JAIST Repository
https://dspace.jaist.ac.jp/

Title	Study on Extracting Conceptual Structures from Legal Texts
Author(s)	ファム, アンカム Jr
Citation	
Issue Date	2012-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/10414
Rights	
Description	Supervisor: Prof. Akira Shimazu, 情報科学研究科, 修士

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

Study on Extracting Conceptual Structures from Legal Texts

By **Pham Anh Cam**

A thesis is submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of

Professor Akira Shimazu

March, 2012

Study on Extracting Conceptual Structures from Legal Texts

By **Pham Anh Cam (1010057)**

A thesis is submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Professor Akira Shimazu
and approved by
Professor Akira Shimazu
Associate Professor Kiyoaki Shirai
Professor Satoshi Tojo

February, 2012 (Submitted)

Study on Extracting Conceptual Structures from Legal Texts

by

Pham Anh Cam (1010057)

School of Information Science

Japan Advanced Institute of Science and Technology

February 06, 2012

Keywords*: Japanese laws, conceptual model, UML class*

Abstract

Law is a system of rules and guidelines which are enforced through social institutions to govern behavior, wherever possible. Therefore, to understand and to check the correctness and the consistence of a law is very important but normally very difficult because of long and difficult texts. Our research aims to take a hand in this work by modeling Japanese laws. Such laws can be seen as specifications of social systems. Our purpose is to clarify a method for extracting a model (conceptual model) of social system which such laws represent.

Our research is conducted as the study of legal engineering, which aims to exam and verify whether a law has been established appropriately according to its purpose, 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 which are to help experts make complete and consistent laws; and to design information systems which work based on laws.

To model the social system represented by Japanese laws, we use a conceptual model. The term conceptual model may be used to refer to model which is represented by concepts or related concepts which are formed after a conceptualization process in the mind. In order to express the result of conceptual modeling, we need a modeling language. A modeling language can be graphical or textual. There are several elaborate graphical notations that could be used for conceptual modeling, ranging from those used in the past for Entity-Relationship

Model [Chen 1975], conceptual graphs [Sowa 1984] to those presently used for object modeling [OMG 1997].

Recently there are some researches about using UML class diagrams to model natural language. However, all of them are applied to English language only and all methods used a semi-formal language as a medium. Different from these above methods, our method is used for Japanese language, concentrate on Japanese legal texts; and we directly extract UML classes from dependencies of Japanese legal terms which are extracted based on Cabocha parse output. Besides, because of a huge part of extracted classes, we don't represent classes graphically but textually.

Our model is built by the following way: the common nouns are extracted to become the UML classes. These classes have attributes, methods and relations extracted correspondingly from the dependencies of legal terms as below. The verbs are extracted to become methods of classes. The modifier nouns, verbs and adjectives are extracted to become attributes of classes. The relations between nouns are extracted to become relations between classes. These classes then are represented textually. There are 3 steps to transform Japanese legal texts to UML class components. First, we parse Japanese law text corpus using Cabocha. Second, we extract dependencies of legal terms from Cabocha output. Third, we transform the dependencies to UML class components.

Our research contribution is that our model can be used as a tool to understand the law easily. Instead of reading a difficult law, people can take a look to the model to find out the related terms, actions as well as the relations between the objectives of the law – legal terms. Moreover, the relations between legal objectives are modeled; the logic of the law is also clearly expressed so that we can check the correctness and consistence of a law easily. We can also design the information systems based on laws using our conceptual model.

The challenges of this research are how to extract correctly and completely relations between Japanese legal terms and how to represent these terms with their characteristics and relations clearly. The prior problem is solved by careful analysis and extraction of the most important Japanese dependencies. The later problem is solved by representing classes with their attributes, methods and relation textually. Graphical representation is not used because there is a huge part of classes are extracted from each law.

Acknowledgements

First, I would like to thank my advisor, Professor Akira Shimazu who gave me a chance to know and receive Asiajinzai scholarship; supported me in 6 months of Japanese studying and 2 years of Master course. Thanks to his encouragement, I could begin, continue and finish the study in JAIST. I could study not only the knowledge of natural language processing in particular but also the thinking way from his kind, guide and experiences in general.

Second, I would like to thank my co-advisor, Professor Shirai Kiyoaki. His comments for my research help me a lot on improving my research and communication skills. I also could study a lot from his enthusiasm on research and teaching.

Third, I would like to thank Professor Ochimizu, Tera sensei and all my other Japanese and subject teachers who taught me a lot in 2 years and 6 months in JAIST. Thanks to the guides of professors, I could obtain some more knowledge and thinking ways.

Forth, I would like to thank Assistant Professor Nguyen Le Minh who helped me a lot in researching as long as in the life in JAIST. I will never forget his always support from the first day I came to JAIST.

I also would like to thank Yoshida and Shinkai coordinators, 11 Hokuriku companies who taught me a lot about Japanese business manner.

I also would like to thank the staffs in Student service section and all other sections who supported me a lot in student life.

I also would like to thank all my Vietnamese and foreigner friends in JAIST who always beside and encourage me. Thanks to the hearts of all, I have a calm and useful time in JAIST.

Last but not least, I would like to thank my family who are living in Vietnam. To be able to see their faces and chat with them every day is my biggest encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Motivation

The law is a social system which determines our society. From this point, it is necessary to understand laws clearly. Unfortunately, because of long and difficult texts, laws are normally difficult to understand for most of us. Because the models have giant power in helping us understand many phenomena easily and clearly, we should model the social systems laws represent in some ways to make laws clearer and thus easier to understand by using a conceptual modeling language. Modeling legal text is even more valuable if it is also used to check the correctness and the consistence of a law or to generate information systems based on laws. Considering all above arguments, to extract a conceptual model of a law is a useful and feasible research. However; according to our study, there are some approaches to extract conceptual model (for example, UML class diagrams) from natural language but there are no researches in extracting from Japanese language, especially, Japanese legal texts. Therefore, we proposed a new method to extract a conceptual model of a social system which Japanese laws represent. We will extract the conceptual model which includes legal terms go along with their characteristics and relations between them. Our research is conducted as the study of legal engineering.

1.2. Introduction

The purpose of my research is to clarify a method for extracting a conceptual model of a social system which Japanese laws represent.

In the next parts of this chapter, I will explain an introduction of legal engineering in the 1.2.1 section, a definition of a social system in the 1.2.2 section, a definition of a conceptual model in section 1.2.3, a brief introduction of my method in section 1.2.4, my main contributions in section 1.2.5 and challenges in section 1.2.6.

1.2.1. Introduction of legal engineering

Legal engineering aims to exam and verify whether a law has been established appropriately according to its purpose, 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 which are to help experts make complete and consistent laws; and to design an information system which works based on laws [1][2].

1.2.2. Definition of social system

Although Vilfredo Pareto had used the term, "social system" earlier but only as a sketch and not as an overall analytical scheme, the first who formulated a systematic theory of social system was Talcott Parsons where it was a part of his AGIL paradigm yet the social system is only a subsystem (I - subsystem) of what Parsons calls action theory [3]. He proposed that societal action systems are composed of four interrelated and interpenetrating subsystems: the behavioral systems of its members (A), the personality systems of those members (G), the society as a system of social organization (I) and the cultural system of that society (L). According to his definition, "*a social system consists in a plurality of individual actors interacting with each other in a situation which has at least a physical or environmental aspect, actors who are motivated in terms of a tendency to the "optimization of gratification" and whose relation to their situations, including each other, is defined and mediated in terms of a system of culturally structured and shared symbols.* ".

1.2.3. Definition of conceptual model

A short example of conceptual model can help better define them. We all know that the models have giant power in helping us understand many phenomena easily and clearly. People process information every time. This processing turns out to be a conceptual model of how things in our surrounding environment work. For example, why red sun-sets and blue skies can be explained by following model:

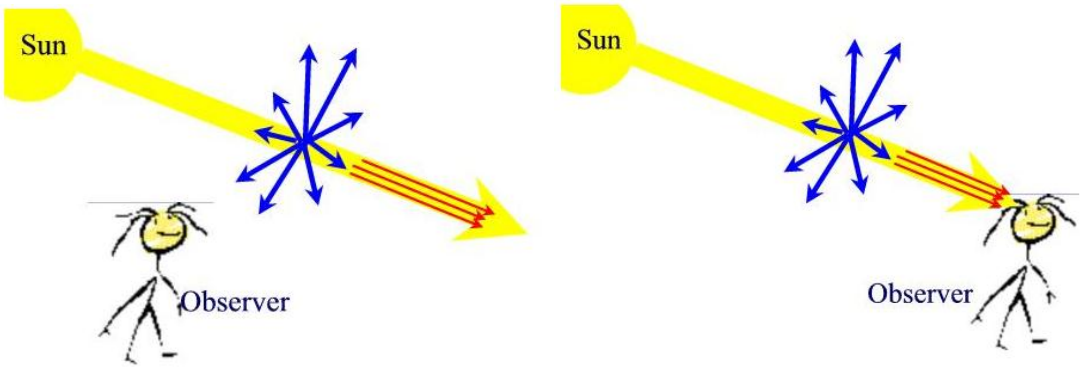


Figure 1.1 Red sun-sets and blue skies model

The figure on the left shows that the perceived color of light reaching your eyes in this position is blue. The reason for this is that the atmospheric scattering of blue light away from the forward direction is much stronger than for red light. The figure on the right shows that the perceived color of light reaching your eyes when looking at the sun near sunset is red for essentially the same reason; much less of the red light is scattered out of the direct beam compared to the larger scattering for blue. The sunlight has the deepest color of red right at sunset when the rays travel through the greatest amount of air before reaching your eyes. Red sunsets are also enhanced by particulate matter in the atmosphere.

