Extract Conceptual Graphs from Plain Texts in Patent Claims

Shih-Yao Yang and Von-Wun Soo

Department of Computer Science, National Tsing Hua University,
HsinChu 300, Taiwan
yao@cs.nthu.edu.tw, soo@cs.nthu.edu.tw

Abstract. This paper develops techniques to extract conceptual graphs from a patent claim by using syntactic information (POS, and dependency tree) and semantic information (background ontology). Due to plenteous technical domain terms and lengthy sentences prevailing in patent claims, it is difficult to apply a NLP Parser directly to parse the plain texts in the patent claim. This paper combines such techniques as regular expressions, finite state machines, Part-Of-Speech tags, conceptual graphs, domain ontology and dependency tree, to convert a patent claim into a formally defined conceptual graph. From 100 patent claims, the average precision and recall of a concept class mapping from the patent claim to domain ontology are 96% and 87% respectively and the average precision and recall for ***Real*** relation class mapping is 97% and 98% respectively. For the concept linking of a relation, the average precision is 79%.

Keywords: Conceptual Graph, Natural Language Processing, Patent Document Analysis, Patent Claims Information Extraction, Ontology, Dependency Tree, Regular Expression.

1 Introduction

Information extraction (IE) from a plain text is not an easy task. Traditional information extraction method [1] extracts patterns, part of sentences, using regular expressions or pattern templates. [2] proposed a natural language processing method to extract the structure and chunk phrases of a patent claim. Both of them did not utilize semantic inferences from the point of view of either a complete sentence or a whole document. [3] proposed a method to analyze the rhetorical structure of a patent claim to improve the readability of the patent claim but did not consider the semantic relations among words in the patent claim. In this paper, we propose a method to extract conceptual graphs from the plain text of a patent claim based on the syntactic information (Part-Of-Speech tags and Dependency Tree) generated by a NLP parser and the semantic information derived from a background domain ontology. After information extraction, the patent claim sentences can be mapped into a single connected conceptual graph [4]. The conceptual graph can also be converted to a corresponding first-order-logic formula [5] in a way that the semantic inferences can be carried out further.

Advances in Artificial Intelligence and Applications
Research in Computer Science 32, 2007, pp. 315–329

Received 14/06/07
Accepted 31/08/07
Final version 17/09/07

However, there are two main problems to build a conceptual graph based on the semantic information extracted from a patent claim: (1) A patent claim sentence is usually so lengthy as to cause a parsing crash; (2) It must have a systematic method to map a patent claim to its corresponding semantic structure, namely a conceptual graph in terms of a background domain ontology. In dealing with problem 1, the system analyzes the organization of a patent claim and uses a finite state machine to split a patent claim sentence into a set of sub-sentences before applying an NLP parser. In dealing with problem 2, we construct conceptual hierarchies for the domain ontology and map contents of a patent claim to the conceptual classes in the domain ontology in terms of Part-Of-Speech tags and the contents themselves, and connect the conceptual classes which have semantic relations in the patent claim by using the dependency relation information in the dependency tree generated by the NLP parser. Our ultimate goal of research is to develop automated approaches to conduct comparison and summarization over patent claims based on the extracted conceptual graphs.

2 Dependence Tree and Conceptual Graphs

2.1 Dependency Tree

A dependency tree [6] is a directed tree that is different from a phrase structure in representing a sentence. Each node in a dependency tree can be represented by a word from a sentence and each edge in a dependency tree may link two words by a syntactic relation. In contrast, each node in a phrase structure is either a Part-Of-Speech tag or a word from a sentence and each edge in a phrase structure may link two nodes without a syntactic relation between them. Both the phrase structure and the dependency tree for the sentence "particles comprise a smectite clay" where smectite is a type of the clay from USPTO [7] 7112123 generated by the Stanford NLP Parser are shown in Figure 1. A phrase structure takes into consideration among multi-word constituents, while a dependency tree considers the relationship between two individual words.

