

Automatic Acquisition of Linguistic Patterns for Conceptual Modeling

Nan Zhou, Xiaohua Zhou
Nan.Zhou@cis.drexel.edu, Xiaohua.Zhou@drexel.edu

1. Introduction

1.1 What is Conceptual Modeling

Conceptual modeling aims to capture the knowledge of a domain as perceived by the modeler and represent it in a way that will help further system analysis and design. Object-oriented methodology has been taken as a common practice for software system analysis and design because of its obvious advantages of handling the “software crisis” (Booch, 1994). Object-oriented analysis (OOA) is “a method of analysis that examines requirements from the perspectives of the classes and objects” (Booch, 1994). Object-oriented design (OOD) is a method of design “encompassing the process of OO decomposition and a notation for depicting both logical and physical as well as static and dynamic models of the system” (Booch, 1994).

An object-oriented system decomposes into classes. Class diagram plays an essential role in conceptual modeling. It shows the existence of classes and their relationships in the logical view of a system. Each of the classes has its associated attributes and operations. In object-oriented systems, the notion of class is carried over from analysis to design, implementation, and testing. Thus, finding a set of domain classes is the most important skill in developing an object-oriented system (Song, et al, 2004). However, practices and experiences showed that discovering domain classes is rather difficult and time-consuming for novice analyzers.

1.2 Methods for Conceptual Modeling

Considerable amount of work has been done in the field of conceptual modeling on approaches for identifying classes. Among those approaches noun analysis is the most popular one (Rumbaugh et al, 1991; Booch, 1994). Other methods use class categories (Larman, 2001), case descriptions (Jacobson, 1992), or Class-Responsibilities-Collaborators (CRC) cards (Wirfs-Brock et al, 1990). Recently Song et al (2004) proposes Taxonomic Class Modeling (TCM) methodology that incorporates the noun analysis, class categories, English sentence structure rules, checklists, and other heuristic rules for modeling.

Some systems have been proposed to automate the process of conceptual modeling. For example, SMART (system model acquisition from requirements text) (Dori et al., 2004) is a good attempt to construct a system model in a semi-automatic way from the system’s free text documentation of the requirements. Zisman et al (Zisman et al, 2003) proposed an approach to support automatic generation and maintenance of traceability relations between different types

of software requirements artifacts, which are represented in the eXtensible Markup Language (XML).

Identifying classes and relationships from natural language is not a new topic. Researches have been tried to apply NL parsing and understanding techniques to automatic extraction of models from requirement documents in natural language. Goldin et al. did lexical analysis to find abstractions in unstructured and un-interpreted text (Goldin and Berry, 1997). LIDA (Linguistic Assistant for Domain Analysis), a methodology and prototype tool, was proposed to provide linguistic assistance in the model development process (Overmyer et al, 2001).

1.3 Problem Description

This research paper is a continuation and part of the implementation of the research work *Auto-Generation of Class Diagram from Free-text Functional Specifications and Domain Ontology* we did for INFO 626: Natural Language Processing (Summer 2004), in which we proposed the framework of a system that automates the building of class diagram from unstructured system requirements documents. One of the essential tasks that the framework targets to undertake is using Natural Language Processing (NLP) techniques to detect the classes and their associated attributes as well as the relationships between.

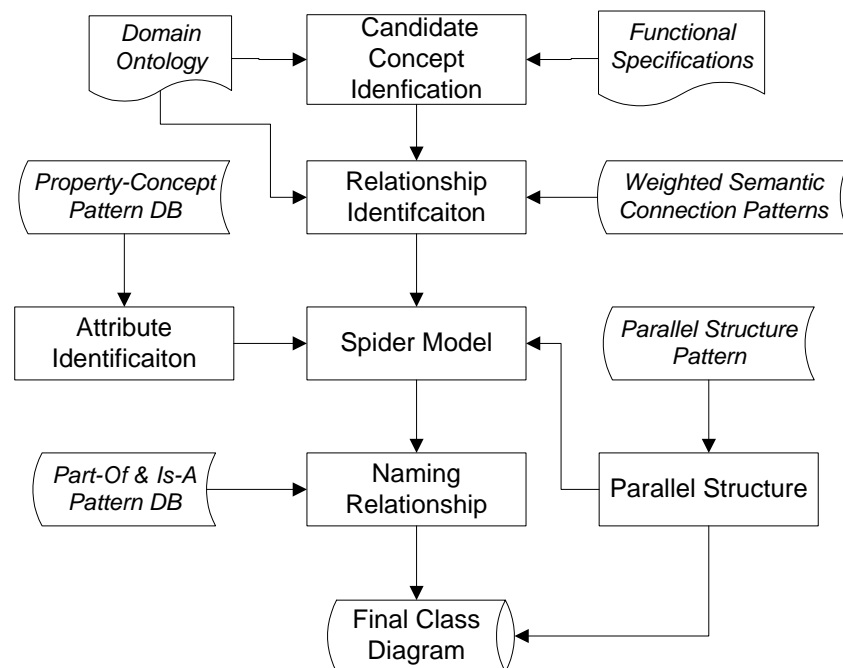


Figure 1. Framework of Class Diagram Auto-generation

As shown in Figure 1, the full implementation of the essential task involves much work including candidate concept identification, relationship identification, attribute identification, naming relationship, etc. Among all these subtasks, some are resolved by linguistic patterns. For example, we use linguistic pattern to distinguish between a class and an attribute.