In the most general sense, a model is anything used in any way to represent anything else. Some models are physical objects, for instance, a toy model which may be assembled, and may even be made to work like the object it represents. They are used to help us know and understand the subject matter they represent.

The term conceptual model may be used to refer to models which are represented by concepts or related concepts which are formed after a conceptualization process in the mind [4]. Conceptual models represent human intentions or semantics. Conceptualization from observation of physical existence and conceptual modeling are the necessary means which human employ to think and solve problems.

1.2.3.1. Type and scope of conceptual models

Conceptual models (models that are conceptual) range in type from the more concrete, such as the mental image of a familiar physical object, to the formal generality and abstractness of mathematical models which do not appear to the mind as an image. Conceptual models also range in terms of the scope of the subject matter that they are taken to represent. A model may, for instance, represent a single thing (e.g. the Statue of Liberty), whole classes of things (e.g. the electron), and even very vast domains of subject matter such as the physical universe. The variety and scope of conceptual models is due to the variety of purposes had by the people using them.

i) Models in philosophy and science

- Metaphysical models

- Epistemological models

ii) Models in systems architecture: System models

iii) Models in information system design

- Conceptual models of human activity systems

- Logico-linguistic models

iv) Data models: Domain model

1.2.3.2. Representation of a conceptual model in computer science

Conceptual modeling is needed to form a description of the domain of application at hand. In order to express the result of conceptual modeling, we need a modeling language.

A modeling language is any artificial language that can be used to express information or knowledge or systems in a structure that is defined by a consistent set of rules. A modeling language can be graphical or textual. Graphical modeling languages use a diagram technique with named symbols that represent concepts and lines that connect the symbols and represent relationships and various other graphical notations to represent constraints. Textual modeling languages typically use standardized keywords accompanied by parameters to make computer-interpretable expressions. [5]

There are several elaborate graphical notations that could be used for conceptual modeling, ranging from those used in the past for Entity-Relationship Model [ERM 1976], conceptual graphs [Sowa 1982] to those presently used for object modeling [UML 1997] or [ORM 1998], [OCML 1998], [ConML2011].

i) Entity - Relationship Model (ERM)

According to Peter Chen's 1976 research [6], in software engineering, an entity-relationship model (ER model for short) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion. Diagrams created by this process are called entity-relationship diagrams or ER diagrams.

The building blocks of this model are entities, relationships, and attributes.

Example of an ERM model

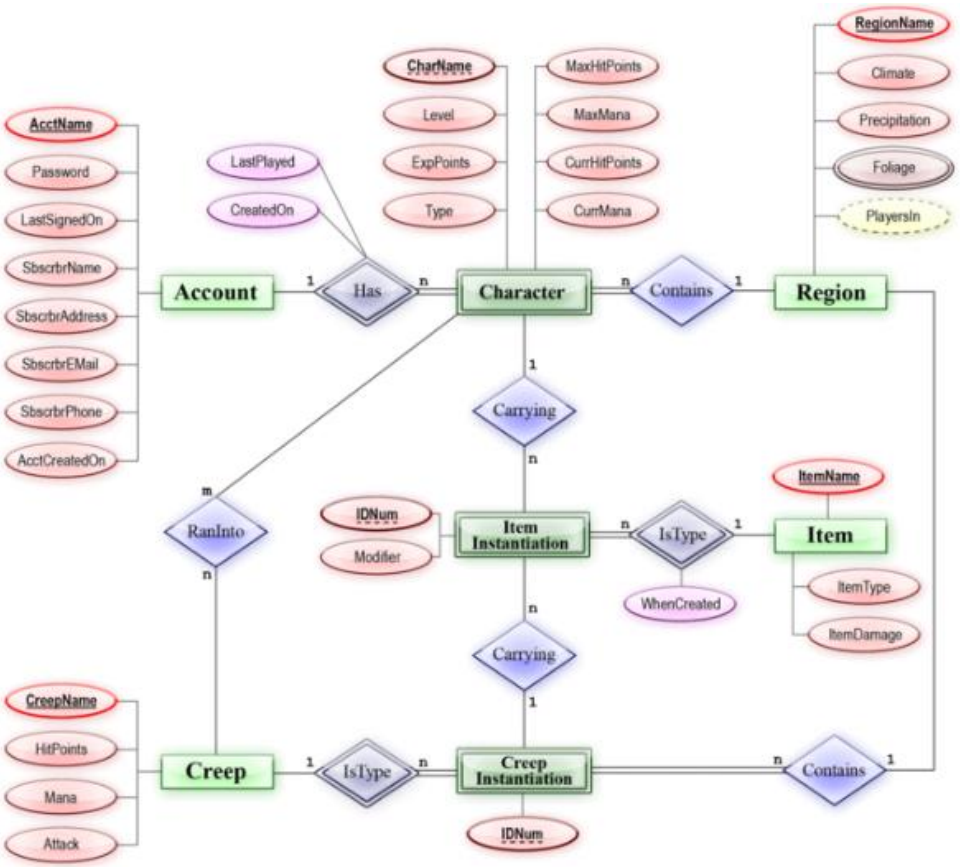


Figure 1.2 An ERM model

ii) Conceptual graphs (CGs)

Conceptual graphs (CGs) are a system of logic proposed by Sowa, 1984 [3] based on the existential graphs of Charles Sanders Peirce and the semantic networks of artificial intelligence. They express meaning in a form that is logically precise, humanly readable, and computationally tractable. With a direct mapping to language, conceptual graphs serve as an intermediate language for translating computer-oriented formalisms to and from natural languages. With their graphic representation, they serve as a readable, but formal design and specification language. CGs have been implemented in a variety of projects for information retrieval, database design, expert systems, and natural language processing.

Example of a conceptual graph:

"John is going to Boston by bus"

Figure 3 shows a conceptual graph with four concepts: [Go], [Person: John], [City: Boston], and [Bus]. It has three conceptual relations: (Agnt) relates [Go] to the agent

John, (Dest) relates [Go] to the destination Boston, and (Inst) relates [Go] to the instrument bus.

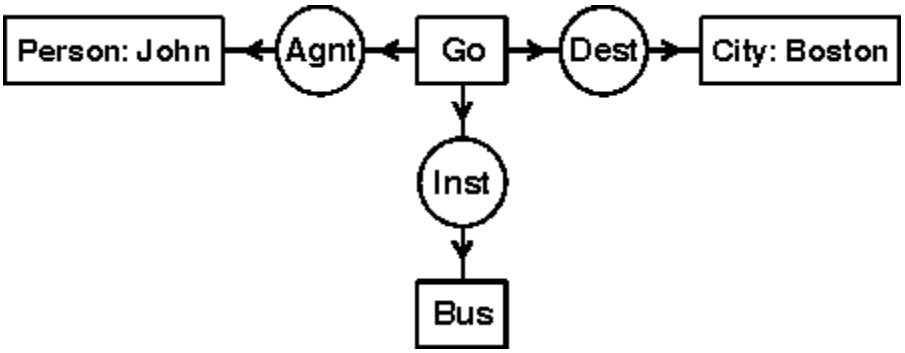


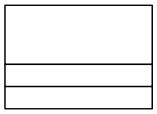
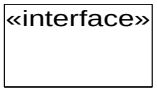
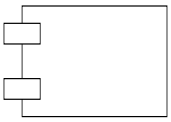
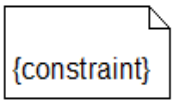
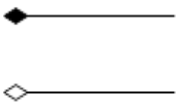
Figure 1.3 A conceptual graph

iii) OMG's Unified Modeling Language™ (UML®)

The OMG's Unified Modeling Language™ (UML®) [8] helps specify, visualize, and document models of software systems, including their structure and design, in a way that meets all of these requirements. (UML can be used for business modeling and modeling of other non-software systems too.) Using any one of the large number of UML-based tools on the market, we can analyze our future application's requirements and design a solution that meets them, representing the results using UML 2.0's thirteen standard diagram types.

We can model just about any type of application, running on any type and combination of hardware, operating system, programming language, and network, in UML. We can do other useful things with UML too: For example, some tools analyze existing source code (or, some claim, object code!) and reverse-engineer it into a set of UML diagrams; execute UML models interpretively or generate program language code from UML; generate Test and Verification Suites from UML models.

UML 2.0 defines thirteen types of diagrams, divided into three categories: Six diagram types represent static application structure; three represent general types of behavior; and four represent different aspects of interactions. The key element in a class model is class. Below are core elements of class diagram.