2.2 Conceptual Graphs

A conceptual graph is a kind of formal knowledge representation in terms of Semantic Networks and existential graphs. It is a bipartite graph which consists of two types of nodes, a concept node denoted as a rectangle and a relation node denoted as a circle, where only concepts can link with a relation. "In a conceptual graph, concept nodes can represent entities, attributes, states, and events, while relation nodes can show how the concepts are interconnected"[4]. A conceptual graph is related to a support which is the background domain ontology for a specific domain. This support can be formalized as:

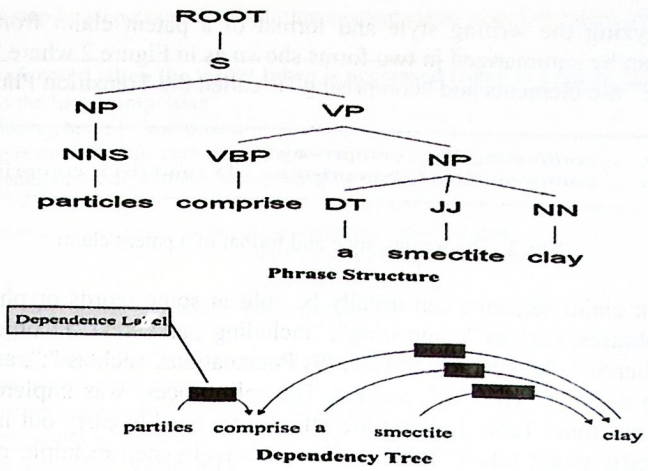


Fig. 1. A phrase structure vs. a dependency tree for the sentence "particles comprise a smectite clay".

1. A concept hierarchy which has the partial ordering operator over the concepts. For concept labels A, B, and C, if $A \leq B$ and $B \leq C$, then $A \leq C$.
2. A relation hierarchy which is disjoined from the concept hierarchy. Each relation links two or more concepts.
3. A set of star graphs. Each star graph consists of a relation and a set of concepts that the relation can link.

A set of individual markers that refer to specific entities. The generic marker "*" refers to an unspecified entity. If an individual marker conforms to type C, it must also conform to all supertypes of C. For example, John is an individual marker of the concept "Man", then it is also an individual marker of the concept "Person" which is a supertype of "Man".

3 Split a Patent Claim into a set of Sub-Sentences

To parse a patent claim using the NLP Parser, one must deal with the problems of a lengthy sentence in the patent claims. We split a long sentence into a set of sub-sentences and then parse each sub-sentence one by one and then combine the parsing results of each sub-sentence. However, to split a patent claim sentence into several sub-sentences, it must make sure that each sub-sentence is not only syntactically parsable but also concept preserving. By "parsable", it means the NLP Parser can parse the sentence in reasonable time. The maximal length of a sub-sentence is set at 70 words in the current application because it is a threshold above which the NLP Parser tends to crash. Concept preserving means the split would not lose or distort the meaning of the original patent claim. To ensure the two requirements, good heuristics of splitting are devised according to writing style and format of a patent claim.

By analyzing the writing style and format of a patent claim from a corpus of patents, it can be summarized in two forms shown as in Figure 2 where "A", "B", "C", "D", and "E" are elements and "comprising" is called the Transition Phrase.

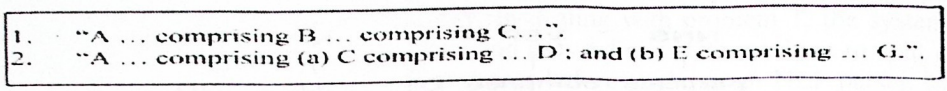


Fig. 2. The writing style and format of a patent claim.

A patent claim sentence can usually be split at some words or phrases: (1) The transition phrases, such as "comprising", "including", and etc; (2) Conjunction words, such as "whercin", "therefore", and etc; (3) Punctuations, such as ";", and "."; (4) List Items, such as "(a)", "(i)", "a)", and etc. The split process was implemented using a finite-state machine. Table 1 shows the actions that need to carry out in order to deal with a specific target token. Table 2 shows a step-by-step example of splitting the patent claim from US patent 6979252 shown in Figure 3. According the devised splitting heuristics, six parsable sub-sentences were obtained and none of them lose or alter their original meaning in the original sentence.

Table 1. Actions performed in dealing with a target token

Target Token	Actions performed on the target token

Transition Phrase (comprising, including, having, consisting of, ...)	1. Set transition_phrase = Target Token;
	2. left_string + transition_phrase + right_string
	Set sub-sentence = left_string + Transition Phrase + temp_word
	// temp_word is a dummy word
	3. Parse the sub-sentence and save the parsing result.
	4. Get the subject of temp_word from the parsing result.
	And add ("the" + subject) before the transition_phrase and continues the process.
	// The system utilizes the determiner "a" and "the" to determine
	// if a noun coming forth for the first time.

