey information. There are many new forms of information on the Web, including web sites, Internet forums, blogs, wikis, and social networks. By analyzing different kinds of information on the Web, we see that:

   - The Web contains huge and various types of data in a lot of domains.
   - Each kind of information on the Web uses a different type of text which has its own structures.

- Analyzing structures of Web texts will be beneficial to deal with many natural language processing applications.

However, most Web texts are plain texts, which can be seen as unannotated data in natural language processing. In our future research, therefore, we will focus on less supervised methods, which try to exploit plain texts, unannotated data, to build learning models.

Appendix A

LSDemo: A Demonstration System for Analyzing Logical Structures of Paragraphs in Legal Articles

This appendix describes LSDemo, a demonstration system for analyzing logical structures of paragraphs in legal articles. The system was built based on the framework presented in Chapter 4.

LSDemo was trained on Japanese National Pension Law (JNPL) corpus. The corpus consists of 83 legal articles, which contain 119 paragraphs with 426 sentences. In the first step, recognizing logical parts, we also exploited the training data for the RRE task [6], which consists of 764 annotated sentences.

LSDemo consists of two main modules: the processing module and the GUI module. The processing module, the core of the system, was implemented using C++. The GUI module, the graphical interface for users, was implemented using C#. The system runs on the Microsoft Windows operating system with Microsoft Visual Studio development toolkit.

A.1 Requirements

LSDemo needs the following software and tools to run correctly:

1. Microsoft Windows operating system (Windows XP or Windows 7)
2. Microsoft Visual Studio development toolkit (Version 2010 or higher)
3. Cabocha, a Japanese dependency structure analyzer [72]
   Available at `http://code.google.com/p/cabocha/`
4. CRF++, an implementation of Conditional Random Fields [70]
   Available at `http://crfpp.sourceforge.net/`
5. LIBSVM, a library for Support Vector Machines [26]
   Available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
6. lp_solve, a mixed Integer Linear Programming solver
   Available at `http://lpsolve.sourceforge.net/`

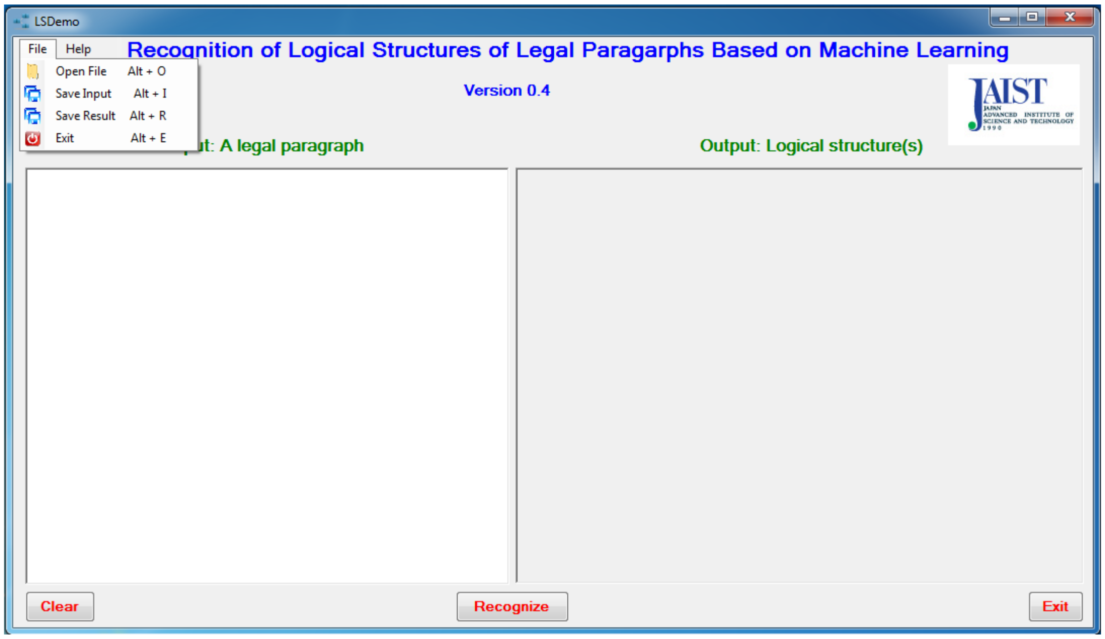


Figure A.1: Graphical user interface of LSDemo.

A.2 Graphical User Interface

Figure A.1 shows graphical user interface (GUI) of LSDemo. The GUI consists of two main text boxes. The left text box serves as the input space, where the input paragraph will be inputted. The right text box is the output space, which displays results.

There are two ways to input a legal paragraph:

1. Type directly into the left text box.

2. Open a file through the ***Open File*** item on the ***File*** menu. The file should contain a legal paragraph with one sentence on each line. Note that sentences should be in Japanese Shift-JIS code.