Construct	Description	Syntax
class	a description of a set of objects that share the same attributes, operations, methods, relationships and semantics.	
interface	a named set of operations that characterize the behavior of an element.	
component	a modular, replaceable and significant part of a system that packages implementation and exposes a set of interfaces.	
node	a run-time physical object that represents a computational resource.	
constraint	a semantic condition or restriction	
association	a relationship between two or more classifiers that involves connections among their instances.	
aggregation	A special form of association that specifies a whole-part relationship between the aggregate (whole) and the component part.	
generalization	a taxonomic relationship between a more general and a more specific element.	

dependency	a relationship between two modeling elements, in which a change to one modeling element (the independent element) will affect the other modeling element (the dependent element).	
realization	a relationship between a specification and its implementation.	

Table 1.1 Core elements of class diagram

iv) Object Role Modeling (ORM)

Object Role Modeling (ORM) [9] is a method for designing and querying database models at the conceptual level, where the application is described in terms easily understood by non-technical users. In practice, ORM data models often capture more business rules, and are easier to validate and evolve than data models in other approaches

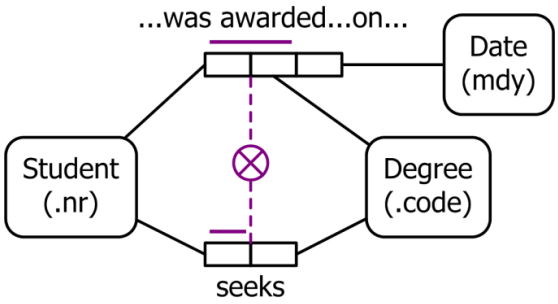


Figure 1.4 An ORM model

This is a fact-oriented modeling approach for specifying, transforming, and querying information at a conceptual level. Unlike Entity-Relationship modeling and Unified Modeling Language class diagrams, fact-oriented modeling is attribute-free, treating all elementary facts as relationships. For information modeling, fact-oriented graphical notations are typically far more expressive than other notations.

v) Operational Conceptual Modeling Language (OCML)

The OCML modeling language [10] supports the construction of knowledge models by means of several types of constructs. It allows the specification and

operationalization of functions, relations, classes, instances and rules. It also includes mechanisms for defining ontologies and problem solving methods, the main technologies developed in the knowledge modeling area. About a dozen projects in KMi are currently using OCML to provide modeling support for applications in areas such as knowledge management, ontology development, e-commerce and knowledge based system development. OCML modeling is also supported by a large library of reusable models, providing a useful resource for the knowledge modeling community. This library can be accessed through the WebOnto editor.

```
(def-relation RANGE (?f-r ?relation)
  "The range of a function or a binary relation is a relation which is
   true for any possible output of the function or second argument of
   the binary relation"
  :no-op (:iff-def (or
                    (and (function ?f-r)
                         (forall (?args ?result)
                                 (=> (= (apply ?f-r ?args) ?result)
                                     (holds ?relation ?result))))
                    (and (binary-relation ?f-r)
                         (forall (?x ?y)
                                 (=> (holds ?f-r ?x ?y))
                                     (holds ?relation ?y))))))
```

Figure 1.5 An OCML relation option

vi) Conceptual modeling languages (ConML)

Conceptual modeling languages (ConML) [11] is easily usable by non-experts in information technologies; this, in turn, imposes certain implications with regard to simplicity and independence of implementation details. It is capable of expressing information related to these following aspects without putting a burden on the user: subjectivity and temporality, rarely taken into account by conventional modeling languages. It is also understandable and familiar to external parties as possible, aiding to the transfer of knowledge and expertise between domains.

The characteristics of the Conceptual Modelling Language (ConML) are presented below. It is capable of representing static (i.e. structural) models, using the object-oriented paradigm. It is oriented towards conceptual modeling rather than implementation. It extends the conventional object-oriented paradigm in order to accommodate temporality and subjectivity aspects. It is easily affordable to non-experts in information technologies. This means that it must exhibit high syntactic, semantic and notational simplicity. The associated notation is easily used by hand (on paper or whiteboard, for instance), as well as on a computer (on screen or hard copy).

As long as it is viable, it keeps syntactic, semantic and notational compatibility with UML.

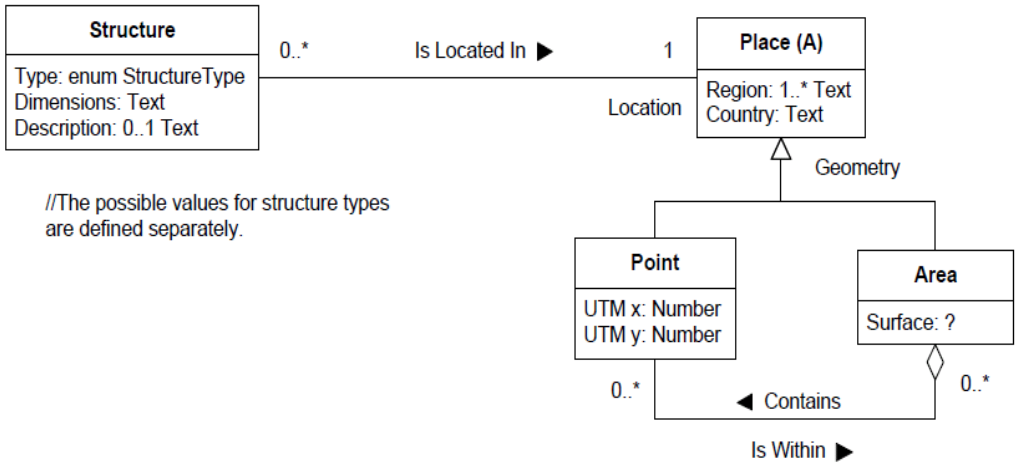


Figure 1.6 A ConML Model

1.2.3.3. Operational model

In contrast of conceptual model, operational models are models that represent problems for the operational level of management.

The difference between conceptual model and operational model can be seen by following example of weather forecast. The operational model is a detail, exact model, not in a concept level as a conceptual level