Connected-word (wherein, therefore, ...)	1. Set sub-sentence = the left string of Connected-word 2. Parse the sub-sentence and save its parsing result. 3. Continues the process.

List Item ("(a)", "(i)", "a)", ...)	1. Read the next token
	If (the next token is a VBG (Verb, gerund/present participle) or a VB (Verb, base from))
	Add "a step which" before the token
	End if

Punctuation (";")	The action is same as the action of Connected-word.

Table 2. A step by step example of splitting a patent claim from US patent 6979252.

Step	**Actions performed when the target token is processed** (Input is a patent claim)
1.	The target is the first "**comprising**" 1. Set transition_phrase = "comprising" *// left_string is the left of the current target token* 2. Set sub-sentence = left_string + "comprising" + "temp_word" "A method for separating and removing soluble polymeric silicates in a polishing slurry, said slurry comprising temp_word" *// in patent document "said" means "that", so we change "said" to "that" before parsing* 3. Parse the sub-sentence and save its parsing result. 4. The subject of "temp_word" is "slurry". Add "slurry" before the "comprises" and continues the process. *// reset the patent claim string* *//right_string is the right of the current target token* Set patent_claim = "the slurry" + "comprises" +right_string
2.	The target is the second "**comprising**" 1. Set transition_phrase = "comprising" 2. Set sub-sentence = left_string + "comprising" + "temp_word" "the slurry comprises a colloidal dispersion of silica, prior to a chemical mechanical polishing process, the method comprising temp_word" 3. Parse the sub-sentence and save the parsing result. The subject of temp_word is "method". Add "method" before the Transition Phrase and continues the process. *// change "comprising" to "comprises" to make it grammatical* Set patent_claim = "the method" + "comprises" + right_string
3.	The target is "**a)**" Read the next token "introducing". Because "introducing" is a VBG. add "a step which" before "introducing".
4.	The target is the first ";" 1. Set sub-sentence = the left string of ";" "the method comprises a step which introducing the polishing slurry into a centrifuge" 2. Parse the sub-sentence and save the parsing result.
5.	The target is "**b)**" Read next token "separating". Because "separating" is a VBG. add "a step_b which" before "separating"
6.	The target is the second ";" 1. Set sub-sentence = the left string of ";" "the method comprises a step which separating via centrifugation the soluble polymeric silicates, as a portion, from the polishing slurry to yield a product slurry" 2. Parse the sub-sentence and saving the parsing result.
7.	The target is "**c)**" Read next token "removing". Because "removing" is a VBG, add "a step which" before "removing" and skip "and" before "c)"
8.	The target is "**wherein**" 1. Set sub-sentence = the left string of "wherein" "the method comprises a step which removing the product slurry from the centrifuge" 2. Parse the sub-sentence and save the parsing result.
9.	The target is the end of patent claim 1. Set sub-sentence = current all token sequences "wherein the product slurry has a lower level of soluble polymeric silicates than does the polishing slurry" 2. Parse the sub-sentence and save the parsing result.

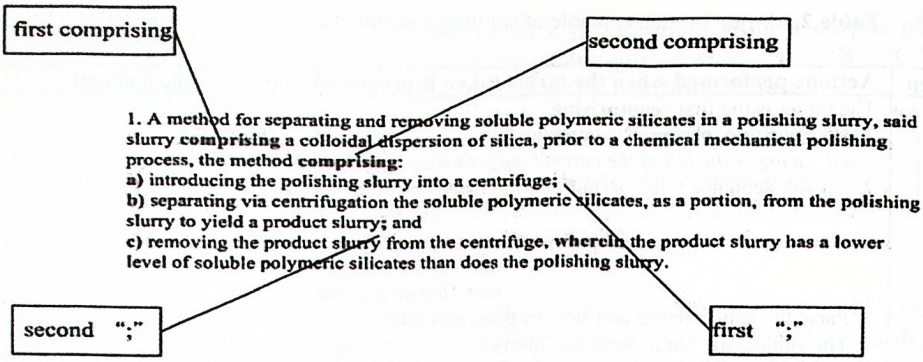


Fig. 3. A patent claim from US patent 6979252

4 Domain Ontology

The background domain ontology includes the concept and relation hierarchies, a set of star graphs that decide which concepts are neighbors of a given relation, and the markers of the concept and relation classes.