The main menu contains the following functions:

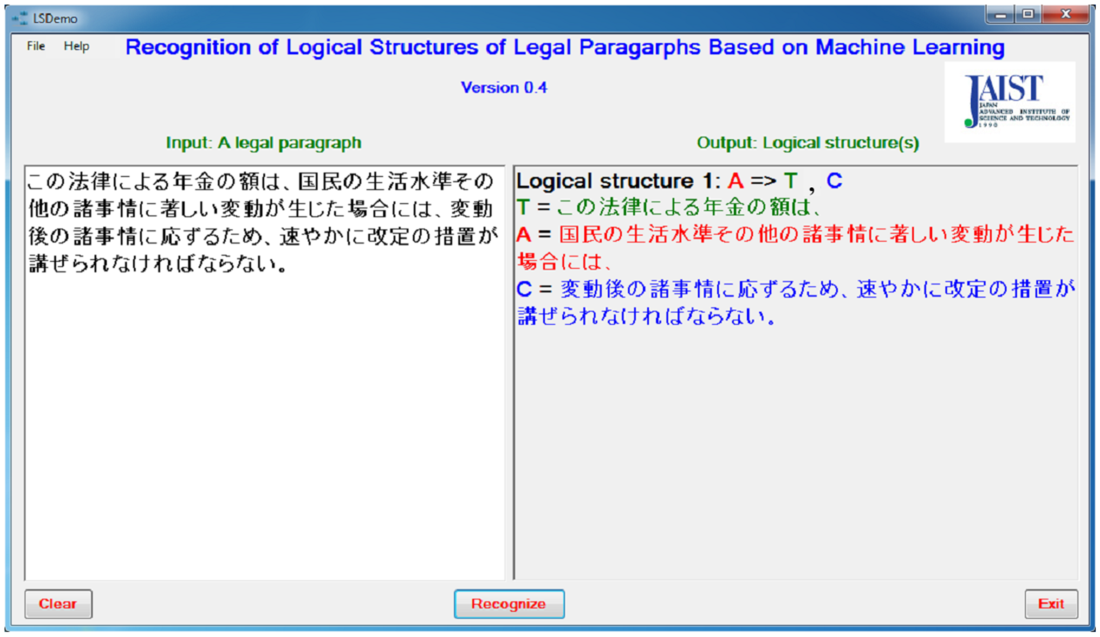
1. To do recognizing logical structures, click on the ***Recognize*** button.

2. To save the input paragraph, click on the ***Save Input*** item on the ***File*** menu.

3. To save the results, click on the ***Save Result*** item on the ***File*** menu.

4. To see the information about authors, click on the ***About*** item on the ***Help*** menu.

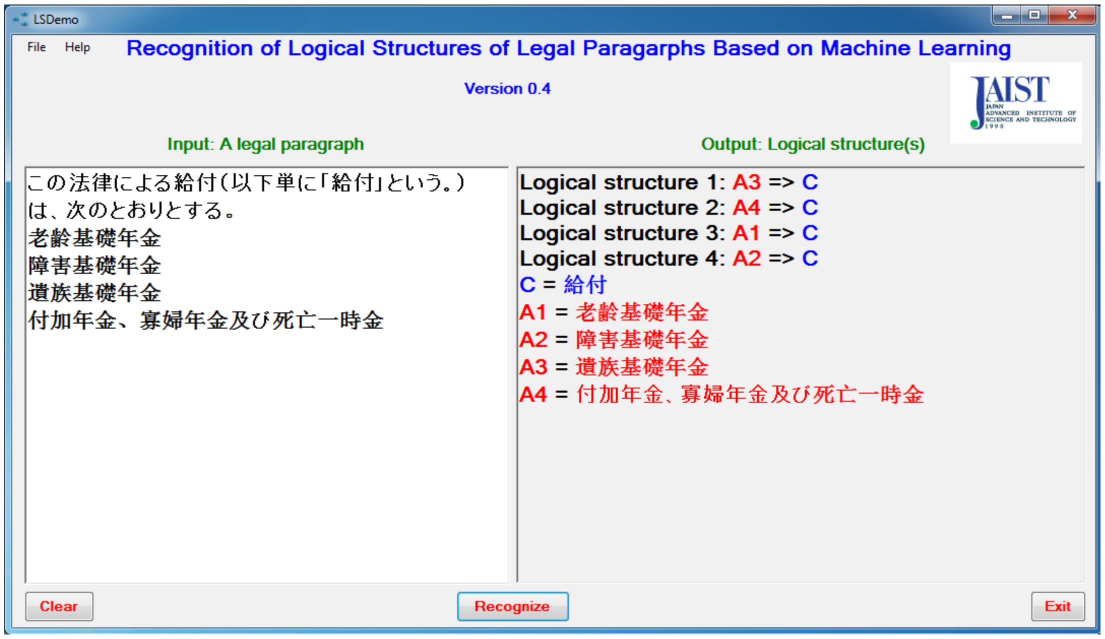
Figure A.2 demonstrates an example, where the input paragraph consists of one sentence. In this case, the output contains one logical structure with three logical parts. Figure A.3 shows another example, where the input paragraph contains multiple sentences. In this case, the output contains four logical parts, which are grouped into two logical structures.