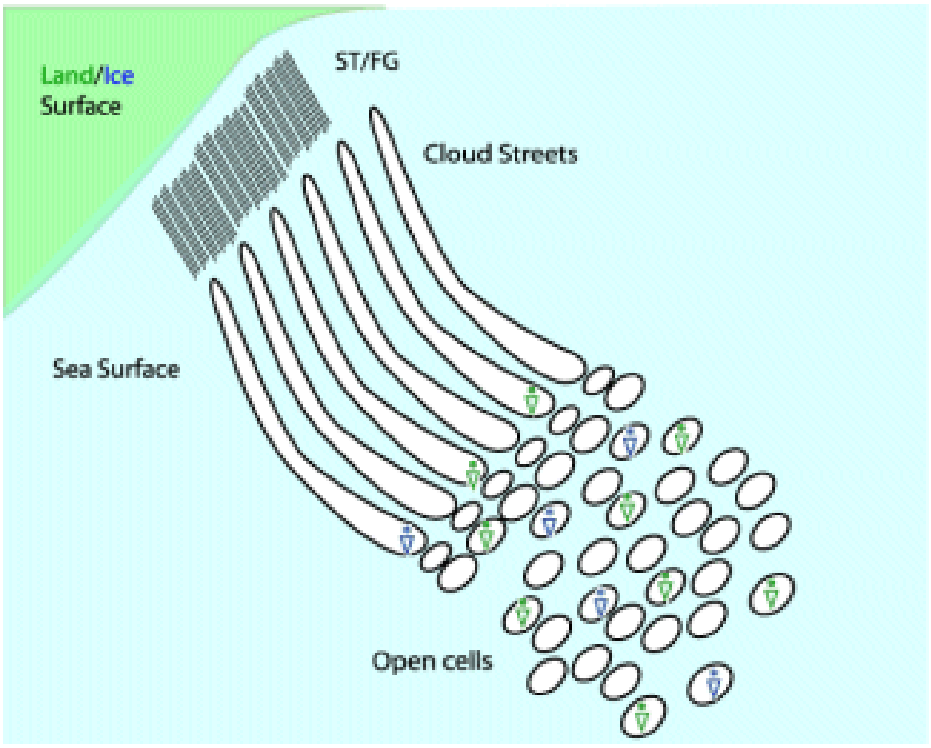


Figure 1.7 Conceptual model: Cloud Streets - Weather Events

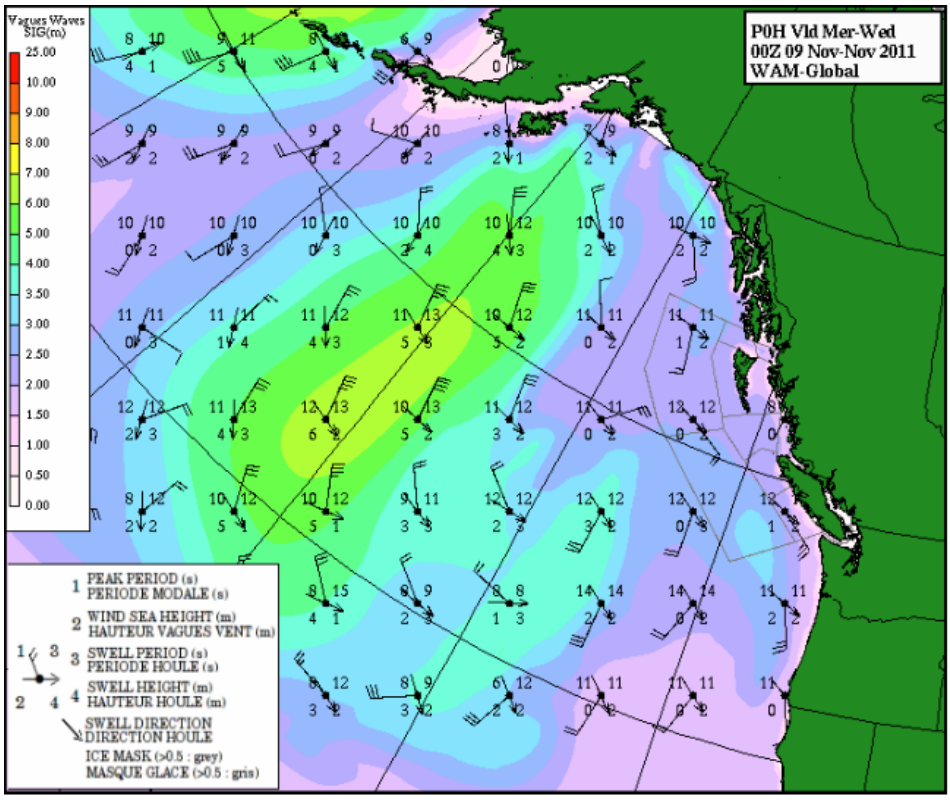


Figure 1.8 Operational Model Forecasts

1.2.4. Introduction of method

We propose a method to extract conceptual model of a social system which laws represent. Our model consists of classes. Each class has attributes, methods and some relations with other classes. There are 3 steps to transform Japanese legal texts to UML class components. First, we parse Japanese law text corpus using Cabocha [12]. Second, we extract dependencies of legal terms from Cabocha output. Third, we transform the dependencies to UML class components.

We started the experiment from parsing legal text using Cabocha. We will use token information: token base, pos, ctype in the next phase to extract the dependencies between tokens.

The next phase is dependency extraction.

Nouns, verbs and adjectives in the sentences are extracted in phrases. For example, “労働保険”, “開催された”, “必要な” phrases are extracted. The POS tagging of the first token in the noun phrase is extracted as the POS tagging of this phrase. The POS tagging of the verb/adjective token in the verb/adjective phrase is extracted as the POS tagging of this phrase.

According to the representation of Stanford parser [13][14], we extracted five types of dependencies: dep(noun1_pos, noun2_pos, verb), dep(noun_pos, verb), no(noun1_pos, noun2), no(noun_pos, verb) and no(noun_pos, adjective).

The dep dependency shows the relation between nouns through a verb. In the sentence which has two or three noun components: subject, direct object or indirect object, we extract the dependency dep(noun1, noun2, verb) in which the nouns are the corresponding components. In the sentence which only has only one noun component (either subject or direct object or indirect object), we extract the dependency dep(noun, verb).

The no dependency is the relation between a noun and its characteristic. The characteristic may be a modifier noun, verb or adjective.

The last phase is the transformation from Japanese dependencies to UML classes.

The class names are extracted from the noun phrases.

The attributes of the class are extracted from noun1 の noun2 dependency and compound nouns. With the noun1 の noun2 dependency, noun2 is the attribute of

noun1. For example, from the phrase エネルギーの使用, we can extract 使用 as the attribute of エネルギー. With the compound noun, the adjective modifier or verb modifier also can become the attribute of the main noun. For example, in 必要な時 compound noun, adjective modifier 必要な becomes the attribute of noun 時 or in 開発された日 compound nount, verb modifier 開始された becomes the attribute of noun 日.

The methods of the class are extracted from the verbs corresponding to the noun – class name. In our research, in a sentence, because the nouns can act as subjects, direct objects or indirect objects, the verbs of the sentence become the method of all subjects, direct objects and indirect objects. For example, in the sentence, この法律は、平成十八年四月一日から施行する, 施行する becomes the method of 法律 and 平成十八年四月一日 classes.

The relations of the class are also extracted from the verbs. We define the binary dependency of subject - direct object, subject – indirect object and direct object – indirect object in each sentence by the following way: Class Name: エネルギー Relation: めぐる with class 法律. The unary dependency of subject, direct object and indirect object with it self is represented by: Class Name: エネルギーRelation: 使用する,消費する with class エネルギー.

1.2.5. Contributions

There are some ways our model can be used

i) To help users understand law easily

Instead of reading a difficult law, people can take a look to my model to find out the related terms and actions of a legal term (using attributes and methods information) and the relations between the objectives of the law – legal terms.

ii) To help users check the correctness and consistent of the law

My model can be used to check that whether the law expresses the social system correctly and consistently or not. The logic of the law is clearly expressed in the model so that we can check the correctness and consistence of a law. The relations between objectives are modeled so that we can clearly see the conflicts of objectives if existin.

iii) To become a core part of an information system based on laws

An information system (IS) is any combination of information technology and people's activities that support operations, management, and decision making. The laws define the rules determine the actions of human: People can do what and cannot do what. Therefore, the laws are good conceptual structure which forms the core part of an information system. From this point, we can develop an information system based on laws using our model.

1.2.6. Challenges

The challenges of this research are on dependency extraction and UML class representation steps. The most difficult thing is how to extract correctly and completely relations between Japanese legal terms. This is not a simple work because it is difficult for me whose Japanese is not the mother language to capture all important Japanese grammar rules, especially in the condition that the corpus is legal sentences which are long ones with complicated grammar structures. Our work needs careful analysis of Japanese grammar in general and Japanese legal structures in particular to extract good dependency rules. As a result, 5 dependencies between nouns: dependencies between subject, direct object and indirect object and dependencies between nouns and modifiers are extracted. The other challenge is that there are hundreds of nouns in a law to be extracted to classes and thus there are a huge part of classes be extracted. Therefore, instead of representing these classes graphically, we represent them textually but very clearly.

The remainder of this thesis is organized as follows.

Chapter 2 introduces some related works. We will introduce some natural language to UML class diagrams transformation researches; Unsupervised Semantic Parsing approach to get clusters and relations of clusters using Markov Logic network; and Cabocha, a Japanese dependency parser.

Chapter 3 describes our method in three steps: Cabocha parsing, Dependency extraction and UML transformation. The result is also reported.

Chapter 4 is the evaluation part which includes internal and external evaluation.

Chapter 5 is the conclusion which summarizes our research.

Chapter 2

Background

As we proposed before, in our research, there are 3 steps to transform Japanese legal texts to UML class components. First, we parse Japanese law text corpus using Cabocha. Second, according to Stanford representation, we extract dependencies of legal terms from Cabocha output. Third, we transform the dependencies to UML class components. Related to this approach, there are some researches introduced below.

The first type of related researches is natural language to UML class diagrams transformation researches. The authors transform natural language to a semi-formal language and then to UML class diagrams. Mathias Kleiner et al [15] and Imran Sarwar Bajwa et al [16] are the typical researchers of transformation of text to SBVR (Semantic of Business Vocabulary and Rules) and then from SBVR to UML approach. In another approach, Narayan Debnath [17] proposed five transformation rules for defining the UML class diagram from ATL model.

Different from these above methods, our method is used for Japanese language and instead of using a semi-formal language as a medium; we directly extract UML classes from dependencies of Japanese legal terms which are extracted based on Cabocha parse output. Besides, because of a huge part of extracted classes, we don't represent classes graphically but textually.

The second type of related researches is Unsupervised Semantic Parsing [18] using Stanford Dependency Parsing [13] [14] as the input and Markov Logic Network [19] to create the network of related clusters.

Because Stanford Dependency Parsing is only for English language, we build Japanese Dependency Parsing by ourselves according to Stanford representation. Also, we don't use logical form as a medium of raw legal text to legal term relation representation. At the first time, we intended to use Unsupervised Semantic Parsing as a tool to extract legal terms and their relations by the following way: The classes are extracted as clusters of syntactic variation of the same meaning legal phrases. The relations of classes are extracted from the relation of above clusters using Markov Logic Network learned by Unsupervised Semantic Parsing. However; while USP is developed for English language, though we already generated the input

correspondingly by extracting Japanese dependencies according to Stanford Dependency Parse, USP cannot run appropriately with Japanese dependencies. Some errors occurred with long Japanese legal sentences and then USP cannot create cluster of the same meaning legal phrases. Moreover, the relations which are represented by Markov Logic Network is not clear. Therefore, we decided to extract directly the relations between legal terms directly from our Japanese dependencies we made before which were intended to use as the input of USP.

The next parts of this chapter are structured as below. The first part which includes Section 2.1, Section 2.2, Section 2.3 is an introduction of 3 natural language to UML class diagrams transformation approaches. The second part, Section 2.is an introduction of Markov Logic Network, Stanford Typed Dependency Parser and Unsupervised Semantic Parsing. The third part, Section 2.5 is an introduction of Cabocha, a Japanese dependency parser.