-Concept hierarchies

By analyzing the format and writing style of a patent claim, we constructed the high levels of concept hierarchy as shown in Figure 4.

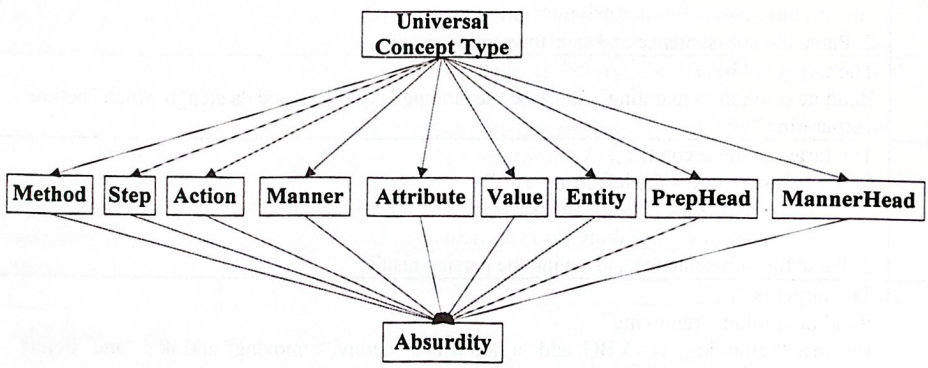


Fig. 4. The Concept Hierarchy of a Patent Claim.

- Relation hierarchies and Star graphs of the relation

We summarize totally 4 relations and Table 3 shows the degree of neighbors for each relation and the candidate concepts that the relation can link.

Table 3. A step by step example of splitting a patent claim from US patent 6979252.

Relation labels	Neighbors (Star graphs of the relation)	Description
Verb	(Entity, Entity, PrepHead, MannerHead);(Entity, SpecialPhrase, PrepHead, MannerHead);(Entity, Value, PrepHead, MannerHead);(Method, Step, PrepHead, MannerHead);(Method, Entity, PrepHead, MannerHead);(Method, Action, PrepHead, MannerHead);(Method, Entity, PrepHead, MannerHead)	The first neighbor represents a subject and the second neighbor represents an object of the verb. The third neighbor PrepHead is a link between relations Prep and Verb, and the fourth neighbor MannerHead is a link between relations AssAtt and Verb.
Contains	(Entity, Entity);(Method, Step);(Method, Entity);(Method, Action);(Method, Entity)	The first neighbor contains the second neighbor.
Prep	(Entity,Action);(Method,Action);(Step,Action);(Entity, Entity);(Method, Entity);(Entity, Value);(PrepHead, Entity);(PrepHead, Value);(PrepHead, Action);(PrepHead, Step)	The first neighbor is the subject of the relation Prep. The second neighbor is the object of the relation Prep.
HasAtt	(Method, Attribute);(Step, Attribute);-(Entity, Attribute);(Attribute, Manner);(MannerHead, Manner)	The second neighbor is the attribute of the first neighbor.

- ***Real*** vs. ***Auxiliary***

The instance of a ***Real*** class is always from the content of a patent claim, while that of a ***Auxiliary*** class is not from the content of a patent claim but can play a role as following:

(a) to link two relations

In a conceptual graph, a relation is not allowed to directly link with another relation. But in reality in a natural language sentence, a relation usually may allow to link with another relation in order to express a complete notion. For example, in the sentence of "*A method for planarizing a wafer, comprising: positioning the wafer on a platen*" from USPTO 7121919, "for" is created as a relation class with label ***Prep*** and "planarizaing", "comprising" and "positioning" are created as relation classes with label ***Verb***. The system creates a Auxiliary concept class ***Action*** to link relation ***Prep***:for to relation ***Verb:***planarizing and to link relation ***Verb:***comprising with relation ***Verb:***positioning as shown in Figure 5.

