Learning Logical Structures of Paragraphs in Legal Articles

Ngo Xuan Bach, Nguyen Le Minh, Tran Thi Oanh, Akira Shimazu
School of Information Science,
Japan Advanced Institute of Science and Technology,
1-1 Asahidai, Nomi, Ishikawa, 923-1292, Japan
{bachnx,nguyenml,oanhtt,shimazu}@jaist.ac.jp

Abstract

This paper presents a new task, learning logical structures of paragraphs in legal articles, which is studied in research on Legal Engineering (Katayama, 2007). The goals of this task are recognizing logical parts of law sentences in a paragraph, and then grouping related logical parts into some logical structures of formulas, which describe logical relations between logical parts. We present a two-phase framework to learn logical structures of paragraphs in legal articles. 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 sub-graphs in a weighted-edge complete graph, where each node corresponds to a logical part, and a complete sub-graph corresponds to a logical structure. We also present an integer linear programming formulation for this optimization problem. Our models achieve 74.37% in recognizing logical parts, 79.59% in recognizing logical structures, and 55.73% in the whole task on the Japanese National Pension Law corpus.

1 Introduction

Legal Engineering (Katayama, 2007) is a new research field which aims to achieve a trustworthy electronic society. 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 kinds of documents. 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* (Bach, 2011a; Bach et al., 2011b; Tanaka eta al., 1993). At the paragraph level, a paragraph usually contains a main sentence[1] and one or more subordinate sentences (Takano et al., 2010).

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 easily understand 1) what does a law sentence say? 2) what cases in which the law sentence can be applied? and 3) what subjects are related to the provision described in the law sentence? 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.

There have been some studies analyzing logical structures of legal texts. (Bach et al., 2011b) presents the RRE task[2], which recognizes the logical structure of law sentences. (Bach et al., 2010) describes an investigation on contributions of words to the RRE task. (Kimura et al., 2009) focuses on dealing with legal sentences including itemized and referential expressions. These works, however, only analyze logical structures of legal texts at the sentence level. At the paragraph

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

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

Proceedings of the 5th International Joint Conference on Natural Language Processing, pages 20–28,
Chiang Mai, Thailand, November 8 – 13, 2011.

level, (Takano et al., 2010) classifies 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.

In this paper, we present a task of learning logical structures of legal articles at the paragraph level. We propose a two-phase framework to complete the task. We also describe experimental results on real legal data.

Our main contributions can be summarized in the following points:

- Introducing a new task to legal text processing, *learning logical structures of paragraphs in legal articles*.
- Presenting an annotated corpus for the task, the *Japanese National Pension Law corpus*.
- Proposing a two-phase framework and providing solutions to solve the task.
- Evaluating our framework on the real annotated corpus.

The rest of this paper is organized as follows. Section 2 describes our task and its two sub-tasks: *recognition of logical parts* and *recognition of logical structures*. In Section 3, we present our framework and proposed solutions. Experimental results on real legal articles are described in Section 4. Finally, Section 5 gives some conclusions.

2 Formulation

Learning 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 usually 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 (Bach, 2011a; Bach et al., 2011b)[3]. Usually, consequent parts describes a law provision, antecedent parts describes cases in which the law provision can be applied, and topic parts describe subjects which are related to the law provision. In this paper, a logical structure can be defined as a set of some related logical parts.

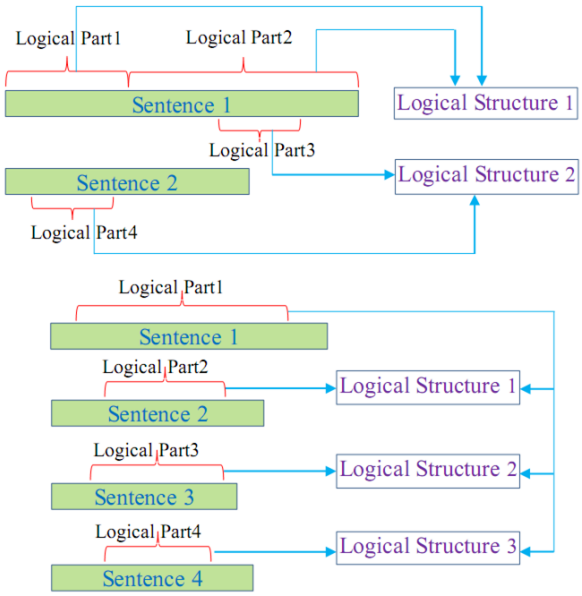

Figure 1: Two cases of inputs and outputs of the task.