LSDemo
File
Help
Recognition of Logical Structures of Legal Paragarphs Based on Machine Learning
Version 0.4
JAIST
Input: A legal paragraph
Output: Logical structure(s)
Clear
Recognize
Exit

この法律による年金の額は、国民の生活水準その他の諸事情に著しい変動が生じた場合には、変動後の諸事情に応ずるため、速やかに改定の措置が講ぜられなければならない。

Logical structure 1: A => T , C
T = この法律による年金の額は、
A = 国民の生活水準その他の諸事情に著しい変動が生じた場合には、
C = 変動後の諸事情に応ずるため、速やかに改定の措置が講ぜられなければならない。

Figure A.2: A running example with a paragraph consisting of one sentence.


LSDemo
File
Help
Recognition of Logical Structures of Legal Paragarphs Based on Machine Learning
Version 0.4
JAIST
Input: A legal paragraph
Output: Logical structure(s)
Clear
Recognize
Exit

この法律による給付(以下単に「給付」という。)は、次のとおりとする。
老齢基礎年金
障害基礎年金
遺族基礎年金
付加年金、寡婦年金及び死亡一時金

Logical structure 1: A3 => C
Logical structure 2: A4 => C
Logical structure 3: A1 => C
Logical structure 4: A2 => C
C = 給付
A1 = 老齢基礎年金
A2 = 障害基礎年金
A3 = 遺族基礎年金
A4 = 付加年金、寡婦年金及び死亡一時金

Figure A.3: A running example with a paragraph containing multiple sentences.

Bibliography

[1] R.P. Abelson and R.C. Schank. *Scripts, Plans, Goals, and Understanding: An Inquiry Into Human Knowledge Structures*. Psychology Press, 1977.

[2] E. Agirre, D. Cer, M. Diab, and A. Gonzalez-Agirre. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval)*, pages 385–393, 2012.

[3] G. Attardi and M. Ciaramita. Tree revision learning for dependency parsing. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 388–395, 2007.

[4] N.X. Bach. A study on recognition of requisite part and effectuation part in law sentences. Master's thesis, School of Information Science, Japan Advanced Institute of Science and Technology, 2011.

[5] N.X. Bach, N.L. Minh, and A. Shimazu. Exploring contributions of words to recognition of requisite part and effectuation part in law sentences. In *Proceedings of the 4th International Workshop on Juris-Informatics (JURISIN)*, pages 121–132, 2010.

[6] N.X. Bach, N.L. Minh, and A. Shimazu. RRE task: The task of recognition of requisite part and effectuation part in law sentences. *International Journal of Computer Processing Of Languages (IJCPOL)*, 23(2):109–130, 2011.

[7] N.X. Bach, N.L. Minh, and A. Shimazu. A reranking model for discourse segmentation using subtree features. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 160–168, 2012.

[8] N.X. Bach, N.L. Minh, and A. Shimazu. UDRST: A novel system for unlabeled discourse parsing in the RST framework. In *Proceedings of the 8th International Conference on Natural Language Processing (JapTAL)*, pages 250–261, 2012.

[9] R. Barzilay, K.R. McKeown, and M. Elhadad. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 550–557, 1999.

[10] J. Bateman, J. Kleinz, T. Kamps, and K. Reichenberger. Towards constructive text, diagram, and layout generation for information presentation. *Computational Linguistics*, 27(3):409–449, 2001.

[11] L. Bentivogli, I. Dagan, H.T. Dang, D. Giampiccolo, and B. Magnini. The fifth pascal recognizing textual entailment challenge. In *Proceedings of Text Analysis Conference (TAC)*, 2009.

[12] A.L. Berger, V.J.D. Pietra, and S.A.D. Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[13] G. Boella and L Di Caro. Extracting definitions and hypernym relations relying on syntactic dependencies and support vector machines. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 532–537, 2013.

[14] A. Borthwick. *A Maximum Entropy Approach to Named Entity Recognition.* PhD thesis, Computer Science Department, New York University, 1999.

[15] B.E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory (COLT)*, pages 144–152, 1992.

[16] R. Brighi, L. Lesmo, A. Mazzei, M. Palmirani, and D.P. Radicioni. Towards semantic interpretation of legal modifications through deep syntactic analysis. In *Proceedings of the 21st International Conference on Legal Knowledge and Information Systems (JURIX)*, pages 202–206, 2008.

[17] C. Bron and J. Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973.

[18] R.H. Byrd, J. Nocedal, and R.B. Schnabel. Representations of quasi-newton matrices and their use in limited memory methods. *Mathematical Programming*, 63(4): 129–156, 1994.

[19] C. Callison-Burch, P. Koehn, and M. Osborne. Improved statistical machine translation using paraphrases. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 17–24, 2006.

[20] L. Carlson and D. Marcu. Discourse tagging manual. Technical Report ISI-TR-545, ISI, 2001.