2.1. SBVR2UML: A Challenging Transformation

In this paper, they proposed that UML is a de-facto standard used for generating the software models. UML supports visualization of the software artifacts. To generate a UML diagram, a software engineer has to collect software requirements in a natural language (such as English) or a semi-formal language (such as SBVR), manually analyze the requirements and then manually generate the class diagrams in an available CASE tool. However, by automatically transforming SBVR Software requirements to UML can seriously share burden of a system analyst and can improve the quality and robustness of software modeling phase. Therefore, they presented the challenging aspect of model transformation from SBVR to UML. The presented approach takes input the software requirements specified in SBVR syntax, parses the input specification, extracts the UML ingredients such as classes, methods, attributes, associations, etc and finally generate the visual representation of the extracted information. The presented approach is fully automated. The presented approach is explained via an example.

i) SBVR Specification

A typical SBVR representation is based on SBVR business vocabulary and SBVR business rules. In SBVR, all the specific terms and definitions of concepts used by an organization or community in course of business are treated as vocabulary. Common examples of SBVR vocabulary are object types, individual concepts, characteristics, fact types, etc. In SBVR, a formal representation under a

business jurisdiction is called a SBVR rule. SBVR rules can be of two types: Structural Rule (used to expresses structure or operation of a particular business entity) and Behavioral Rule (used to express the conduct of a business entity).

ii) Transformation model from SBVR to UML

This section explains the used approach to automatically map SBVR representation i.e. SBVR business rules to a UML class model. To map SBVR to a UML class model, they have to extract SBVR vocabulary from given SBVR rules and then map the SBVR vocabulary to basic elements of a UML class model (such as classes, associations, etc.) and finally generate a graphical representation of class model. The used approach works in following 5 phases:

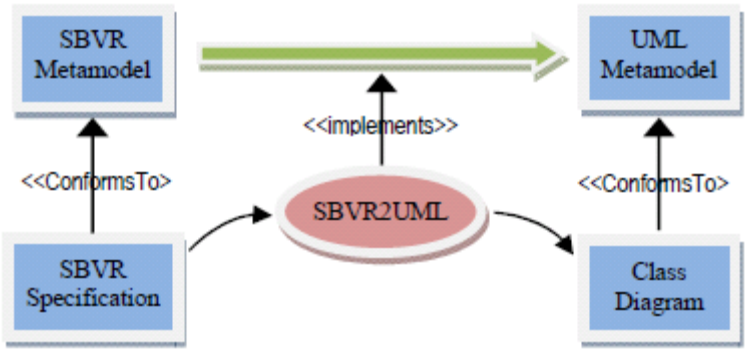


Figure 2.1 SBVR to UML Transformation Framework

Their models can be summarized in the following table.

Natural Language	SBVR Metamodel Element	UML Metamodel Element
Common nouns	Object Types	Class
Proper nouns	Individual Concepts	Object
Auxiliary + Action verbs	Verb Concepts	Class Method
Noun + Verb / Noun + Verb + Noun / Associative, Pragmatic Relations	Unary / Binary / Associative Fact Types	Association

Is-property-of, Possessed nouns	Characteristic	Class Attribute
Indefinite articles, plural nouns and cardinal numbers	Quantifications	Cardinalities
“is-part-of”, “included-in”, “belong-to”	Partitive Fact Type	Generalization
“is-category-of”, “is-typeof”, “is-kind-of”	Categorization Fact Types	Aggregation

Table 2.1 The mapping from Natural language to SBVR and UML metamodels

2.2. Parsing SBVR-Based Controlled Languages

In this article, they proposed an original method for extracting a SBVR terminal model out of a controlled English text and then transform it into a UML class diagram. They describe a model-driven engineering approach in which constraint-programming based search is combined with model transformation. The use of an advanced resolution technique (configuration) as an operation on models allows for non-deterministic parsing and language flexibility. In addition to the theoretical results, preliminary experiments on a running example are provided.

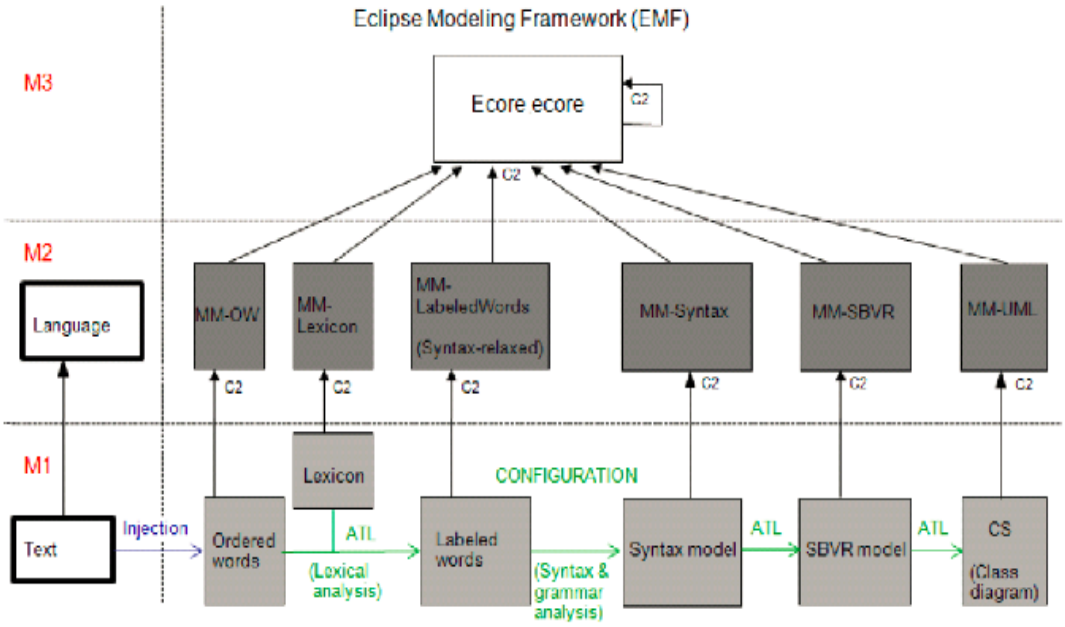


Figure 2.2 Process overview of Parsing SBVR-Based Controlled Languages

Syntactical Categorization constraints

Each verb phrase for which the heading verb is transitive is composed of at least one noun phrase. This constraint applies to the relation isComposedOf of a category:

A verb phrase is always preceded by a noun phrase. The constraint applies to the attributes “begin” and “end” of categories, which are obtained from their associated word(s):

i) The subject of an active verb occurs before the verb phrase:

ii) The head of a verb’s subject shares the same plural:

iii) A transitive verb expresses a fact type.:

iv) The head of a subject of a verb expresses either an object type or an individual concept.

2.3. An ATL Transformation from Natural Language Requirements Models to Business Models of a MDA Project

MDA is a software development framework where the core is a set of automatic transformations of models. One of these models, the CIM (Computer Independent Model), is used to define the business process model. Though a complete automatic construction of the CIM is not possible, they have proposed the integration of some natural language requirements models and they have defined a strategy to derive a CIM from these models. In this paper, they present an ATL transformation that implements this strategy to obtain a UML class diagram representing a preliminary CIM from requirements models. This transformation fits with MDA approach.

They proposed five ATL transformation rules for defining the UML class diagram.

i) Transformation Subject to Class

ii) Transformation Object to Class

iii) Transformation Subject Behavioral Response to Method

iv) Transformation Subject Information to Method Parameter

v) Transformation LELRelationships to Class Relationships

2.4. Unsupervised Semantic Parsing (USP)

2.4.1. Markov Logic Network

2.4.1.1. Markov Logic's Intuition

Problem of uncertainty

In most real world scenarios, logic formulas are typically but not always true. For example, the statement "Every bird flies" is not true with the case of ostrich. The same phenomenon occurs with the statement "Every predator of a bird is a bird". The lion is the predator of an ostrich but it is not a bird. The statement "Every prey has a predator" is also not true when predators are extinct. Therefore, a world failing to satisfy even a single formula would not be possible. There could be no possible world satisfying all formulas.

Markov Logic's intuition is to handle uncertainty. We can relax the hard constraint assumption on satisfying all formulas. A possible world not satisfying a certain formula will simply be less likely. The more formula a possible world satisfies, the more likely it is. Each formula can have a weight indicating how strong a constraint it should be for possible worlds. Higher weight indicates higher probability of a world satisfying the formula.