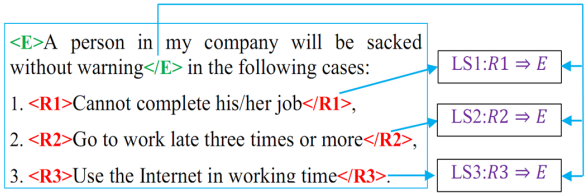

Figure 2: An example in natural language (E means *Effectuation part*, R means *Requisite part*, and LS means *Logical Structure*).

Figure 1 shows two cases of the inputs and outputs of the task. In the first case, 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, the input is a paragraph consisting of four sentences, and the outputs are four logical parts, which are grouped into three logical structures. An example in natural language[4] is presented in Figure 2.

2.1 Sub-Task 1: Recognition of Logical Parts

Let s be a law sentence in the law sentence space S, then s can be represented by a sequence of words $s = [w_1 w_2 \dots w_n]$. A legal paragraph x in the legal paragraph space X is a sequence of law sentences $x = [s_1 s_2 \dots s_l]$, where $s_i \in S, \forall i = 1, 2, \dots, l$. For each paragraph x, we denote a log-

[3] 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 this paper.

[4] Because law sentences are very long and complicated, we use toy sentences to illustrate the task.

ical 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 sub-task, 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 $\prec$. Formally, the solution space can be described as follows: $Y = \{y \subseteq P | \forall u, v \in y, u \not\sim v\}$. The learning problem in this sub-task 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
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 $RepO$.

Compared with previous works (Bach et al., 2011b), we introduce three new kinds of logical parts: Ob, $RepO$, and $RepR$.

2.2 Sub-Task 2: Recognition of Logical Structures

In the second sub-task, the goal is to recognize a set of logical structures given a set of 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 sub-task, each vertex of the graph corresponds to a logical part, and a complete sub-graph 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 sub-graph 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 sub-graph 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 sub-graphs 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 the 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.

3 Proposed Solutions

3.1 Multi-layer Sequence Learning for Logical Part Recognition

This sub-section presents our model for recognizing logical parts. We consider the recognition problem as a multi-layer sequence learning problem. First, we give some related notions.

Let s be a law sentence, and P be the set of logical parts of s, $P = \{p_1, p_2, \ldots, p_m\}$. $Layer^1(s)$ (outer most 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

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

Figure 3 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 $Layer^3$.

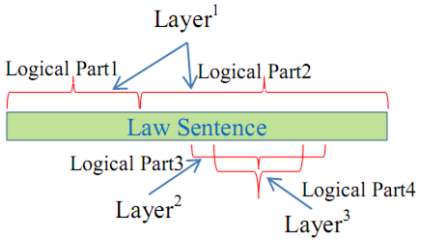

Figure 3: A law sentence with logical parts in three layers.

Input Sentence	w_1	w_2	...	w_{k-1}	w_k	w_{k+1}	w_{k+2}	...	w_{n-1}	w_n
Layer1	Topic Part 2 (T_2)					Consequence Part (C)				
Layer2						Left Side Part (EL)				
Labels in Layer1	I-T_2	I-T_2	...	I-T_2	E-T_2	I-C	I-C	...	I-C	E-C
Labels in Layer2						O	I-EL	...	E-EL	O

Figure 4: An example of labeling in the multi-layer model.

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 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.

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 (Lafferty et al., 2001; Kudo, CRF toolkit). 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 $Layer^{i-1}$.