[21] L. Carlson, D. Marcu, and M.E. Okurowski. RST discourse treebank, 2002.

[22] X. Carreras and L. Marquez. Filtering-ranking perceptron learning for partial parsing. *Machine Learning*, 60(1-3):41–71, 2005.

[23] X. Carreras, L. Marquez, V. Punyakanok, and D. Roth. Learning and inference for clause identification. In *Proceedings of the 13th European Conference on Machine Learning (ECML)*, pages 35–47, 2002.

[24] Y.S. Chan and H.T. Ng. Maxsim: A maximum similarity metric for machine translation evaluation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technology (ACL-HLT)*, pages 55–62, 2008.

[25] D. Chiang. Hierarchical phrase-based translation. *Computational Linguistics*, 33 (2):201–228, 2007.

[26] C. Chih-Chung and L. Chih-Jen. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[27] J. Clarke and M. Lapata. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research (JAIR)*, 31:399–429, 2008.

[28] M. Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

[29] M. Collins and T. Koo. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70, 2005.

[30] S. Corston-Oliver. *Computing Representations of the Structure of Written Discourse*. PhD thesis, University of California, Santa Barbara, 1998.

[31] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[32] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[33] J.N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

[34] D. Das and N.A. Smith. Paraphrase identification as probabilistic quasi-synchronous recognition. 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 (ACL-IJCNLP)*, pages 468–476, 2009.

[35] L. Deléger and P Zweigenbaum. Extracting lay paraphrases of specialized expressions from monolingual comparable medical corpora. In *Proceedings of the 2nd Workshop on Building and Using Comparable Corpora: from Parallel to Non-parallel Corpora*, pages 2–10, 2009.

[36] P. Denis and J. Baldridge. Joint determination of anaphoricity and coreference resolution using integer programming. In *Proceedings of North American Chapter of the Association for Computational Linguistics - Human Language Technologies(NAACL-HLT)*, pages 236–243, 2007.

[37] M. Denkowski and M. Lavie. Extending the meteor machine translation metric to the phrase level. In *Proceedings of the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 250–253, 2010.

[38] G. Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the 2nd International Conference on Human Language Technology Research (HLT)*, pages 138–145, 2002.

[39] P.A. Duboue and J. Chu-Carroll. Answering the question you wish they had asked: The impact of paraphrasing for question answering. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 33–36, 2006.

[40] C. Dyer. Using a maximum entropy model to build segmentation lattices for MT. In *Proceedings of North American Chapter of the Association for Computational Linguistics - Human Language Technologies(NAACL-HLT)*, pages 406–414, 2009.

[41] A. Echihabi and D. Marcu. A noisy-channel approach to question answering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL)*, pages 16–23, 2003.

[42] S. Fernando and M. Stevenson. A semantic similarity approach to paraphrase detection. In *Proceedings of the Computational Linguistics UK (CLUK)*, 2008.

[43] A. Finch, Y.S. Hwang, and E. Sumita. Using machine translation evaluation techniques to determine sentence-level semantic equivalence. In *Proceedings of the 3rd International Workshop on Paraphrasing*, pages 17–24, 2005.

[44] R. Florian and D. Yarowsky. Modeling consensus: Classifier combination for word sense disambiguation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 25–32, 2002.

[45] G.D.Jr. Forney. The viterbi algorithm. *IEEE*, 61:268–278, 1973.

[46] A. Fraser, R. Wang, and H. Schütze. Rich bitext projection features for parse reranking. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 282–290, 2009.

[47] J. Ganitkevitch, C. Callison-Burch, C. Napoles, and B Van Durme. Learning sentential paraphrases from bilingual parallel corpora for text-to-text generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1168–1179, 2011.

[48] B.J. Grosz, A.K. Joshi, and S. Weinstein. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225, 1995.

[49] C. Grover, B. Hachey, I. Hughson, and C. Korycinski. Automatic summarisation of legal documents. In *Proceedings of the 9th International Conference on Artificial-Intelligence and Law (ICAIL)*, pages 243–251, 2003.

[50] N. Habash and A.E. Kholy. SEPIA: Surface span extension to syntactic dependency precision-based MT evaluation. In *Proceedings of the Workshop on Metrics for Machine Translation at AMTA*, 2008.

[51] A. Hanamoto, T. Matsuzaki, and J. Tsujii. Coordination structure analysis using dual decomposition. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 430–438, 2012.

[52] H. Hernault, P. Piwek, H. Prendinger, and M. Ishizuka. Generating dialogues for virtual agents using nested textual coherence relations. In *Proceedings of IVA*, pages 139–145, 2008.

[53] H. Hernault, D. Bollegala, and M. Ishizuka. A sequential model for discourse segmentation. In *Proceedings of the 11th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, pages 315–326, 2010.

[54] H. Hernault, H. Prendinger, D.A. duVerle, and M. Ishizuka. HILDA: A discourse parser using support vector machine classification. *Dialogue & Discourse*, 1(3):1–33, 2010.