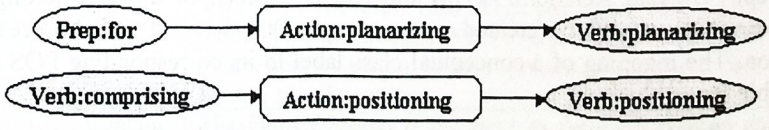


Fig. 5. An example of using a Auxiliary concept to link two relations.

(b) to link two concepts

In a conceptual graph, a concept can not directly link with another concept. The system creates a ***Auxiliary*** relation class to link these two concepts if no ***Real*** relation class can link them. For example, in the phrase of "smectite clay', "smectite" is created as a concept class with label ***Attribute*** and "clay'" is created as a concept class with label ***Entity***. The system uses relation class ***HasAtt***:* to link concept ***Entity***:clay' to concept ***Attribute:***smectite as shown in Figure 6 where ***HasAtt***:* is a ***Auxiliary*** class.

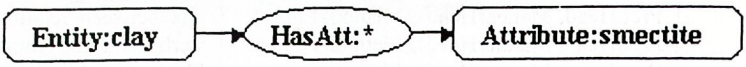


Fig. 6. An example of using a Auxiliary concept to link two concepts.

(c) to link a relation with a set of same concept classes (neighbors)

In a conceptual graph, each relation has a fixed degree (number) of neighbors but it is difficult to decide the degree of neighbors for a relation in advance because a relation may have potentially an infinite number of neighbors. For example, A Verb (Relation) may have more than one Modifier (Concept).

Table 4 shows the types of conceptual class label in terms of either real/Auxiliary

Table 4. The types of conceptual class label in terms of either real/Auxiliary.

Type	Conceptual class label
Real	Method, Step, Manner, Attribute, Entity, Value, Verb, Contains, Prep
Auxiliary	Action, PrepHead, MannerHead, HasAtt

5 Build the Conceptual Graph Based on the Parsed Dependency Trees

To build the conceptual graph from the dependency instances generated by NLP Parser, there are three problems:

Problem 1: *How to decide what the conceptual classes (concepts or relations) of the head and its dependent in a dependency instance and how to assign correct conceptual class label to them?*

For the problem 1, the system divides the Part-Of-Speech into two categories, Concept-POS and Relation-POS. If the Part-Of-Speech of a word belongs to the Concept-POS, it will be created as a concept. Otherwise, it will be created as a relation. The mapping of a conceptual class label to its corresponding POS is shown in Table 5.

Problem 2: *How to link the conceptual classes between a head and its dependent?*

To link the conceptual classes between *a head* and its *dependent*, only three kinds of combinations (concept_1, concept_2), (relation, concept), and (relation_1, relation_2) are allowed. Table 6 shows the dependency relations between a head and its dependent and their corresponding conceptual graph in terms of their dependency

labels, "nsubj", "dobj", "advmod" and "amod". For the dependency label "advmod", because a "verb" may have more than one manner, the system creates a concept "MannerHead" to link the manners of a verb.

Problem 3: *How to combine the conceptual graphs of all sub-sentences?*

To combine all conceptual graphs of each sub-sentence into a single conceptual graph, the system uses the indefinite or definite determiners "A" and "The" respectively to check if a concept is a reference. If a concept C has a definite determiner "The" means that it is a reference, the arc connected to C will be changed to connect to the same concept that has the determiner "A" but has a same name as the concept C.

✍ **A Real Example to extract the concept graph**

We use sub-sentence "*the method comprises a step which introducing the polishing slurry into a centrifuge*" split from US patent 6979252 to show the processes to extract the conceptual graph as shown in Table 2. Figure 7 is the Part-Of-Speech for each word and Table 7 shows the processes of extracting a conceptual graph from dependency tree of the sentence step by step. Figure 8 shows the final conceptual graph of the sentence.