3.2 ILP for Recognizing Logical Structures

Suppose that G' is a sub-graph 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 sub-graphs (or cliques) of G' will satisfy all the *minimal*, *complete*, *maximal*, and *significant* constraints. We also note that, a set of cliques that satisfies all these four constraints will form a sub-graph that has two properties like properties of G'.

Let Λ be the set of all such sub-graphs G' of G, the sub-task 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.

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 (Punyakanok, 2004), coreference resolution (Denis and Baldridge, 2007), summarization (Clarke and Lapata, 2008), dependency parsing (Martins et al., 2009), 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 available tools[5]

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

make it a practical solution for many NLP problems (Martins et al., 2009).

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 $\{x_{ij}\}_{1 \leq i < j \leq N}$. The values of $\{x_{ij}\}$ are set as follows. If $(i, j) \in e(G')$ then $x_{ij} = 1$, otherwise $x_{ij} = 0$. ILP formulations for the first step can be described as follows:

//- - - - - - - - Objective function - - - - - - - -//

$$Maximize : \sum_{1 \leq i < j \leq N} f(i, j) * x_{ij} \quad (1)$$

//- - - - - - - - - - Constraints - - - - - - - - - -//

$$Integer : \{x_{ij}\}_{1 \leq i < j \leq N}. \quad (2)$$

$$0 \leq x_{ij} \leq 1, (1 \leq i < j \leq N). \quad (3)$$

$$\sum_{i=1}^{j-1} x_{ij} + \sum_{k=j+1}^{N} x_{jk} \geq 1, (1 \leq j \leq N). \quad (4)$$

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 (Bron and Kerbosch, 1973).

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

$f(p_1, p_2) = Prob(C(p_1, p_2) = +1) - 0.5$.

Function f will receive a value from -0.5 to $+0.5$, and it equals to zero in the case that the classifier assigns the same probability to $+1$ and -1.

4 Experiments

4.1 Corpus

We have built a corpus, Japanese National Pension Law (JNPL) corpus, which consists of 83 legal articles[6] of Japanese national pension law. The architecture of JNPL is shown in Figure 5. The law consists of articles, articles consist of paragraphs, and paragraphs contain sentences. A sentence may belong to items, sub-items, or sub-sub-items of a paragraph.

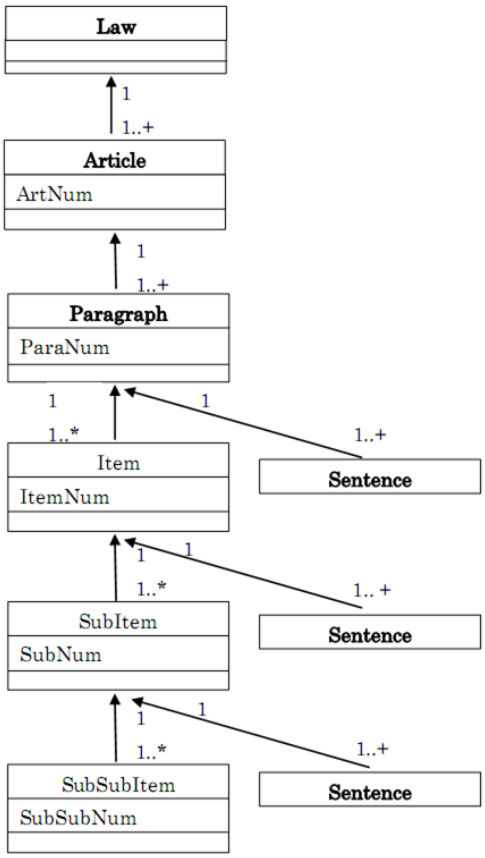

Figure 5: The architecture of JNPL.

Figure 6 illustrates 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.

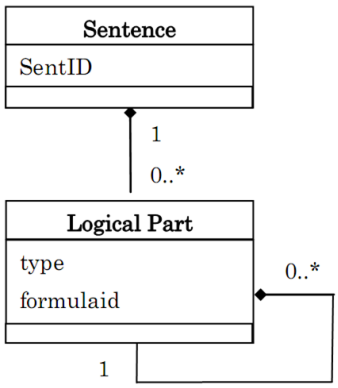

Figure 6: Relationship between a sentence and logical parts.