[55] V. Hoste, I. Hendrickx, W. Daelemans, and A. Van Den Bosch. Parameter optimization for machine-learning of word sense disambiguation. *Natural Language Engineering*, 8(3), 2002.

[56] C.W. Hsu, C.C. Chang, and C.J. Lin. A practical guide to support vector classification, 2010. `www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.

[57] L. Huang. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 586–594, 2008.

[58] E.T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106 (4):620–630, 1957.

[59] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning (ECML)*, 1998.

[60] S. Joty, G. Carenini, and Raymond T. Ng. A novel discriminative framework for sentence-level discourse analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 904–915, 2012.

[61] S. Joty, G. Carenini, R. Ng, and Y. Mehdad. Combining intra- and multi-sentential rhetorical parsing for document-level discourse analysis. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 486–496, 2013.

[62] T. Katayama. The current status of the art of the 21st COE programs in the information sciences field. verifiable and evolvable e-society - realization of trustworthy e-society by computer science - (in Japanese). *Information Processing Society of Japan*, 46(5):515–521, 2005.

[63] T. Katayama. Legal engineering - an engineering approach to laws in e-society age. In *Proceedings of the 1st International Workshop on Juris-Informatics (JURISIN)*, 2007.

[64] T. Katayama, A. Shimazu, S. Tojo, K. Futatsugi, and K. Ochimizu. E-society and legal engineering (in Japanese). *Journal of the Japanese Society for Artificial Intelligence*, 23(4):529–536, 2008.

[65] Y. Kimura, M. Nakamura, and A. Shimazu. Treatment of legal sentences including itemized and referential expressions - towards translation into logical forms. *New Frontiers in Artificial Intelligence*, 5447 of LNAI:242–253, 2009.

[66] D. Klein and C. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 423–430, 2003.

[67] R. Koeling. Chunking with maximum entropy models. In *Proceedings of Conference on Computational Natural Language Learning (CoNLL)*, pages 139–141, 2000.

[68] T. Koo, A.M. Rush, M. Collins, T. Jaakkola, and D. Sontag. Dual decomposition for parsing with non-projective head automata. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1288–1298, 2010.

[69] Z. Kozareva and A. Montoyo. Paraphrase identification on the basis of supervised machine learning techniques. In *Proceedings of the 5th International Conference on Natural Language Processing (FinTAL)*, pages 524–533, 2006.

[70] T. Kudo. CRF++: Yet another CRF toolkit, 2010. Software available at `http://crfpp.sourceforge.net/`.

[71] T. Kudo and Y. Matsumoto. Chunking with support vector machines. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*, 2001.

[72] T. Kudo and Y. Matsumoto. Japanese dependency analysis using cascaded chunking. In *Proceedings of Conference on Computational Natural Language Learning (CoNLL)*, pages 63–69, 2002. Software available at `http://code.google.com/p/cabocha/`.

[73] T. Kudo, K. Yamamoto, and Y. Matsumoto. Applying conditional random fields to japanese morphological analysis. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 230–237, 2004.

[74] T. Kudo, J. Suzuki, and H. Isozaki. Boosting-based parse reranking with subtree features. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 189–196, 2005.

[75] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, pages 282–289, 2001.

[76] G. Lame. Using NLP techniques to identify legal ontology components: concepts and relations. *Artificial Intelligence and Law*, 12(4):379–396, 2004.

[77] Y.K. Lee and H.T. Ng. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 41–48, 2002.

[78] G. Leusch, N. Ueffing, and H. Ney. A novel string-to-string distance measure with applications to machine translation evaluation. In *Proceedings of the Ninth Machine Translation Summit (MT Summit IX)*, 2003.

[79] A. Louis, A. Joshi, and A. Nenkova. Discourse indicators for content selection in summarization. In *Proceedings of the 11th annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL)*, pages 147–156, 2010.

[80] N. Madnani, J. Tetreault, and M. Chodorow. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 182–190, 2012.

[81] W.C. Mann and S.A. Thompson. Rhetorical structure theory. toward a functional theory of text organization. *Text*, 8:243–281, 1988.

[82] D. Marcu. *The Theory and Practice of Discourse Parsing and Summarization.* MIT Press, Cambridge, 2000.

[83] M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. Building a large annotated corpus of english: The peen treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[84] A.F.T. Martins, N.A. Smith, and E.P. Xing. Concise integer linear programming formulations for dependency parsing. In *Proceedings of Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 342–350, 2009.

[85] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, pages 591–598, 2000.

[86] L.T. McCarty. Deep semantic interpretations of legal texts. In *Proceedings of the 11th international conference on Artificial intelligence and law (ICAIL)*, pages 217–224, 2007.

[87] R. Mihalcea, C. Corley, and C. Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI)*, pages 775–780, 2006.

[88] T.M. Mitchell. *Machine Learning.* MIT Press and McGraw Hill, 1997.