The use of linguistic patterns in auto-generation of class diagram is summarized as follows:

- Use Property-Concept Pattern to distinguish between classes and attributes.

- Use Part-Of Pattern to detect aggregation relationship between classes.
- Use IS-A Pattern to detect generalization relationship between classes.
- Use Numeric Pattern to tell three types of association relationships, one-one, one-many, and many-many.
- Use Parallel Structure Pattern to find more attributes, more parts, and more child classes.

In the 2004 paper, we leave the creation of linguistic patterns open. That is, it can be either hand-crafted or automatically learned from training examples. Manual creation of patterns is not only time-consuming but also error prone. With this particular concern, **we here propose the use of machine learning to automatically acquire linguistic patterns for conceptual modeling.**

The remainder of the paper is organized as follows. Section 2 reviews the applications of linguistic patterns in conceptual modeling, information extraction, and text mining. Section 3 reviews the various representations of linguistic patterns. Section 4 reviews all existing work on linguistic pattern learning methods. A solution to stated problem is proposed in section 5, which is followed by the conclusion part.

2. Review of Linguistic Pattern Applications

2.1 Conceptual Modeling

The efforts of detecting language patterns for the benefit of conceptual modeling can be traced back to the research work Chen has done in the early 1980s. In *English Sentence Structure and Entity-Relationship Diagrams* (Chen, 1983), Chen studied the correspondence between English sentence structure and ER diagrams, and proposed eleven rule for translation. Hearst proposed a method for the automatic acquisition of the hyponymy lexical relation from unrestricted text and identified a set of lexica-syntactic patterns (Hearst, 1992). Later on, Berland and Charniak extended the work to extracting parts of objects from wholes (e.g. “speedometer” from “car”) (Berland et al, 1999).

2.2 Information Extraction & Text Mining

Pattern-matching is a frequently used approach in information extraction (IE). It can be applied to not only free text and semi-structured text (Califf and Mooney, 1998, 2004; Downey et al., 2004; Freitag, 1998; Soderland, 1999), but also generation of HTML wrappers (Chang and Lui, 2001; Hsu and Dung, 1998; Kushmerick, Weld, and Doorenbos, 1997; Muslea, Minton, and Knoblock, 1999).

The IE systems for free text can use linguistic patterns to extract various types of information ranging from the simplest (e.g. noun phrase) to the most complex (e.g. the sentiment of a sentence). Argamon, Dagan and Krymowski (1998) use shallow natural language patterns to extract Noun Phrase (NP), Verb-Object (VO) relations, and Subject-Verb (SV) relations in English sentences. GATE (A General Architecture for Text Engineering) (Cunningham, 2002)

employs patterns represented by regular expressions (Cunningham, Maynard, and Tablan, 2000) to implement basic natural language processing components such as tokenization, named entity recognition, and coreference resolution (Dimitrov et al., 2004). Various Is-A patterns are applied to annotating instances of concepts in certain ontology (Cimiano, Handschuh and Staab, 2004; Heart, 1992).

Apart from extracting single-slot information, most IE systems can also use complicated patterns constrained by syntactic and semantic conditions to fill multi-slot template (Alani et al., 2003; Huffman, 1996; Krupka, 1995; Soderland et al. 1995). Moreover, IE systems are able to identify domain-specific events such as protein-protein interactions (Hu et al., 2004), biomedical pathways (Hirschman et al., 2002), and quality complaints (Barbuceanu et al., 2003) by pattern matching.

Besides extraction of values, IE systems sometimes are asked to classify the document or a segment of text. For example, a sentiment analyzer (Yi et al., 2003) is designed to distinguish between positive and negative comments; AutoSlog-TS is used to classify each sentence in document as subjective or objective (Riloff, 2003). For such systems, linguistic patterns, especially the occurrence of certain words, are strongly indicative of text classifications.

Text mining, also known as text data mining or knowledge discovery from textual databases, generally refers to the process of extracting interesting and non-trivial patterns or knowledge from unstructured text document (Tan, 1999). Usually, text mining is indispensable of information extraction because current visualization, clustering, statistical, or summarizing methods can't process unstructured text directly so far. Therefore, linguistic patterns are also very important to text mining.

3. Review of Linguistic Pattern Representations

Pattern representation is important in the sense that it's closely tied into how the pattern extraction mechanism may work. Pattern representation has to be realized in a form that will facilitate the extraction process and storing of the results found in a most meaningful way.

Kim & Moldovan (1995) summarize the characteristics of information extraction as follows:

- A small number of even categories leading to many-to-one mappings.
- The types of information are predefined.
- Information can be found anywhere in the sentence.

These characteristics apply to our case of detecting certain linguistic patterns too. Therefore, an efficient representation should be able to map expressions in various forms to one of desired categories, and detect the information carrying words or phrases from anywhere in the sentence (Kim & Moldovan 1995).