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 be-

[6]Because building corpus is an expensive and time-consuming task, we only annotate a part of JNPL.

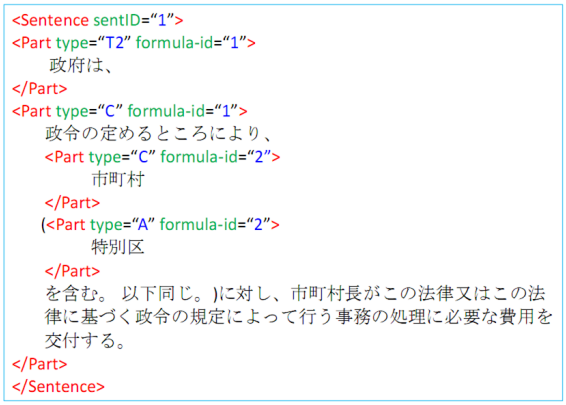

Figure 7: An annotated sentence in the JNPL corpus. The sentence contains two logical structures with four logical parts.

long to one logical structure). An example of annotated sentence in the JNPL corpus is shown in Figure 7.

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

Table 1 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.

4.2 Evaluation Methods

We divided the JNLP corpus into 10 sets, and conducted 10-fold cross-validation tests. For the first sub-task, we evaluated the performance of our system by precision, recall, and F_1 scores as follows:

$$precision = \frac{|correct\ parts|}{|predicted\ parts|}, recall = \frac{|correct\ parts|}{|actual\ parts|},$$
$$F_1 = \frac{2*precision*recall}{precision+recall}.$$

For the second sub-task, we used MUC precision, recall, and F_1 scores as described in (Vilain et al., 1995). We summarize them here for clarity.

Let $P_1, P_2, \ldots, P_n$ be n predicted logical structures, and $G_1, G_2, \ldots, G_m$ be the correct answers or gold logical structures. To calculate recall, for each gold logical structure $G_i (i = 1, 2, \ldots, m)$, let $k(G_i)$ be the smallest number such that there exist $k(G_i)$ predicted structures $P^i_1, P^i_2, \ldots, P^i_{k(G_i)}$ which satisfy $G_i \subseteq \cup_{j=1}^{k(G_i)} P^i_j$:

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

To calculate 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 sub-task.

4.3 Experiments on Sub-Task 1

4.3.1 Baseline: Filter-Ranking Perceptron Algorithm

We chose the Filter-Ranking (FR) Perceptron algorithm proposed by (Carreras and Marquez, 2005; Carreras et al., 2002) as our baseline model because of its effectiveness on phrase recognition problems, especially on problems that accept the *embedded* relationship[7]. We use FR-perceptron algorithm to recognize logical parts in law sentences one by one in an input paragraph.

For *beginning/end* predictors, we got features of words, POS tags, and Bunsetsu[8] tags in a window size 2. Moreover, with *beginning* predictor, we used 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 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)
3. Word (POS) uni-gram, word (POS) bi-gram, and word (POS) tri-gram.

4.3.2 Experimental Results

In our experiments, we focus on paragraphs in Type A, B, and C defined in (Takano et al., 2010). In these types, the first sentence 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

[7]We re-implement the FR-perceptron algorithm by ourself.

[8]In Japanese, a Bunsetsu is an unit of sentence which is similar to a chunk in English.

Table 1: 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

Table 2: Experimental results for Sub-task 1 on the JNLP corpus(W:Word; P: POS tag; B: Bunsetsu tag)

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

sentences usually contain logical parts in two layers.

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 (Lafferty et al., 2001; Kudo, CRF toolkit).

Experimental results on the JNPL corpus are described in Table 2. 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 (Kudo, Cabocha), a Japanese morphological and syntactic analyzer. The best model (word and Bunsetsu tag features) achieved 74.37% in F_1 score. It improves 11.04% in F_1 score (30.11% in error rate) compared with the baseline model.

Table 3 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.