[89] E. Mjolsness and D. DeCoste. Machine learning for science: State of the art and future prospects. *Science*, 293(5537):2051–2055, 2001.

[90] M-F. Moens, E. Boiy, R.M. Palau, and C. Reed. Automatic detection of arguments in legal texts. In *Proceedings of the 11th International Conference on Artificial Intelligence and Law (ICAIL)*, pages 225–230, 2007.

[91] M. Mohri, A. Rostamizadeh, and A Talwalkar. *Foundations of Machine Learning.* MIT Press, 2012.

[92] M. Muramatsu, Y. Yasumura, and K. Nitta. A tagging tool for logical structure of legal sentences. Technical report of ieice, 2002.

[93] M. Murata, K. Uchimoto, Q. Ma, and H. Isahara. Bunsetsu identification using category-exclusive rules. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*, pages 565–571, 2000.

[94] T. Nakagawa, T. Kudo, and Y. Matsumoto. Revision learning and its application to part-of-speech tagging. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 497–504, 2002.

[95] M. Nakamura, S. Nobuoka, and A. Shimazu. Towards translation of legal sentences into logical forms. In *Proceedings of the 1st International Workshop on Juris-Informatics (JURISIN)*, 2007.

[96] A. Nenkova and K. McKeown. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2-3):103–233, 2011.

[97] L.M. Nguyen, A. Shimazu, and H.X. Phan. Semantic parsing with structured svm ensemble classification models. In *Proceedings of Joint 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL)*, pages 619–626, 2006.

[98] S. Niessen, F.J. Och, G. Leusch, and H. Ney. An evaluation tool for machine translation: Fast evaluation for mt research. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC)*, 2000.

[99] J. Nivre, J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of Conference on Computational Natural Language Learning (CoNLL)*, pages 221–225, 2006.

[100] J. Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.

[101] F.J. Och and H.Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 295–302, 2002.

[102] K. Pala, P. Rychly, and P. Smerk. Morphological analysis of law texts. In *Proceedings of the First Workshop on Recent Advances in Slavonic Natural Language Processing(RASLAN)*, pages 21–26, 2007.

[103] K. Pala, P. Rychly, and P. Smerk. Automatic identification of legal terms in czech legal texts. *Semantic Processing of Legal Texts*, pages 83–94, 2010.

[104] K. Papineni, S. Roukos, T. Ward, and W.J. Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, 2002.

[105] S. Parker. BADGER: A new machine translation metric. In *Proceedings of the Workshop on Metrics for Machine Translation at AMTA*, 2008.

[106] F. Peng and A. McCallum. Information extraction from research papers using conditional random fields. *Information Proceesing and Management*, 42(4):963–979, 2006.

[107] T. Proisl, P. Greiner, S. Evert, and B Kabashi. KLUE: Simple and robust methods for polarity classification. In *Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 395–401, 2013.

[108] V. Punyakanok, D. Roth, W. Yih, and D. Zimak. Semantic role labeling via integer linear programming inference. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pages 1346–1352, 2004.

[109] A. Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 133–142, 1996.

[110] M. Regneri and R. Wang. Using discourse information for paraphrase extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 916–927, 2012.

[111] V. Rus, P.M. McCarthy, M.C. Lintean, D.S. McNamara, and A.C. Graesser. Paraphrase identification with lexico-syntactic graph subsumption. In *Proceedings of the Twenty-First International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pages 201–206, 2008.

[112] A.M. Rush and M. Collins. A tutorial on dual decomposition and lagrangian relaxation for inference in natural language processing. *Journal of Artificial Intelligence Research (JAIR)*, 45(1):305–362, 2012.

[113] A.M. Rush, D. Sontag, M. Collins, and J. Tommi. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–11, 2010.

[114] M. Rushdi Saleh, M. T. Martín-Valdivia, A. Montejo-Ráez, and L. A. Ureña López. Experiments with SVM to classify opinions in different domains. *Expert Systems with Applications*, 38(12):14799–14804, 2011.

[115] S.R. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man and Cybernetics*, 21(3):660–674, 1991.

[116] K. Sagae. Analysis of discourse structure with syntactic dependencies and data-driven shift-reduce parsing. In *Proceedings of the 11th International Workshop on Parsing Technologies (IWPT)*, pages 81–84, 2009.

[117] J. Saias and P. Quaresma. A methodology to create legal ontologies in a logic programming based web information retrieval system. *Law and the Semantic Web*, pages 185–200, 2005.

[118] E.F. Tjong Kim Sang and F.D. Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh Conference on Natural Language Learning (CoNLL)*, pages 142–147, 2003.

[119] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 213–220, 2003.

[120] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*, 2006.

[121] M. Snover, N. Madnani, B. Dorr, and R. Schwartz. TER-Plus: Paraphrase, semantic, and alignment enhancements to translation edit rate. *Machine Translation*, 23 (23):117–127, 2009.

[122] R. Socher, E.H. Huang, J. Pennington, A.Y. Ng, and C.D. Manning. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems 24 (NIPS)*, pages 801–809, 2011.