The information extraction systems we reviewed use various ways to represent patterns found from the extraction process. (See Table 1: Summary of Extraction Pattern Learning Systems)

AutoSlog (Riloff, 1993) uses concept nodes to represent patterns. Each concept has a conceptual anchor that activates it and a linguistic pattern that guarantees its applicability together with the set of enabling conditions. The conceptual anchor is a triggering word and the enabling conditions are the constraints on the pattern components. Figure 2 illustrates the representation using an example:

CONCEPT NODE:
 Name: target-subject-passive-verb-bombed
 Trigger: bombed
 Variable Slots: (target (*S* 1))
 Constraints: (class phys-target *S*)
 Constant Slots: (type bombing)
 Enabling Conditions: ((passive))

Figure 2: Example of AutoSlog concept node

The extraction patterns generated by **HASTEN** (Krupka 1995) are called *Egraphs*, which can be seen as lists of (SemanticLabel, StructureElement) pairs. The semantic label has a conceptual anchor and other potential items of interest. The structural element associated to the semantic label represents a set of syntactic and semantic constraints. An example is illustrated in Figure 3.

BOMBING:
 TARGET: NP “semantic = physical-object”
 ANCHOR: VG “root = bomb”
 PERPETRATOR: NP “semantic = terrorist-group”

Figure 3: Example of Egraph used by HASTEN

CRYSTAL (Soderland et al. 1995) represents patterns in multi-slot concept nodes that allow both semantic and exact word constraints on any component phrase. For example, if the task is to extract both the target and the name of the perpetrator in the following sentence:

The Parliament building was bombed by Carlos.

The concept node is shown as in Figure 4.

Concept type: BUILDING BOMBING
 SUBJECT: Classes include: <PhysicalTarget>
 Terms include: BUILDING
 Extract: target
 VERB: Root: BOMB
 Mode: passive
 PREPOS-PHRASE: Preposition: BY
 Classes include: <PersonName>
 Extract: perpetrator name

Figure 4: Concept Node in CRYSTAL

The **PALKA** (Parallel Automatic Linguistic Knowledge Acquisition) system (Kim & Moldovan 1995) uses a special form *FP-structure* to represent a pattern. *FP-structure* (Frame-Phrasal pattern structure) is a pair of a meaning frame and a phrasal pattern. The knowledge base is organized as a network of *FP-structures* and a concept hierarchy. A meaning frame is represented by a root, a set of slots, and semantic constraints on fillers. A phrasal pattern is an ordered combination of lexical entries or semantic categories. Each slot of the frame is linked to the corresponding element in the phrasal pattern to combine a phrasal pattern and a meaning frame. Figure 5 illustrates how the structure works using the same example “bombing”.

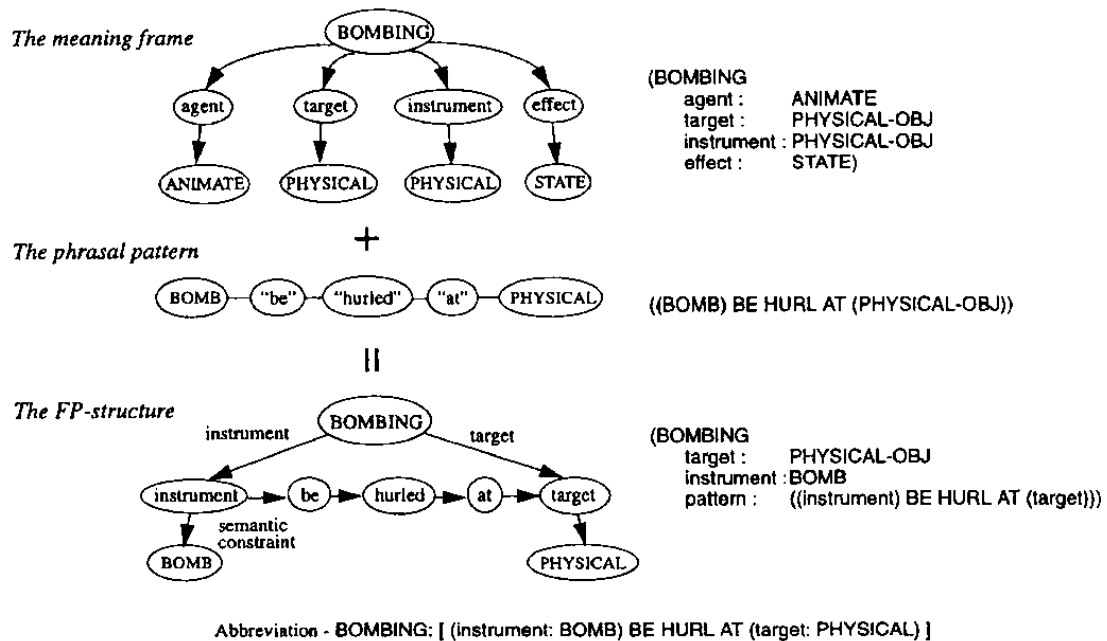


Figure 5: The FP-structure pattern representation

SRV (Freitag, 1998) uses first-order logic extraction patterns based on attribute-value tests and the relational structure of the documents. The pattern illustrated in Figure 6 extracts the names of the companies that were the acquisition targets. The extraction rule is read as: the company name includes a single upper-case word and is followed by a lower-case word.

DOCUMENT-1: ... to purchase 4.5 mln Trilogy shares at ...
DOCUMENT-2: ... acquire another 2.4 mln Roach shares ...
Acquisition:- length(< 2),
some(?A [] capitalized true),
some(?A [next-token] all-lower-case true),
some(?A [right-AN] wn-word 'stock').