2.4.1.2. Markov Logic's definition

A Markov Logic Network (MLN) is a set of pairs (F, w) where F is a formula in first-order logic and w is a real number. Together with a set of constants, it defines a Markov network with one node for each grounding of each predicate in the MLN, one feature for each grounding of each formula F in the MLN, with the corresponding weight w [16]

$w1$	$\forall x\ Bird(x) \Rightarrow Flies(x)$
$w2$	$\forall x, y\ (Predates(x, y) \wedge Bird(y) \Rightarrow Bird(x))$

- Two constants: Sparrow, Eagle

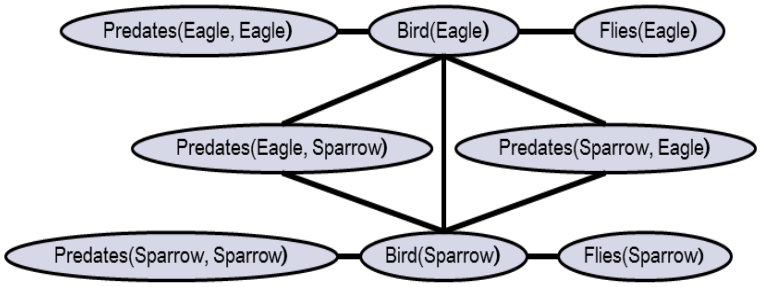


Figure 2.3 Markov Logic Network example

2.4.1.3. Markov Logic's joint probability

A ground MLN specifies a joint probability distribution over possible worlds (i.e. truth value assignments to all ground atoms). The probability of a possible world x is

$$p(x) = \frac{1}{Z} \exp\left(\sum_{i=1}^{F} w_i n_i(x) \right)$$

where: The sum ranges over formulas in the MLN (i.e. clique templates in the Markov Network)

2.4.2. Stanford Typed Dependency Parser

The Stanford typed dependencies representation was designed to provide a simple description of the grammatical relationships in a sentence that can easily be understood and effectively used by people without linguistic expertise who want to extract textual relations. In particular, rather than the phrase structure representations that have long dominated in the computational linguistic community, it represents all sentence relationships uniformly as typed dependency relations. That is, as triples of a relation between pairs of words, such as the "subject of distributes is Bell". Their experience is that this simple, uniform representation is quite accessible to non-linguists thinking about tasks involving information extraction from text and is quite effective in relation extraction applications.

Here is an example sentence:

Bell, based in Los Angeles, makes and distributes electronic, computer and building

products.

For this sentence, the Stanford Dependencies (SD) representation is:

nsubj(makes-8, Bell-1)
nsubj(distributes-10, Bell-1)
partmod(Bell-1, based-3)
nn(Angeles-6, Los-5)
prep in(based-3, Angeles-6)
root(ROOT-0, makes-8)
conj and(makes-8, distributes-10)
amod(products-16, electronic-11)
conj and(electronic-11, computer-13)
amod(products-16, computer-13)
conj and(electronic-11, building-15)
amod(products-16, building-15)
dobj(makes-8, products-16)
dobj(distributes-10, products-16)

These dependencies map straightforwardly onto a directed graph representation, in which words in the sentence are nodes in the graph and grammatical relations are edge labels. Figure 2.4 gives the graph representation for the example sentence above.

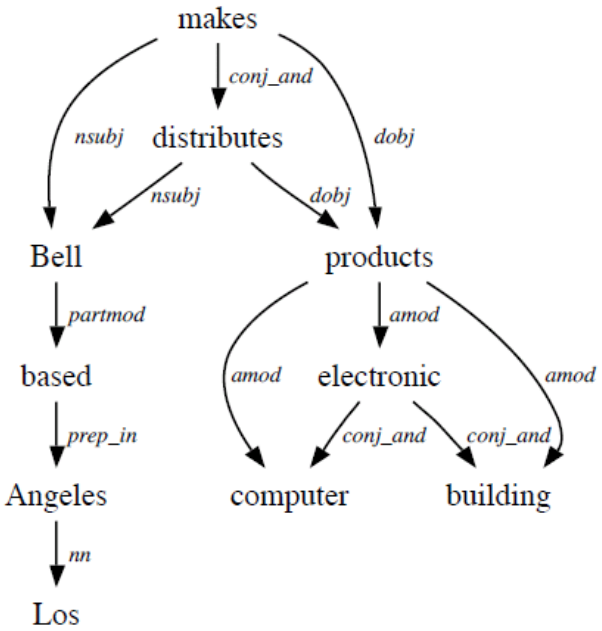


Figure 2.4 Graph representation of STD

2.4.3. Unsupervised semantic parsing (USP)

In Unsupervised semantic parsing (USP) [17], Hoifung Poon and Pedro Domingos presented the first unsupervised approach to the problem of learning a semantic parser, using Markov logic. Their USP system transforms dependency trees into quasi-logical forms, recursively induces lambda forms from these, and clusters them to abstract away syntactic variations of the same meaning. The MAP semantic parse of a sentence is obtained by recursively assigning its parts to lambda-form clusters and composing them. The results of their methods are clusters of syntactic variation of the same meaning phrases, Markov Logic Network of clusters and MAP semantic parse.

Experimental results

They evaluate on an end task: Question answering by applying USP to extract knowledge from text and answer questions. The gold logical forms are not predefined. They evaluate their method by counting number of answers and accuracy and have following result:

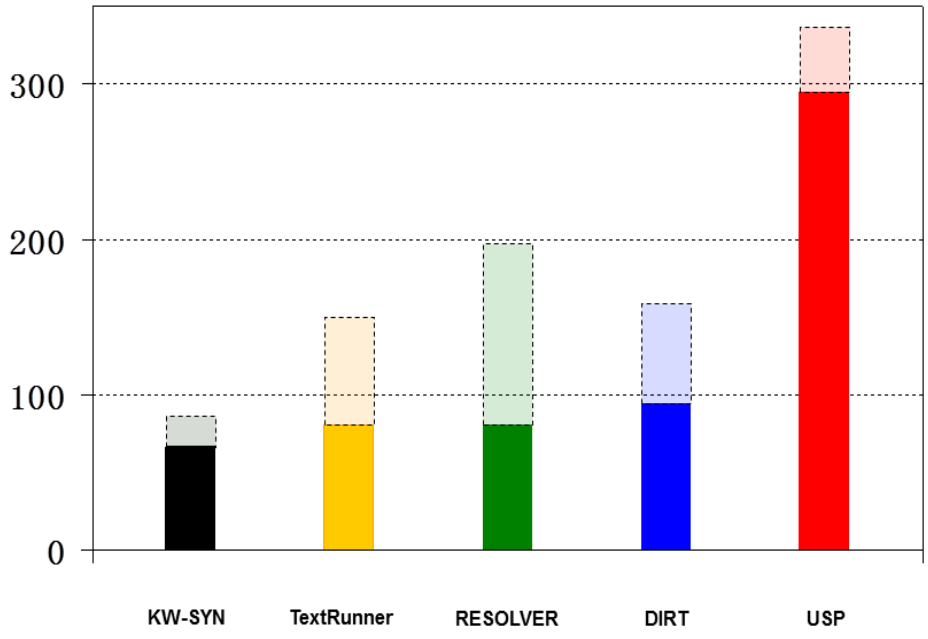


Figure 2.5 The number of correct answers in USP evaluation experiment

Their method gets three times as many correct answers as second best and the highest accuracy: 88%

2.5. CaboCha (Yet Another Japanese Dependency Structure Analyzer)

CaboCha is a Japanese dependency parser based on Support Vector Machines. To our best knowledge, Cabocha is the most accurate statistical Japanese dependency parser system (it reaches to 89.29% accuracy). In addition, because it is a Cascaded Chunking Model using a non-backtracking definitive analysis algorithm, the parse can be performed relatively efficient.

Features

Based on Support Vector Machines (SVMs) method, it is a high-performance dependency analyzer.

Use a fast SVM method which is called PKE (presented at ACL 2003).

Use a definitive analysis algorithm (Cascaded Chunking Model) which is a high-efficiency analysis one.

Consider dependency relations as "dynamic features" that significantly contribute to improve the accuracy.

Use definition of IREX so that it can analysis specific expression.

Flexible input format. All types of raw sentences, as well as morphological parsed data, segment pre-separated data or given data which is related in part can be parsed.

The feature which uses the identification of the dependency can be redefined by the users.

Only if the data is provided, the learning can be performed by the user.

Use Double-Array which contains the high-speed structure Trie in the internal dictionary,

Provision of C / C + + / Perl / Ruby library.

Chapter 3

Method

Our model is built by the following way: the common nouns are extracted to become the classes. These classes have attributes, methods and relations extracted correspondingly from the dependencies of legal terms like in table 1. These classes then are represented textually.

POS Tagging	**UML class component**
Noun	Class
Verb	Method of Class
Modifier Noun, Verb Adjective	Attribute of Class
Relation between Nouns	Relation between Classes

Table 3.1: Mapping from POS tagging to UML class components.

There are 3 steps to transform Japanese legal texts to UML class components: Cabocha parsing, dependency extraction and UML class transformation. Our system is presented in Figure 3.1

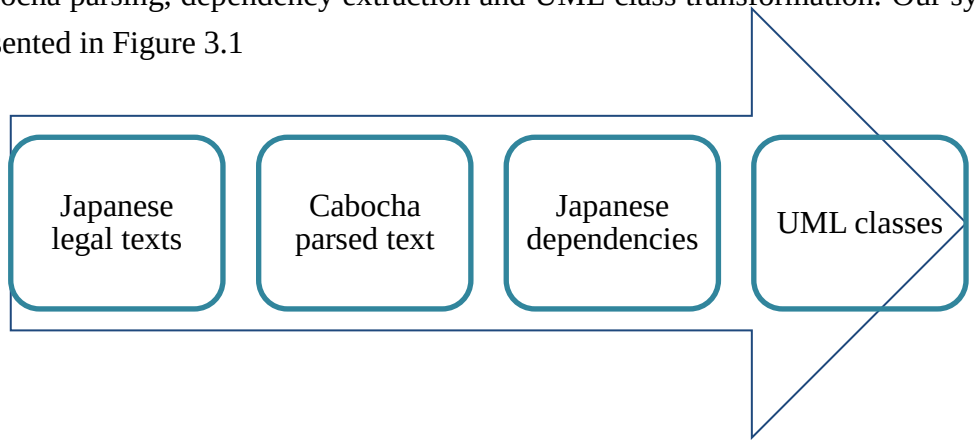


Figure 3.1 Japanese law conceptual model extraction system

3.1. Data processing

The legal text is processed by the following way:

Each line ends with 。is a sentence.