[123] R. Soricut and D. Marcu. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 149–156, 2003.

[124] P. Spinosa, G. Giardiello, M. Cherubini, S. Marchi, G. Venturi, and S. Montemagni. NLP-based metadata extraction for legal text consolidation. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law (ICAIL)*, pages 40–49, 2009.

[125] R. Subba and B. Di Eugenio. Automatic discourse segmentation using neural networks. In *Proceedings of the Workshop on the Semantics and Pragmatics of Dialogue (SemDial)*, pages 189–190, 2007.

[126] M. Sun and J.Y. Chai. Discourse processing for context question answering based on linguistic knowledge. *Knowledge-Based Systems*, 20(6):511–526, 2007.

[127] M. Surdeanu and C.D. Manning. Ensemble models for dependency parsing: Cheap and good? In *Proceedings of the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 649–652, 2010.

[128] M. Surdeanu, M. Ciaramita, and H. Zaragoza. Learning to rank answers to non-factoid questions from web collections. *Computational Linguistics*, 37(2):351–383, 2011.

[129] C. Sutton and A. McCallum. *An Introduction to Conditional Random Fields for Relational Learning*. MIT Press, 2006.

[130] K. Takano, M. Nakamura, Y. Oyama, and A. Shimazu. Semantic analysis of paragraphs consisting of multiple sentences - towards development of a logical formulation system. In *Proceedings of the 23rd International Conference on Legal Knowledge and Information Systems (JURIX)*, pages 117–126, 2010.

[131] K. Tanaka. About semantic function of the legal-effect's restrictive part. *Natural Language*, 98(21):1–8, 1998.

[132] K. Tanaka, I. Kawazoe, and H. Narita. Standard structure of legal provisions - for the legal knowledge processing by natural language - (in Japanese). In *IPSJ Research Report on Natural Language Processing*, pages 79–86, 1993.

[133] L.X. Tang, S. Geva, A. Trotman, and Y. Xu. A voting mechanism for named entity translation in englishchinese question answering. In *Proceedings of the 4th International Workshop on Cross Lingual Information Access at COLING 2010*, pages 43–51, 2010.

[134] H.L. Thanh, G. Abeysinghe, and C. Huyck. Automated discourse segmentation by syntactic information and cue phrases. In *Proceedings of IASTED*, 2004.

[135] H.L. Thanh, G. Abeysinghe, and C. Huyck. Generating discourse structures for written texts. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pages 329–335, 2004.

[136] P. Thomas, M. Neves, T. Rocktäschel, and U Leser. WBI-DDI: Drug-drug interaction extraction using majority voting. In *Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 628–635, 2013.

[137] M. Tofiloski, J. Brooke, and M. Taboada. A syntactic and lexical-based discourse segmenter. In *Proceedings of the Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 77–80, 2009.

[138] Y. Tsuruoka. A simple C++ library for maximum entropy classification, 2006. Software available at `http://www-tsujii.is.s.u-tokyo.ac.jp/~tsuruoka/maxent/`.

[139] O. Uzuner, B. Katz, and T. Nahnsen. Using syntactic information to identify plagiarism. In *Proceedings of the 2nd Workshop on Building Educational Applications using Natural Language Processing*, pages 37–44, 2005.

[140] V.N. Vapnik. *Statistical Learning Theory.* Wiley-Interscience, 1998.

[141] G. Venturi. Legal language and legal knowledge management applications. *Semantic Processing of Legal Texts*, pages 3–26, 2010.

[142] M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. A model-theoretic coreference scoring scheme. In *Proceedings of MUC-6*, pages 45–52, 1995.

[143] J. Völker, S.F. Langa, and Y. Sure. Supporting the construction of spanish legal ontologies with Text2Onto. *Computable Models of the Law*, pages 105–112, 2008.

[144] S. Walter. Linguistic description and automatic extraction of definitions from german court decisions. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC)*, 2008.

[145] S. Walter and M. Pinkal. Automatic extraction of definitions from german court decisions. In *Proceedings of the COLING 2006 Workshop on Information Extraction Beyond The Document*, pages 20–28, 2006.

[146] S. Wan, R. Dras, M. Dale, and C. Paris. Using dependency-based features to take the para-farce out of paraphrase. In *Proceedings of the 2006 Australasian Language Technology Workshop (ALTA)*, pages 131–138, 2006.

[147] S. Wang and C Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 90–94, 2012.

[148] H. Wu and H. Wang. Improving statistical word alignment with ensemble methods. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*, pages 462–473, 2005.

[149] A. Wyner, R.M. Palau, M.F. Moens, and D. Milward. Approaches to text mining arguments from legal cases. *Semantic Processing of Legal Texts*, pages 60–79, 2010.

[150] M. Zhang and H Li. Tree kernel-based SVM with structured syntactic knowledge for BTG-based phrase reordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 698–707, 2009.