Table 3: 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**

4.4 Experiments on Sub-Task 2

4.4.1 Baseline: a Heuristic Algorithm

Our baseline is a heuristic algorithm to solve this sub-task on graphs. This is an approximate algorithm which satisfies *minimal*, *complete*, *maximal*, and *significant* constraints. 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.

In the first 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 8. The maximum value on an edge is -0.1, so the first logical structure will contain node 1 and node 3. The second logical structure contains node 2 and node 4[9].

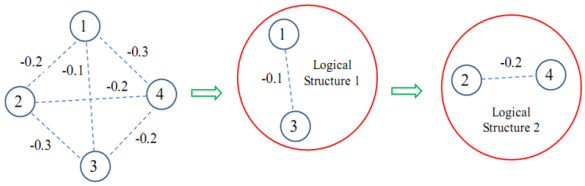

Figure 8: An example of the first case.

In the second case, we first consider the subgraph which only contains non-negative value edges. In this sub-graph, we repeatedly build logical structures with as many logical parts as possi-

[9]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.

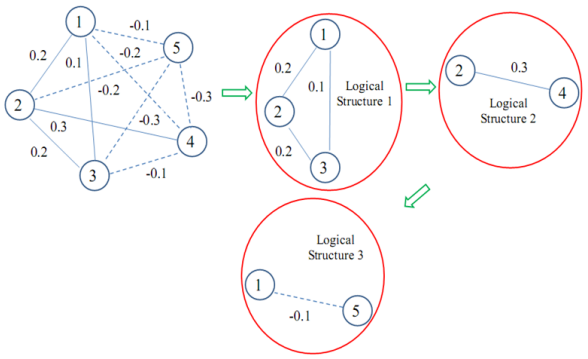

Figure 9: An example of the second case.

ble. After building successfully a logical structure, we remove all the nodes and the edges according to it on the graph. When have no positive edge, we will build logical structures with exactly two logical parts.

An example of the second case is illustrated in Figure 9. First, we consider the sub-graph with positive edges. This sub-graph 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. 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 $\{1, 5\}$.

4.4.2 Experimental Results

In our experiments, to learn a maximum entropy binary classification we used the implementation of Tsuruoka (Tsuruoka, MEM). 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.

We conducted experiments on this sub-task in two settings. In the first setting, we used annotated logical parts (gold inputs) as the inputs to the system. The purpose of this experiment is to evaluate the performance of the graph-based method on Sub-task 2. In the second setting, predicted logical parts (end-to-end) outputted by the Sub-task 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.

Table 4: Experiments on Sub-task 2

	Gold Input Setting		
Model	Prec(%)	Recall(%)	F_1(%)
Heuristic	**81.24**	71.19	75.89
ILP	76.56	**82.87**	**79.59**
	End-to-End Setting		
Model	Prec(%)	Recall(%)	F_1(%)
Heuristic	54.88	47.84	51.12
ILP	**57.51**	**54.06**	**55.73**

In the second setting, end-to-end setting, because input logical parts may differ from the correct logical parts, we need to modify the MUC scores. Let $P_1, P_2, \ldots, P_n$ be n predicted logical structures, and $G_1, G_2, \ldots, G_m$ be the gold logical structures. For each gold logical 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 input 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.

Table 4 shows experimental results on the second sub-task. The ILP model outperformed the baseline model in both settings. It improved 3.70% in the F_1 score (15.35% in error rate) in the gold-input setting, and 4.61% in the F_1 score (9.43% in error rate) in the end-to-end setting compared with the baseline model (heuristic algorithm).

5 Conclusion

We have introduced the task of learning logical structures of paragraphs in legal articles, a new task which has been studied in research on Legal Engineering. 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 with multi-layer sequence learning model and ILP formulation to complete the task. Our results provide a baseline for further researches on this interesting task.

In the future, we will continue to improve this task. On the other hand, we also investigate the task of translating legal articles into logical and formal representations.

Acknowledgments

This work was partly supported by the 21st Century COE program 'Verifiable and Evolvable e-Society', Grant-in-Aid for Scientific Research, Education and Research Center for Trustworthy e-Society, and JAIST Overseas Training Program for 3D Program Students.