Figure 6: An SRV extraction task

RAPIER (Robust Automated Production of Information Extraction Rules) (Califf & Mooney, 1998, 2004) uses pairs of sample documents and filled templates to induce pattern-match rules

that directly extract fillers for the slots in the template. RAPIER's rule representation makes use of limited syntactic and semantic information. Each extraction rule is made up of three patterns: a pre-filler pattern that must match the text immediately preceding the slot-filler; a slot-filler pattern that must match the actual slot-filler, and a post-filler pattern that must match the text immediately following the filler. Each pattern is a sequence of pattern elements, which have two types: pattern items and pattern lists. A pattern item matches exactly one word or symbol from the document that meets the constraints while a pattern list specifies a maximum length N and matches 0 to N words/symbols from the document.

KNOWITALL (Etzioni et al, 2004) aims to automate the process of extracting large collections of facts from the web in an autonomous, domain-independent, and scalable manner. It instantiates a set of extraction rules for each class and relation from a set of generic domain-independent templates. The following is a sample of the syntactic patterns that KNOWITALL uses for rule templates:

NP1 f“,”g “such as” NPList2
 NP1 f“,”g“and other” NP2
 NP1 f“,”g“including” NPList2
 NP1 “is a” NP2
 NP1 “is the” NP2 “of” NP3
 “the” NP1 “of” NP2 “is” NP3

SPIE (Scalable and Portable Information Extraction) method (Hu et al, 2004) uses tuple to represent pattern. The following is an excerpt of the definition of a pattern in SPIE:

*A pattern is a 5-tuples $\langle prefix, entity_tag1, infix, entity_tag2, suffix \rangle$, where *prefix*, *infix*, and *suffix* are vectors associating weights with terms *entity_tag1* and *entity_tag2*. *Prefix* is the part of sentence before *entity_tag1*, *infix* is the part of sentence between *entity_tag1* and *entity_tag2* and *suffix* is the part of sentence after *entity_tag2*.*

Below is some sample extraction patterns generated from MedLine for protein-protein interactions.

{“”, <Protein>, “interacts with”, <Protein>, “”}
 {“ ”, <Protein>, “binds to”, <Protein>, “”}
 {“Bind of ”, <Protein>, “to”, <Protein>, “”}
 { “Complex of ”, <Protein>, “and ”, <Protein>, “”}

In our 2004 **Automation paper**, we constructed a Meta structure for linguistic patterns using the following 7-tuple structure, given a pair of concepts, to determine if the concept pair is property-concept relationship.

$\langle first, second, first\text{-}role, first\text{-}prep, second\text{-}role, second\text{-}prep, verb \rangle$

Where

first: the first concept is property or concept

second: the second concept is property or concept
first-role: the role the first concept serves in the sentence
first-prep: the preposition before the first concept if any
second-role: the role the second concept serves in the sentence
second-prep: the preposition before the second concept if any
verb: the verb used when the two concepts are subject and object respectively

The following 4 examples illustrated the linguistic patterns. Since prepositions are limited and very few verbs indicate master-subordinate relationship, the patterns are limited.

Example pattern 1: {concept, property, pre-noun modifier, null, modified noun, null, null }

Any overdue fee is added up to *customer's outstanding balance*.

Example pattern 2: {property, concept, noun, null, post pp, for, null }

The *bar code ID* for each *item* is entered and video information from inventory is displayed.

Example pattern 3: {property, concept, object, null, pp, in, null }

System determines the rental price and due date and displays *title*, *due date*, and *price* in the *transaction*.

Example pattern 4: {property, concept, object, null, pp, for, null }

A store employee creates an *account* for a new *customer* by inputting customer information.

4. Review of Linguistic Pattern Learning Methods

Pattern-based approach has been proved effective for applications of message understanding such as information extraction, text mining, and conceptual modeling. However, it is difficult to construct a large number of domain-specific patterns. Hand-creation of patterns is not only time consuming, but also error prone. Out of this motivation, researchers have built many systems that can automatically acquire patterns from text using supervised machine learning algorithms. Using the criterion of (Muslea, 1999) and (Soderland, 1998), we classify pattern learning systems into three groups for discussion. They are: (1) for free text, (2) for semi-structured text, and (3) for wrapper induction.

4.1 Patterns for Free Text

AutoSlog

AutoSlog (Riloff, 1993) comes up with extraction patterns called concepts or concept nodes. Each concept has a conceptual anchor that activates the linguistic pattern. AutoSlog uses 13 heuristics rules to learn linguistic patterns from training examples. For example, “<subject> passive-verb” is one heuristics rule, where passive-verb is a placeholder and it will be replaced with specific words after learning. Therefore, “<victim > was murdered” may be a learned linguistic pattern. However, the text must be semantically tagged before AutoSlog can apply heuristics to linguistic pattern learning.

Riloff (1996) came up with a new version called AutoSlog-TS that does not require of the text to be tagged. AutoSlog-TS will produce thousands of patterns from training corpus and use the relevance of the pattern to the domain to rank all extracted patterns. The motivation behind the relevance measure is that the domain specific expressions will appear substantially more often in relevant texts than irrelevant texts.

Riloff (2003) further used AutoSlog-TS to learn extraction patterns for subjective expressions. He adopted a bootstrapping process (Jones, McCallum, Nigam, and Riloff, 1999) to iteratively learn the extraction patterns.

SCT

SCT (Semantic Classification Tree) (Kuhn and Mori, 1995) is a general approach to natural language understanding. It is similar to decision trees where each leaf node stands for a certain predefined class. Each node of SCT represents a regular expression with words in training sentences as tokens. Therefore, the learning of patterns is equivalent to the building of SCT from training examples.