The numbers and the spaces in the first position of the sentences are removed.

For example, the sentence

２　前項の財政均衡期間（第十六条の二第一項において「財政均衡期間」という。）は、財政の現況及び見通しが作成される年以降おおむね百年間とする。

is processed to:

前項の財政均衡期間（第十六条の二第一項において「財政均衡期間」という。）は、財政の現況及び見通しが作成される年以降おおむね百年間とする。

The sub parts of the act are combined to become a sentence, separated by a comma 、by the following way: All continuous lines without 。ending is combined to the above nearest sentence ends with 。.

For example,

第五条　この法律において、「被用者年金各法」とは、次の各号に掲げる法律をいう。

一　厚生年金保険法（昭和二十九年法律第百十五号）

二　国家公務員共済組合法（昭和三十三年法律第百二十八号）

三　地方公務員等共済組合法（昭和三十七年法律第百五十二号）

四　私立学校教職員共済法

Becomes

第五条　この法律において、「被用者年金各法」とは、次の各号に掲げる法律をいう一　厚生年金保険法（昭和二十九年法律第百十五号）、二　国家公務員共済組合法（昭和三十三年法律第百二十八号）、三　地方公務員等共済組合法（昭和三十七年法律第百五十二号）、四　私立学校教職員共済法

3.2. Cabocha parsing

We started the experiment from parsing legal text using Cabocha [8]. CaboCha is a Japanese dependency parser based on Support Vector Machines. To our best knowledge, Cabocha is the most accurate statistical Japanese dependency parser system (it reaches to 89.29% accuracy). In addition, it is a Cascaded Chunking Model using a non-backtracking definitive analysis algorithm so that the parse can be performed relatively efficient.

Here is an example of Cabocha parse for

労働保険の保険料の徴収等に関する法律

<sentence>
<chunk id="0" link="1" rel="D" score="1.05433" head="1" func="2">
<tok id="0" read="ロウドウ" base="労働" pos="名詞-サ変接続" ctype="" cform="" ne="O">労働</tok>
<tok id="1" read="ホケン" base="保険" pos="名詞-一般" ctype="" cform="" ne="O">保険</tok>
<tok id="2" read="ノ" base="の" pos="助詞-連体化" ctype="" cform="" ne="O">の</tok>
</chunk>
<chunk id="1" link="2" rel="D" score="1.80505" head="4" func="5">
<tok id="3" read="ホケン" base="保険" pos="名詞-一般" ctype="" cform="" ne="O">保険</tok>
<tok id="4" read="リョウ" base="料" pos="名詞-接尾-一般" ctype="" cform="" ne="O">料</tok>
<tok id="5" read="ノ" base="の" pos="助詞-連体化" ctype="" cform="" ne="O">の</tok>
</chunk>
<chunk id="2" link="3" rel="D" score="0" head="7" func="8">
<tok id="6" read="チョウシュウ" base="徴収" pos="名詞-サ変接続" ctype="" cform="" ne="O">徴収</tok>
<tok id="7" read="トウ" base="等" pos="名詞-接尾-一般" ctype="" cform="" ne="O">等</tok>
<tok id="8" read="ニカンスル" base="に関する" pos="助詞-格助詞-連語" ctype="" cform="" ne="O">に関する</tok>
</chunk>

<chunk id="3" link="-1" rel="O" score="0" head="9" func="9">
<tok id="9" read="ホウリツ" base="法律" pos="名詞-一般" ctype="" cform="" ne="O">法律</tok>
</chunk>
</sentence>

We will use token information: token base, pos, ctype… in the next phase to extract the dependencies between tokens.

On implementation, we store a law text in 2 dimensions - dynamic array inputArray[i][j]. This array keeps the useful information of tokens extracted from Cabocha parsing. The useful information of each line in Cabocha parsing is stored in inputArray[i]. Each token line is transformed to a row started by sentenceID, chunkID, tokenID and followed by token base, token pos, ctype, token and ctype if having. Following is the input array of above Cabocha parsing text.

001 労働名詞-サ変接続労働

002 保険名詞-一般保険

003 の助詞-連体化の

014 保険名詞-一般保険

015 料名詞-接尾-一般料

016 の助詞-連体化の

027 徴収名詞-サ変接続徴収

028 等名詞-接尾-一般等

029 に関する助詞-格助詞-連語に関する

0310 法律名詞-一般法律

3.3. Dependency extraction

According to the representation of Stanford parser [8], we extracted five types of dependencies: dep(noun1_pos, noun2_pos, verb), dep(noun_pos, verb), no(noun1_pos, noun2), no(noun_pos, verb) and no(noun_pos, adjective).

Nouns, verbs and adjectives in the sentences are extracted in phrases. For example, “労働保険”, “開催された”, “必要な” phrases are extracted. The POS

tagging of the first token in the noun phrase is extracted as the POS tagging of this phrase. The POS tagging of the verb/adjective token in the verb/adjective phrase is extracted as the POS tagging of this phrase.

The dep dependency shows the relation between nouns through a verb. In the sentence which has two or three noun components: subject, direct object or indirect object, we extract the dependency dep(noun1, noun2, verb) in which the nouns are the corresponding components. In the sentence which only has only one noun component (either subject or direct object or indirect object), we extract the dependency dep(noun, verb).

The no dependency is the relation between a noun and its characteristic. The characteristic may be a modifier noun, verb or adjective.

3.3.1. dep(noun1_pos, noun2_pos, verb)

This dependency is extracted by the following way: the nouns are either subject, direct object or indirect object and thus in general, we have 3 dependencies: dep(subj_pos, dobj_pos, verb), dep(subj_pos, iobj_pos, verb) and dep(dobj_pos, iobj_pos, verb).

A noun phrase followed by は/が/とは is defined as a subject.

A noun phrase followed by を is defined as a direct object.

A noun phrase followed by で/から/に/について/により/において is defined as an indirect object.

In the case of there are many continuous noun phrases separated by a comma or some conjunction words (for example, また/及び/と…), all nouns are extracted.

We used pos of the first word for compound nouns, considering the use of the word meaning.

For example, from the sentence

事業者は労働者に給料を払う。

We extract three dependencies:

dep(事業者_名詞-一般, 労働者_名詞-サ変接続, 払う)

dep(事業者_名詞-一般, 給料_名詞-一般, 払う)

dep(労働者_名詞-サ変接続, 給料_名詞-一般, 払う)

3.3.2. dep(noun_pos, verb)

Depend on the type of the noun, we extract one of three dependencies: dep(subj_pos, verb), dep(dobj_pos, verb) and dep(iobj_pos, verb).

For example, from the sentence

労働者は働く。

We have the dependency:

dep(労働者_名詞-サ変接続, 働く)

3.3.3. no(noun1_pos, noun2)

This dependency is extracted from the following structures:

noun1 は/が/とは… noun2 です/である”/をいう

noun1 <space>…noun2 をいう

For example, from the sentence:

事業者　事業を行う者で、労働者を使用するものをいう。

We extract the dependency no(事業者_名詞-一般, もの).

This dependency also can be extracted from “noun1 の noun2” structure. If before and after の、there are many continuous nouns separated by a comma or conjunction words, all the pairs are extracted.

For example, from the structure:

工場又は事務所の輸送、建築物、機械器具

We can extract these following dependencies:

no(工場_名詞-一般, 輸送), no(工場_名詞-一般, 建築物)

no(工場_名詞-一般, 機械器具)

no(事務所_名詞-一般, 輸送)

no(事務所_名詞-一般, 建築物)

no(事務所_名詞-一般, 機械器具).

3.3.4. no(noun_pos, verb)

This dependency is extracted from structure “verb noun”. The verb is extracted in full form of core form along with its tense.

For example, from phrase

働いた労働者

We extract the dependency no (労働者_名詞-サ変接続, 働いた)

3.3.5. no(noun_pos, adjective)

This dependency is extracted from following structures

adjective noun

noun は/が adjective です/である

For example, from phrase

必要な事項

we extract dependency no(事項_名詞-一般, 必要な)

3.4. UML transformation

In this step, the extracted dependencies in above phases are transformed into UML classes. All noun phrases go with the pos is ”名詞-一般” or ”名詞-サ変接続” are transformed to classes. The dep dependencies are transformed to methods and relations of the class. The no dependencies are transformed to attributes of classes. The detail transformations are presented in Table 3.2.