We would like to give special thanks to Kenji Takano and Yoshiko Oyama, who analyzed law sentences and built the corpus, and the reviewers, who gave us valuable comments.

References

N.X. Bach. 2011a. A Study on Recognition of Requisite Part and Effectuation Part in Law Sentences. *Master Thesis*, School of Information Science, Japan Advanced Institute of Science and Technology.

N.X. Bach, N.L. Minh, A. Shimazu. 2011b. RRE Task: The Task of Recognition of Requisite Part and Effectuation Part in Law Sentences. In *International Journal of Computer Processing Of Languages (IJCPOL)*, Volume 23, Number 2.

N.X. Bach, N.L. Minh, A. Shimazu. 2010. Exploring Contributions of Words to Recognition of Requisite Part and Effectuation Part in Law Sentences. In *Proceedings of JURISIN*, pp. 121-132.

C. Bron and J. Kerbosch. 1973. Algorithm 457: Finding All Cliques of an Undirected Graph. In *Communications of the ACM, Volume 16, Issue 9*, pp. 575-577.

X. Carreras and L. Marquez. 2005. Filtering-Ranking Perceptron Learning for Partial Parsing. In *Machine Learning*, Volume 60, Issue 1-3, pp. 41-71.

X. Carreras, L. Marquez, V. Punyakanok, D. Roth. 2002. Learning and Inference for Clause Identification. In *Proceedings of ECML*, pp. 35-47.

J. Clarke and M. Lapata. 2008. Global Inference for Sentence Compression: An Integer Linear Programming Approach. In *Journal of Artificial Intelligence Research (JAIR)*, Volume 31, pp. 399-429.

P. Denis and J. Baldridge. 2007. Joint Determination of Anaphoricity and Coreference Resolution Using Integer Programming. In *Proceedings of HLT-NAACL*, pp. 236-243.

T. Katayama. 2007. Legal Engineering - An Engineering Approach to Laws in e-Society Age. In *Proceedings of JURISIN.*

Y. Kimura, M. Nakamura, A. Shimazu. Treatment of Legal Sentences Including Itemized and Referential Expressions - Towards Translation into Logical Forms. *New Frontiers in Artificial Intelligence*, volume 5447 of LNAI, pp.242-253.

T. Kudo. Yet Another Japanese Dependency Structure Analyzer. *http://chasen.org/ taku/software/cabocha/.*

T. Kudo. CRF++: Yet Another CRF toolkit. *http://crfpp.sourceforge.net/.*

J. Lafferty, A. McCallum, F. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of ICML*, pp.282-289.

A.F.T. Martins, N.A. Smith, E.P. Xing. 2009. Concise Integer Linear Programming Formulations for Dependency Parsing. In *Proceedings of ACL*, pp.342-350.

M. Nakamura, S. Nobuoka, A. Shimazu. 2007. Towards Translation of Legal Sentences into Logical Forms. In *Proceedings of JURISIN*.

V. Punyakanok, D. Roth, W. Yih, D. Zimak. 2004. Semantic Role Labeling Via Integer Linear Programming Inference. In *Proceedings of COLING*, pp. 1346-1352.

K. Takano, M. Nakamura, Y. Oyama, A. Shimazu. 2010. Semantic Analysis of Paragraphs Consisting of Multiple Sentences - Towards Development of a Logical Formulation System. In *Proceedings of JURIX,* pp. 117-126.

K. Tanaka, I. Kawazoe, H. Narita. 1993 Standard Structure of Legal Provisions - for the Legal Knowledge Processing by Natural Language - (in Japanese). In *IPSJ Research Report on Natural Language Processing*, pp. 79-86.

Y. Tsuruoka. A simple C++ library for maximum entropy classification. *http://www-tsujii.is.s.u-tokyo.ac.jp/ tsuruoka/maxent/.*

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, L. Hirschman. 1995. A Model-Theoretic Coreference Scoring Scheme. In *Proceedings of MUC-6*, pp. 45-52.