PALKA

PALKA (Kim and Moldovan, 1995) system acquires linguistic patterns represented by FP-structure, a pair of a meaning frame and a phrasal pattern. PALKA is suited to learn syntactic patterns for events, for example bombing frame is the relationship of bombing instrument and target. The learning procedure is straightforward. Some heuristics rules are applied to establishment of FP structures. Then learned patterns will be generalized or specialized by taking both positive and negative examples into account.

HASTEN

Hasten (Krupka, 1995) uses a K-nearest neighbor approach with a set of hand-picked instances as exemplars. Each exemplar represents a positive example in the training corpus. An exemplar consists of elements each of which is a token or phrase in the sentence along with constraints such as semantic class, verb root, verb voice and exact word. In training phase, a weight for each sentence will be calculated to indicate the reliability of the sentence. In testing phase, a new sentence is compared to all exemplars and the k-nearest cases will be found.

CRYSTAL

CRYSTAL (Soderland et al. 1995) derives from a training corpus a domain-specific dictionary of concept nodes definitions that allow syntactic and semantic constraints on the phrase of interest. CRYSTAL will create an initial dictionary from annotated instances. Then it employs an induction algorithm to relax the constraints on each CN definitions. Semantic constraints are relaxed by moving up the semantic hierarchy or dropping the constraints. Exact word constraints are relaxed by dropping all but a subsequence of the words or dropping the constraint.

LEIP

LEIP (Huffman, 1996) uses heuristics in a manner similar to AutoSlog to learn multi-slot extraction rules. But LEIP can only extract multi-slot rules because it finds context for a slot only in terms of its syntactic relationship to other slots. LIEP rules are induced from positive training examples only. Thus, it may overgeneralize the rules.

MBSL

Argamon, Dagan and Krymolowski (1998) propose a memory-based approach to learning shallow natural language patterns for the extraction of Noun Phrase (NP), Verb-Object (VO) relations, and Subject-Verb (SV) relations in English sentences. The extracted pattern is a sequence of part of speech tags.

RHB⁺

RHB⁺ (Sasaki and Matsuo, 2000) is type-oriented inductive logic program system that can induce rules in the form of logic program with (semantic) type from positive training examples. In both training phase and testing phase, the text must be semantically annotated.

CBPLA

The basis of CBPLA (Clustering-based Pattern Learning Approach) is that examples (instances) of the same pattern should be similar to each other (Zheng, Wang and Li, 2003). The CBPLA system uses single link hierarchical clustering algorithm to group all pattern examples based on a similarity measure using common subsequence, and then merges examples in the same cluster to produce final generalized patterns.

SPIE

SPIE (Hu et al., 2004) is not a general extraction pattern learning system. It uses a bootstrapping process to automatically learn patterns in the form of 5-tuple for extraction of protein-protein interactions from PubMed. In order to learn patterns from sentences, a sentence alignment method is used to group similar patterns together and then to learn each group separately for generalized patterns.

4.2 Patterns for Semi-structured Text**RAPIER**

The RAPIER system (Califf and Mooney, 1998, 2004) learns single-slot extraction patterns that make use of limited syntactic information (e.g. part of speech of each word in text) and semantic information (e.g. hypernym links from WordNet). RAPIER is inspired by inductive logic program (ILP) methods, and primarily consists of a specific to general (bottom-up) relational learning.

SRV

SRV (Freitag, 1998) induces extraction patterns represented by first-order logic. Differing from other learning systems for Information Extraction, SRV requires and learns over an

explicitly predefined set of features including simple features (e.g. character type) and relational features (e.g. next_token and subject_verb). The system conducts top-down search and adds predicates (the assertion of certain feature) greedily attempting to “cover” as many positive, and as few negative examples as possible.

WHISK

WHISK (Soderland, 1999) generates multi-slots extraction patterns (rules) for semi-structured or free text. The pattern can impose semantic class, syntactic constraints, and delimiter tags on the extracted phrases, and be written in the way similar to but different from regular expressions. WHISK induces rules top-down typically beginning with an “empty” rule that covers all instances, and then adding terms to the rule one at a time until the Laplacian (error rate) on all instances can’t be optimized any more. Each word, number, punctuation, or HTML tag in the instances are considered a term. If a word is annotated as a semantic class, the semantic class can either be considered a term. WHISK does a form of hill climbing and therefore can’t guarantee that the rules it grow are optimal, where optimal is defined as the lowest Laplacian expected error on the training corpus.

PL

PL (Pattern Learner) (Downey et al., 2004) uses bootstrapping process to automatically learn patterns represented by sequential contextual words for web document annotation (identifying instances of concepts in certain ontology). Since thousands of patterns will come up, two heuristics are applied to pattern selection.

4.3 Patterns for Wrappers

WIEN

WIEN (Kushmerick, Weld, and Doorenbos, 1997) is the first wrapper induction system. The system looks for a common suffix and a common prefix as delimiters for each slot. The process of finding delimiters is similar to WHISK’s step of anchoring each slot with terms immediately preceding and following the extracted phrase.

SoftMealy

SoftMealy (Hsu and Dung, 1998) is a wrapper induction algorithm that acquires extraction rules in form of finite-state transducer. The extracted rules are useful for HTML documents containing more than one formatting conventions or various attribute permutations. The generalization algorithm induces contextual rules by taxonomy tree climbing.