[151] H. Zhao and C. Kit. Parsing syntactic and semantic dependencies with two single-stage maximum entropy models. In *Proceedings of Conference on Computational Natural Language Learning (CoNLL)*, pages 203–207, 2008.

[152] C. Zhu, R.H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560, 1997.

[153] C. Zirn, M. Niepert, H. Stuckenschmidt, and M. Strube. Fine-grained sentiment analysis with structural features. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 336–344, 2011.

Publications

Book Chapter

[1] **Ngo Xuan Bach**, Kunihiko Hiraishi, Nguyen Le Minh, Akira Shimazu. A Joint Model for Vietnamese Part-of-Speech Tagging Using Dual Decomposition. *Knowledge-based Information Systems in Practice*, Springer (Accepted).

Journal Articles

[2] **Ngo Xuan Bach**, Nguyen Le Minh, Akira Shimazu. Exploiting Discourse Information to Identify Paraphrases. *Expert Systems with Applications*, 41(6):2832–2841, 2014.

[3] **Ngo Xuan Bach**, Nguyen Le Minh, Tran Thi Oanh, Akira Shimazu. A Two-Phase Framework for Learning Logical Structures of Paragraphs in Legal Articles. *ACM Transactions on Asian Language Information Processing (ACM TALIP)*, 12(1), article 3, 2013.

[4] **Ngo Xuan Bach**, Nguyen Le Minh, Akira Shimazu. RRE Task: The Task of Recognition of Requisite Part and Effectuation Part in Law Sentences. *International Journal of Computer Processing Of Languages (IJCPOL)*, 23(2):109–130, 2011

[5] Oanh Thi Tran, **Bach Xuan Ngo**, Minh Le Nguyen, Akira Shimazu. Automated reference resolution in legal texts. *Artificial Intelligence and Law*, 22(1):29–60, 2014.

[6] Minh Quang Nhat Pham, Minh Le Nguyen, **Bach Xuan Ngo**, Akira Shimazu. A learning-to-rank method for information updating task. *Applied Intelligence*, 37(4):499–510, 2012.

[7] Oanh Thi Tran, **Bach Xuan Ngo**, Minh Le Nguyen, Akira Shimazu. A List-wise Approach to Coreference Resolution Using Learning-to-rank. Submitted to Knowledge-Based Systems.

Referred Conference Papers

[8] **Ngo Xuan Bach**, Kunihiko Hiraishi, Nguyen Le Minh, Akira Shimazu. Dual Decomposition for Vietnamese Part-of-Speech Tagging. In *Proceedings of the 17th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES)*, Procedia Computer Science, pages 123–131, 2013.

[9] **Ngo Xuan Bach**, Nguyen Le Minh, Akira Shimazu. EDU-Based Similarity for Paraphrase Identification. In *Proceedings of the 18th International Conference on Applications of Natural Language to Information Systems (NLDB)*, LNCS 7934, pages 65–76, 2013 (**Received Outstanding Paper Award for Young Researchers in Computer & Communications 2013, awarded by NEC C&C Foundation**).

[10] **Ngo Xuan Bach**, Nguyen Le Minh, Akira Shimazu. UDRST: A Novel System for Unlabeled Discourse Parsing in the RST Framework. In *Proceedings of the 8th International Conference on Natural Language Processing (JapTAL)*, LNCS/LNAI 7614, pages 250–261, 2012.

[11] **Ngo Xuan Bach**, Nguyen Le Minh, Akira Shimazu. A Reranking Model for Discourse Segmentation using Subtree Features. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL 2012 Conference)*, pages 160–168, 2012.

[12] **Ngo Xuan Bach**, Nguyen Le Minh, Tran Thi Oanh, Akira Shimazu. Learning Logical Structures of Paragraphs in Legal Articles. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 20–28, 2011.

[13] Oanh Thi Tran, **Bach Xuan Ngo**, Minh Le Nguyen, Akira Shimazu. Answering Legal Questions by Mining Reference Information. In *Proceedings of the 7th International Workshop on Juris-informatics (JURISIN)*, 2013.

[14] Oanh Thi Tran, **Bach Xuan Ngo**, Minh Le Nguyen, Akira Shimazu. Reference Resolution in Japanese Legal Texts at Passage Levels. In *Proceedings of the 5th International Conference on Knowledge and Systems Engineering (KSE)*, Springer-Verlag, pages 237–249, 2013.

[15] Oanh Thi Tran, **Bach Xuan Ngo**, Minh Le Nguyen, Akira Shimazu. A Listwise Approach to Coreference Resolution in Multiple Languages. In *Proceedings of the 25th Pacific Asia Conference on Language, Information and Computation (PACLIC)*, pages 400–409, 2011.

[16] Le Minh Nguyen, **Ngo Xuan Bach**, Akira Shimazu. Supervised and Semi-Supervised Sequence Learning for Recognition of Requisite Part and Effectuation Part in Law Sentences. In *Proceedings of the 9th International Workshop on Finite-State Methods and Natural Language Processing (FSMNLP)*, pages 21–29, 2011.