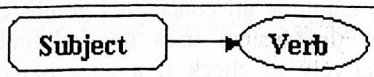
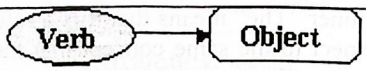
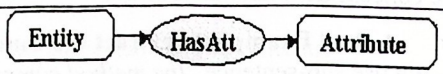
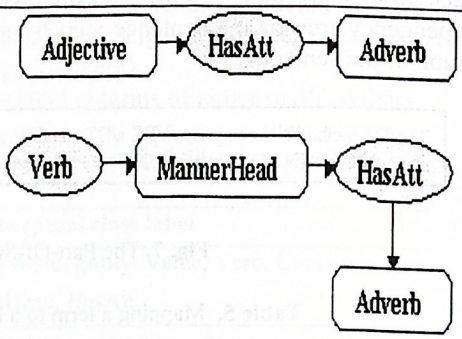
"the/DT method/NN comprises/VBZ a/DT step/NN which/WDT introducing/VBG the/DT polishing/VBG slurry/NN into/IN a/DT centrifuge/NN"

Fig. 7. The Part-Of-Speech of a sentence.

Table 5. Mapping a term to a Real Conceptual Class Label.

Conceptual class	Conceptual Class Label	Part-Of-Speech of the term
Concept-POS	Method	**NN** (Noun, singular or mass) and the term phrase is "method"
	Step	**NN** (Noun, singular or mass) and the term phrase is "step"
	Manner	**RB** (Adverb):**RBR** (Adverb, comparative); **RBS** (Adverb, superlative)
	Attribute	**JJ** (Adjective):**JJR** (Adjective, comparative);**JJS** (Adjective, superlative):**VBG** (Verb, gerund/present participle);**VBN** (Verb, past participle);
	Entity	**NN** (Noun, singular or mass);**NNS** (Noun, plural);**NNP** (Proper noun, singular);**NNPS** (Proper noun, plural);
	Value	**CD** (Cardinal number);
Relation-POS	Verb	**VB** (Verb, base form);**VBD** (Verb, past tense);**VBG** (Verb, gerund/present participle);**VBN** (Verb, past participle);**VBP** (Verb, non-3rd person, singular, present); **VBZ** (Verb, 3rd person, singular, present)
	Contain	The Part-Of-Speech is the same as the Verb relation and the verb phrase must be the transition phrase, "comprising", "having", "including", and "consisting of"
	Prep	**IN** (Preposition);**TO** (infinitival to);**RP** (Particle)

Table 6. The relation between a dependent and its head and their corresponding conceptual graph.

Dependency	The relation between a dependent and its head	Conceptual graph
nsubj	A subject depends on a verb and the conceptual class is **(relation, concept)**	
dobj	A direct object depends on a verb and the conceptual class is **(relation, concept)**	
amod	An adjective depends on a noun, the conceptual class is **(concept_1, concept_2)**. The system creates a ***Auxiliary*** relation "HasAtt" to link the two concepts.	
Advmod	1. An adverb depends on an adjective and the conceptual class is **(concept_1, concept_2)**. The system creates a ***Auxiliary*** relation "HasAtt" to link the two concepts. 2. An adverb depends on a verb and the conceptual class is **(concept, relation)**. The system creates a concept "MannerHead" to link the relation "Verb" and creates a ***Auxiliary*** relation "HasAtt" to link "MannerHead" with the concept "Adverb".	

6 Experiments

The corpus used for our test consists of 1700 patent documents downloaded from USPTO related to the CMP (chemical mechanism polishing) domain by using the search keywords "CMP" and "chemical mechanism polishing" in the Title field. Only the first claim (independent claim) of the 1700 patent documents was selected as the test corpus. From the 1700 patent claims, the average length is 150 words and there are 1458 patent claims whose lengths are greater than 70. Using the split heuristics in Table 3, we obtained 16505 (99%) sub-sentences whose lengths are less than 70.

Table 7. Extract the conceptual graph step by step from dependency tree of a sentnece.

Step	Triple	Extraction of Semantic Relation
1	det(method, The)	Create concept "Method" with generic marker "*"
2	nsubj(comprises, method)	1. Create relation "Contains" with generic marker "comprises" 2. Link "Method:*" to "Contains:comprises"
3	det(step, a)	Create concept "Step" with generic marker "*"
4	dobj(comprises, step)	Link "Contains:comprises" to "Step:*"
5	rel(introducing, which)	Create relation "Verb" with generic marker "introducing"
6	rcmod(step, introducing)	Link "Step:*" to "Verb:introducing"
7	det(slurry, the)	Create concept "Entity" with generic marker "slurry"
8	amod(slurry,polishing)	1. Create concept "Attribute" with generic marker "polishing" 2. Create relation "HasAtt" with generic marker "*" 3. Link "Entity:slyrrt" to "HasAtt:*" 3. Link "HasAtt:*" to "Attribute: polishing"
9	dobj(introducing, slurry)	Link "Verb:introducint" to "Entity:slurry"
10	prep(introducing, into)	1. Create concept "PrepHead" with generic marker "*" 2. Create relation "Prep" with generic marker "into" 3. Link "Verb:introducint" to "PrepHead: *" 4. Link "PrepHead:*" to "Prep:into"
11	det(centrifuge, a)	Create concept "Entity" with generic marker "centrifuge"
12	pobj(into, centrifuge)	Link "Prep:into" to "Entity:centriguge"