STALKER

STALKER (Muslea, Minton, and Knoblock, 1999) is a wrapper induction system that performs hierarchical information extraction. It uses Embedded Catalog Tree (ECT) formalism to describe the hierarchical organization of the documents. For a given ECT, STALKERS generates one extraction rule for each node in the tree, together with an additional rule for each LIST node.

IEPAD

IEPAD (Chang and Lui, 2001) generates repetitive patterns in the form of sequential HTML tags (the text as a whole between two HTML tags is viewed as special call Text). The system learning patterns by construction of PAT tree over all training examples.

System Name	Input Text	Pattern Representation	Multi-slot	Learning Algorithm
AutoSlog	Free	Concept Node (CN)	N	Heuristic Rules
SCT	Free	Regular expression	N	SCT Building
PALKA	Free	FP Structure	N	Heuristic Rules
HASTEN	Free	Egraph	Y	k-nearest neighbor approach
CRYSTAL	Free	Concept Node (CN)	Y	Induction Algorithm
LEIP	Free		Y	Heuristic Rules
MBSL	Free	N-grams	N	Memory-based learning
RHB ⁺	Free	Logic program with type	N	Type-oriented ILP
CBPLA	Free		N	Clustering
SPIE	Free	Tuple	N	Clustering
RAPIER	Semi	Limited syntactic and semantic constraints	N	Bottom-up Relational Learning
SRV	Semi	First-order logic	N	Top-down Relational Learning
WHISK	Semi, Free	Similar to regular expression	Y	Top-down Heuristic Search
PL	Semi	Sequential context words	N	Heuristic rules
WIEN	Wrapper	Suffix and Prefix	Y	Induction Search
SoftMealy	Wrapper	Finite-state transducer	Y	Induction
STALKER	Wrapper	Finite Automata	N	Sequential Cover Algorithm
IEPAD	Wrapper	Sequential HTML tags	Y	PAT Building

Table 1: Summary of Extraction Pattern Learning Systems.

5. Proposed Solution

The goal of our task is to predict the relationship of two concepts in text using linguistic patterns. The concept pair can either have no relationship, or one of the following relationships: concept-property, generalization, aggregation, or association that can be further classified into one of the three sub-relationships, namely, one-one, one-many, or many-many. Thus, our task is text classification rather than attribute-value extraction. Moreover, like the problem of protein-protein interaction, the problem we are addressing is also very specific. According to the above characteristics of the problem, we propose the solution below.

5.1 Representation of Patterns

Inspired by AutoSlog, PALKA and LEIP that extract information in a very specific domain, we propose heuristic rules (templates) as the first choice for pattern representation. Because our task is very specific, it is possible for us to come up with templates using our knowledge. The placeholder of each template will be replaced with specific words after learning. More specifically, we can use either multiple templates as AutoSlog does, or one very flexible template.

The relationship of two concepts depends on both syntactic and semantic constraints. The syntactic constraints include the constituent (i.e. subject, object, prepositional phrase, etc.) the concepts appear, the verb of the clause, the preposition of the prepositional phrases preceding or following the concepts, etc. The semantic constraints include the semantic type of suffix and prefix of the concepts. Naturally, we think of first-order logic as the second choice for pattern representation, and correspondingly use top-down relational learning to induce the patterns as SRV (Freitag, 1998) does.

Last, our problem, the relationship of two concepts is similar to protein-protein interaction (Hu et al., 2004). Another possible way to represent the patterns is a 5-tuple <prefix, concept1, infix, concept2, suffix>, where prefix is the part of sentence before concept1, infix is the part of sentence between concept1 and concept2, and suffix is the part of sentence after concept2. Manually examining the data, we find that the expression of concepts relationship is more diversified and more flexible than those of protein-protein interaction. Therefore, the coverage of the learned patterns in our problem may be lower than that of protein-protein interaction.

5.2 Learning Methods

As stated in the preceding subsection, we propose three ways to represent expected patterns. They are heuristic template, first-order logic, and 5-tuple respectively. Correspondingly, we suggest three supervised learning methods to generate linguistic patterns from training examples.

For heuristic templates, we use straightforward template filling approach to find initial patterns and then apply some heuristic rules to pattern selection. For first-order logic representation, we use top-down search, a greedy cover algorithm, to produce rules (patterns). For 5-tuple representation, we adopt the same approach to learn patterns as stated in (Hu et al., 2004).

5.3 Experiments

We have manually collected 10 real-word problem statements and their class diagrams for system analysis. Because each statement contains more than 100 concept-concept (class-class or class-attribute) pairs, the total number of positive examples will be over 1000 that is considered sufficient for supervised training. We will conduct 5-folder cross-validation on the

data set and use two frequently used measures, precision and recall, to evaluate the quality of the learned patterns.

In both training phase and testing phase, the texts need to be annotated prior to the experiment. Because the class names and attribute names in the class diagram will not necessarily appear as exact words or phrases in the text, the annotating process is not self-evident. Usually, hand-annotation achieves high accuracy and coverage, but it costs more efforts and time and consequently refrain us from performing test on larger data set. Automatic annotation on the other hand, is more efficient, but may achieve relatively lower accuracy and coverage. In the experiment, we will try both two complementary annotating methods and compare their results.

6. Conclusions

Auto-generation of class diagram from unstructured user requirement documents is a challenging problem in the field of conceptual modeling. However, we are not going to address all problems in this paper. Usually linguistic patterns are used to distinguish between a class and an attribute, and to classify the variety of inter-class relationships. But hand-creation of patterns is not only time-consuming, but also error prone. Thus, we propose a solution to automatic acquisition of linguistic patterns from training examples.