Dependency	Class	Attribute	Method	Relation
dep(noun1, noun2, verb)	noun1, noun2		verb	noun1 verb noun2
dep(noun, verb)	noun		verb	noun verb noun
no(noun1, noun2)	noun1, noun2	noun2		
no(noun, verb)	noun	verb		
no(noun, adjective)	noun	adjective		

Table 3.2 UML transformation

3.5. Transformation result

We run our model in 108 Japanese laws [20]. A law is parsed using Cabocha parser and then Japanese Dependencies are extracted. At last, these dependencies are transformed to UML classes. The classes go along with their attributes, methods and relations are represented textually. This representation is the conceptual structure we would like to extract.

Below is some sample of extracted dependencies from 国民年金法.

no(実施_名詞-サ変接続, 円滑な_名詞-形容動詞語幹)
dep(dobj_実施_名詞-サ変接続, verb_図る)

no(百一条_名詞-数, 二_名詞-数)

no(国民年金制度_名詞-一般, 目的_名詞-一般)
dep(subj_国民年金制度_名詞-一般, iobj_二項_名詞-数, verb_規定する)

dep(subj_国民年金制度_名詞-一般, dobj_こと_名詞-非自立-一般, verb_連帯する)

Table 3.3 is the statistical result of 国民年金法

Index	Count
The number of raw sentences	3.055
The number of processed sentences	2.185
The number of Japanese dependencies	32.723
The number of classes	2.069
The number of Attributes	3.506
The number of Methods	3.528
The number of Relations	8.646

Table 3.3 Statistical result of 国民年金法

Below is the sample of extracted UML class of 国民年金法

5. Class Name: 積立金

Attribute:
- 運用
- 管理
- 積立て
- 額
- 規定する
- 係る

Method:
- なる
- 留意し
- 行う
- 寄託する
- 預託する
- 積み立てなければならない

Relation:

なる with class 一部
行う with class 効率的
寄託する with class 運用
預託する with class 財政融資資金
積み立てなければならない with class 基金

Chapter 4

Evaluation

We evaluated our model internally and externally.

Internal evaluation was taken to evaluate the accuracy of our result itself.

External evaluation was taken by comparing the conceptual model built by human with our result.

The most important advantage of our model is that our model can capture a huge part of common nouns with their characteristics and transform them to classes go along with attributes with the high accuracy while human work frequently fails to extract some terms and their characteristics or relations. We even can extract more legal terms with their characteristics and relations if we relax the dependency extraction by adding some new extraction rules.

We applied external evaluation to 国民年金法 and internal evaluation to 108 Japanese laws [] and get the good result as well.

4.1. Internal evaluation

Because our dependency based on the basic structures of Japanese grammar, the terms with their characteristics and dependencies are extracted with relatively high accuracy.

However, the extracted terms and their characteristics and relations include useless words (for example ため，もの，こと…),

There are also some relations which are meaningless, for example, from the phrase:

第百五条第三項の規定による届出

The extracted dependency no(届出，よる) actually is not very meaningful.

Besides, there are some relations still are not captured. For example, relation in sentences included parentheses like following:

一　厚生年金保険法　（昭和二十九年法律第百十五号）

The other type of relations which still not be captured is the relation between the acts and their sub parts in which the sub parts end with 。. For example, the relations between legal terms in the first sentence and the legal terms in the next three sentences in the following example are still not captured.

第四十条　遺族基礎年金の受給権は、受給権者が次の各号のいずれかに該当するに至つたときは、消滅する。

一　死亡したとき。

二　婚姻をしたとき。

三　養子となつたとき（直系血族又は直系姻族の養子となつたときを除く。）。

4.2. External evaluation

We took external evaluation by comparing our result with the human work of national pension law 国民年金法.

From 国民年金法, human extracted legal terms in 7 levels in which legal terms in next level are the characteristics of legal terms in the prior level. While the human work gets 19 legal terms in level 1, 115 legal terms in level 2, 155 legal terms in level 3, 175 legal terms in level 4, 169 legal terms in level 5, 101 legal terms in level 6 and 1 legal term in level 7, in total, 723 legal terms are captured.

Our model capture 2069 nouns as classes with 3506 attributes, 3528 methods and 8646 relations. In which, 436/723 (62.74%) legal terms which are captured in human work are captured in our work.

There are some terms which appeared in human work cannot be captured in our work because they are complicated phrases which actually don't appear in the law (for example, 10 月の前月までの分 or 支給を停止するものとされた年金給付_A20P1 or第九十条の二第一項の規定によりその一部の額につき保険料を納付することを要しないものとされている者). (The experts created these terms by themselves).

Chapter 5

Conclusion and Future Work

5.1. Conclusion

In this study, we proposed the method to extract conceptual model of Japanese laws. Our model uses Cabocha parsed information to extract the dependencies of legal terms and then transform them to UML classes with attributes, methods and relations. We presented the classes textually. The advantage of our research is the ability to capture a huge part of legal terms go along with their characteristics and other related terms. This conceptual model not only is able to be used to understand the law clearly but also to check the correctness and consistence of the law. It is also able to use to develop an information system based on laws. On studying this research, we met some challenges on how to extract Japanese Typed Dependencies enough and correctly and how to represent UML classes effectively. After deeply analyzing, we decided to extract Japanese Typed Dependencies from basis Japanese grammar structures, based on the relations between Subjects, Direct Objects and Indirect Objects of the sentence. The relations between nouns and their modifiers are also captured. According to our knowledge, this is the first Japanese Typed Dependency developed. This tool aims to develop for Japanese legal texts but it can be improved furthermore easily for general Japanese texts. Although there are some relations still not captured and some relations are meaningless, this is a useful tool. To represent UML classes, because there are a huge part of noun phrases in legal text, graphical representation is impossible. Therefore, we represented UML classes textually.

5.2. Future work

In the future work, we will improve current dependency rules and develop some others to better capture relations in Japanese legal texts. Moreover, we will expand this Dependency tool for general Japanese texts. Besides, we also evaluate our model using large data. In the longer plant, we will use this conceptual structure as a basis to develop information system based on laws.

References

[1] T. Katayama, 2005. "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). In IPSJ (Information Processing Society of Japan) Journal, 46(5), pp.515-521.

[2] T.Katayama, 2007. "Legal engineering – an engineering approach to laws in e-society age". In Proceedings of the 1st International Workshop on JURISIN.

[3] John Sowa, 1984. "Conceptual Structures". Information Processing in Mind and Machine. Reading, MA. Addison-Wesley. ISBN 978-0201144727.

[4] Yucong Duan, Christophe Cruz, 2011. "Formalizing semantic of natural language through conceptualization from existence". International Journal of Innovation, Management and Technology (2011) 2 (1), pp. 37-42.

[5] Xiao He, 2007. "A metamodel for the notation of graphical modeling languages". Computer Software and Applications Conference, 2007. COMPSAC 2007. Vol.1. 31st Annual International, Volume 1, 24–27 July 2007, pp 219-224.

[6] Peter Pin-Shan Chen, 1975. "The Entity Relationship Model: Toward a unified view of data". Journal of ACM Transactions on Database Systems (TODS) - Special issue: papers from the international conference on very large data bases. Volume 1 Issue 1, March 1976 September 22–24, 1975, Framingham, MA.

[8] http://www.uml.org

[9] Terry Halpin, 1998. "Object-Role Modeling (ORM/NIAM)". In Handbook on Architectures of Information Systems, edited by P. Bernus, K. Mertins & G. Schmidt, Springer-Verlag, Berlin, 1998, pp. 81-101.

[10] Enrico Motta, 2011. "An Overview of the OCML Modelling Language". In Proceedings of 8th Workshop on Knowledge Engineering Methods & Languages, KEML'98.

[11] Gonzalez-Perez, Cesar; Parcero-Oubiña, César, 2011. "A Conceptual Model for cultural heritage definition and motivation", 39th Annual Conference on Computer Applications and Quantitative Methods in Archaeology, CAA 2011.

[12] http://www.code.google.com/p/cabocha/

[13] http://nlp.stanford.edu/software/lex-parser.shtml

[14] Marie-Catherine de Marne_e, Christopher D. Manning, "Stanford typed dependencies manual". September 2008.

[15] Mathias Kleiner, Patrick Albert, Jean Bézivin, 2009. "Parsing SBVR-Based Controlled Languages". In Model Driven Engineering Languages and Systems.

[16] Imran Sarwar Bajwa, Ali Samad and Shahzad Mumtaz. 2009. "Object Oriented Software Modeling Using NLP Based Knowledge Extraction". European Journal of

Scientific Research, Vol.35 No.1, ISSN 1450-216X, pp. 22-33.

[17] Narayan Debnath et al, 2011. “An ATL Transfo- rmation from Natural Language Requirements Models to Business Models of a MDA Project”. Security Workshop, 2011 11th International Conference on ITS Telecommunications.

[18] Hoifung Poon and Pedro Domingos. 2009. “Unsupervised Semantic Parsing”. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP.

[19] Matt Richardson and Pedro Domingos. 2006. “Markov logic networks”. Machine Learning, Vol.62, Iss.1-2, ISSN: 08856125, pp.107-136.

[20] http://www.japaneselawtranslation.go.jp/

Publication

[1] Pham Anh Cam, Minh Le Nguyen, Akira Shimazu, 2012. Study on extracting conceptual structures from legal texts. Natural Language Processing Conference, Hiroshima, March, 2012. Accepted.

Appendix

The UML classes result of Japanese national pension law 国民年金法