Before proposing the solution, we review the past works in the follow three areas: (1) the application of linguistic patterns in conceptual modeling, information extraction, and text mining, (2) the representation of linguistic patterns, and (3) the learning methods for linguistic patterns. Then we select three suitable representations and corresponding learning methods as the solution to our specific problem.

As of our best knowledge, firstly, there is no systematic work on linguistic patterns, including the representation, acquisition, and application of patterns for conceptual modeling yet. Secondly, no researchers have conducted experiment on automatic acquisition of linguistic patterns for conceptual modeling. Therefore, though we don't invent any new representations and pattern learning methods, our proposal and future experiment will have important implications for future work in automating conceptual modeling.

References

Alani, H., Kim, S., Millard, D. E., Weal, M. J., Lewis, P. H., Hall, W. and Shadbolt, N. R., "Automatic Extraction of Knowledge from Web Documents", In *Proceedings of 2nd International Semantic Web Conference - Workshop on Human Language Technology for the Semantic Web and Web Service*, Sanibel Island, Florida, USA, 2003.

Argamon S., Dagan, I., and Krymolowski, Y., "A memory-based approach to learning shallow natural language patterns", *Proceedings of the 17th international conference on Computational linguistics*, Montreal, Quebec, Canada, 1998, pp. 67-73.

Barbuceanu, M., Fox, M.S., Hong, L., Lallement Y., and Zhang, Z., "Building Agents for the Customer Service Front", *the Fifteenth Annual Conference on Innovative Applications of Artificial Intelligence (IAAI-03)*, Acapulco, Mexico, 2003, pp. 35-42.

Berlan, M. & Charniak E., "Finding Parts in Very Large Corpora", In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*, 1999.

Booch, G., "*Object-Oriented Analysis and Design with Applications*", 2nd Ed., Benjamin Cummings, 1994.

Brüninghaus, S., Ashley, K.D., "Improving the Representation of Legal Case Texts with Information Extraction Methods", *Proceedings of the 8th International Conference on Artificial Intelligence and Law*, St. Louis, Missouri, United States, 2001, pp. 42-51.

Califf, M.E. and Mooney, R.J., "Relational Learning of Pattern Matching Rules for Information Extraction", *Proceedings of AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*, 1998, pp. 6-11.

Califf, M.E. and Mooney, R.J., "Bottom-Up Relational Learning of Pattern Matching Rules for Information Extraction", *Journal of Machine Learning Research*, Vol. 4, Issue 2, 2004, pp. 177-210.

Chang, C.H. and Lui, S.C., "IEPAD: information extraction based on pattern discovery", *Proceedings of the tenth international conference on World Wide Web*, Hong Kong, 2001, pp. 681-688.

Chang, C.H., Lui, S.C., and Wu, Y.C., "Applying Pattern Mining to Web Information Extraction", PAKDD 2001.

Chen, K.J., Tsuei, W., Chien, L.F., "PAT-Trees with the Deletion Function as the Learning Device for Linguistic Patterns", *COLING-ACL 1998*, pp. 244-250.

Chen, P.P., "English Sentence Structure and Entity-Relationship Diagrams", *Information Sciences*, 29, 1983, pp. 127-149.

Cimiano, P., Handschuh, S., and Staab, S., "Towards the Self-Annotating Web", *Proceedings of the 13th international conference on World Wide Web*, New York, USA, 2004, pp.462-471.

Cunningham, H., "GATE, A General Architecture for Text Engineering", *Computers and the Humanities*, Vol. 36, 2002, pp. 223-254

Cunningham, H., Maynard, D., and Tablan., V., "JAPE: a Java Annotation Patterns Engine (Second Edition)", Technical report CS--00--10, University of Sheffield, Department of Computer Science, 2000.

Dimitrov, M., Bontcheva, K., Cunningham, H., and Maynard, D., "A Light-weight Approach to Coreference Resolution for Named Entities in Text", *Anaphora Processing: Linguistic, Cognitive and Computational Modeling*, 2004.

Dori, D., et al, "SMART: System Model Acquisition from Requirements Text", *BPM 2004*, LNCS 3080, 2004, pp. 179-194.

Downey, D., Etzioni, O., Soderland, S., and Weld, D., "Learning Text Patterns for Web Information Extraction and Assessment", *AAAI-2004 Workshop on Adaptive Text Extraction and Mining*, 2004.

Etzioni, O., Cafarella, M., and Downey, D., "Web-Scale Information Extraction in KnowItAll", *Proceedings of the 13th international conference on World Wide Web*, New York, USA, 2004, pp.100-110.

Freitag, D., "Information Extraction from Html: Application of A General Learning Approach", *Proceedings of the 15th Conference on Artificial Intelligence*, 1998, pp. 517-523.

Goldin, L. and Berry, D.M., "A prototype Natural Language Text Abstraction Finder for Use in Requirements Elicitation", *Automated Software Engineering Journal*, Vol. 4, No. 4, 1997, pp. 375-412.

Heart, M.A., "Automatic Acquisition of Hyponyms from Large Text Corpora", In *Proceedings of the 14th International Conference on Computational Linguistics*, 1992.

Hearst, M.A., "Untangling Text Data Mining", *Proceedings of ACL'99: the 37th Annual Meeting of the Association for Computational Linguistics*, 1999.

Hirschman, L., Park, J.C., Tsujii, J., Wong, L., and Wu, C., "Accomplishments and Challenges in Literature Data Mining for Biology", *Bioinformatics*, Vol. 18, No. 12, 2002, pp. 1553-1561.

Hsu, C.N., and Dung, M.T., "Generating Finite-state Transducers for Semi-structured Data

Extraction from the Web”, *Information Systems*, Vol. 23, No. 8, 1998, pp. 521–538.

Hu, X., Yoo, I., Song, I.Y., Song, M., Han, J., and Lechner, M., “Extracting and Mining Protein-Protein Interaction Network from Biomedical Literature”, *Proceedings of the 2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, 2004.

Huffman, S., “Learning Information Extraction Patterns from Examples”, *Statistical and Symbolic Approaches to Learning for Natural Language Processing*, pp.246--260, 1996.

Jacobson, I., Christerson, M., Jonsson, P., and Overgaard, G., “*Object-Oriented Software Engineering: A Use Case Driven Approach*”. Addison-Wesley, 1992.

Jones, R., McCallum, A., Nigam, K., and Riloff, E., "Bootstrapping for Text Learning Tasks", *IJCAI-99 Workshop on Text Mining: Foundations, Techniques, and Applications*, 1999.

Katz, B., “From Sentence Processing to Information Access on the World Wide Web”, *AAAI Spring Symposium*, 1997.

Kim, J.T. and Moldovan, D.I., “Acquisition of Linguistic Patterns for Knowledge-Based Information Extraction”, *IEEE Transactions on Knowledge and Data Engineering*, Volume 7, Issue 5, 1995, pp. 713-724.

Kitani, T., Eriguchi, Y., Hara, M., “Pattern Matching In the Textract Information Extraction System”, *COLING 1994*, pp. 1064-1070.

Krupka, G.R., “Description of the SRA System as used for MUC-6”, In *Proceedings of the Fourth Message Understanding Conference (MUC-6)*, Morgan Kaufmann, 1995, pp. 221-236.

Kuhn, R. and Mori, R., “Application of Semantic Classification Trees to Natural Language Understanding”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 5, 1995.

Kushmerick, N., Weld, D., and Doorenbos, R., “Wrapper Induction for Information Extraction”, In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, 1997.

Larman, C., “*Applying UML and Patterns (2nd)*”, Prentice Hall, 2001.

Muslea, I., “Extraction Patterns for Information Extraction Tasks: A Survey”, In *Proceedings of AAAI '99 Workshop on Machine Learning for Information Extraction*, 1999.

Muslea, I., Minton, S., and Knoblock, C., “A Hierarchical Approach to Wrapper Induction”, In *Proceedings of the 3rd International Conference on Autonomous Agents*, Seattle, WA, 1999.

Overmyer, S., Lavoie, B., and Rambow, O., "Conceptual Modeling through Linguistic Analysis Using LIDA", *IEEE*, 2001.

Riloff, E., "Automatically Constructing a Dictionary for Information Extraction Tasks", *Proceedings of the Eleventh National Conference on Artificial Intelligence*, AAAI Press/the MIT Press, 1993, pp. 811-816

Riloff, E., "Automatically Generating Extraction Patterns from Untagged Text", *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996, pp. 1044-1049.

Riloff, E. and Wiebe, J., "Learning Extraction Patterns for Subjective Expressions", *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, 2003.

Rumbaugh, J., Jacobson, I., and Booch, G., "*The Unified Modeling Language: Reference Manual*". Addison Wesley, 1999.

Sasaki, Y. and Matsuo, Y., "Learning Semantic-Level Information Extraction Rules by Type-Oriented ILP", *18th International Conference on Computational Linguistics (Coling-2000)*, Saarbrücken, 2000, pp. 698-704.

Soderland, S., Fisher, D., Aseltine, J., and Lehnert, W., "CRYSTAL: Inducing a Conceptual Dictionary", *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1995, pp. 1314-1319.

Soderland, S., "Learning Information Extraction rules for Semi-structured and free text", *Machine Learning*, Vol. 34, 1998, pp. 233-272.

Song, I.Y., et al, "A Taxonomic Class Modeling Methodology for Object-Oriented Analysis", *Information Modeling Methods and Methodologies*, 2004.

Tan, A.H., "Text mining: The state of the art and the challenges", In *proceedings, PAKDD'99 Workshop on Knowledge discovery from Advanced Databases (KDAD'99)*, Beijing, April, 1999, pp. 71-76,

Wirfs-Brock, R., Wilkerson, B., and Wiener, L., *Designing Object-Oriented Software*, Prentice Hall, 1990.

Yi, J., Nasukawa, T., Bunescu, R., and Niblack, W., "Sentiment Analyzer: Extracting Sentiment about a Given Topic using Natural Language Processing Techniques", *Third IEEE International Conference on Data Mining (ICDM 2003)*, 2003, pp. 427 – 434.

Zheng, J., Wang, X., and Li, F., "Clustering-based Automatic Generation of Extraction Patterns", *Proceedings of Natural Language Processing and Knowledge Engineering*, 2003, pp. 168- 173

Zisman, A., Spanoudakis, G., Perez-Minana, E., and Krause, P., “Tracing Software Requirements Artefacts”, *Proceedings of the 2003 International Conference on Software Engineering Research and Practice*, 2003.