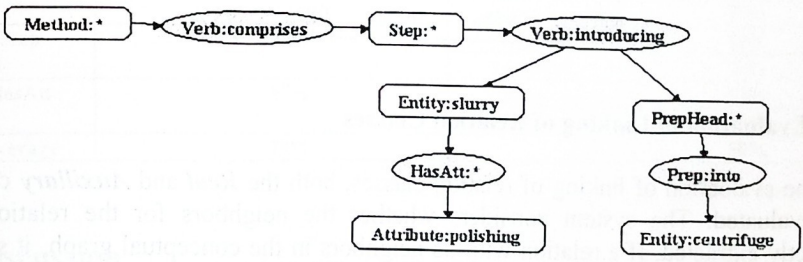


Fig. 8. The conceptual graph of a sentence.

To evaluate the conceptual graph building, we select 100 patent claims as the test corpus and build the correct conceptual graph of each patent claims manually. The system compares the conceptual graphs extracted by the automated approach with those extracted manually and uses precision and recall to evaluate the automated approach.

6.1 Evaluation of Mapping from Contents to Concept or Relation Classes

For the evaluation of mapping from contents to concept or relation classes, only the ***Real*** classes are evaluated because they are extracted directly from the content of the patient claims. The system only consider if it appears in both the content of the original sentence and the corresponding conceptual graph. The measure formulas are below:

$$\textbf{Precision for the class label} = \frac{(\text{PatentClass I CGClass})}{\text{CGClass}} \quad (1)$$

$$\textbf{Recall for the class label} = \frac{\text{PatentClass I CGClass}}{\text{PatentClass}} \quad (2)$$

Where:

- PatentClass : The PatentClass is the set of the concepts or relations in the patent claims.
- CGClass : The CGClass is the set of the concepts or relations in the conceptual graphs.

Table 8 and Table 9 show the precision and recall for the ***Real*** concept and relation classes respectively.

Table 8. The precision and recall for the ***Real*** concept classes.

	Method	Step	Manner	Attribute	Entity	Value	**Average**
Precision	100%	99%	94%	93%	92%	100%	**96%**
Recall	100%	95%	72%	85%	95%	78%	**87%**

Table 9. The precision and recall for the extraction of the ***Real*** relation classes

	Verb	Contains	Prep	**Average**
Precision	96%	98%	98%	**97%**
Recall	95%	99%	98%	**98%**

6.2 Evaluation of Linking of Relation Classes

For the evaluation of linking of relation classes, both the ***Real*** and ***Auxiliary*** classes are evaluated. The system considers whether the neighbors for the relation are correctly extracted. If a relation with its neighbors in the conceptual graph, it should have corresponding neighbors for the corresponding relation in the patent claim.

$$\textbf{Precision for the Neighbor}_i \textbf{ of the relation class } R_k = \frac{\sum_{R_L \in \text{CGRelation}(R_L)} N\_\text{Value}(N_i(R_k))}{\text{CGRelation}(R_k)} \quad (3)$$

$$\textbf{Precision for the relation class } R_k = \frac{\sum_{R_k \in CGRelation(R_k)} R\_Value(R_k)}{CGRelation(R_k)} \quad (4)$$

Where:

- CGRelation(R_k): The CGRelation(R_k) is the set of the relation R_k in the conceptual graphs.
- $N_i(R_k)$: $N_i(R_k)$ is the i_{th} neighbor of R_k.
- N_Value($N_i(R_k)$) : If the i_{th} neighbor of the R_k in the conceptual graph is the same as the one in the patent claim, then N_Value($N_i(R_k)$) = 1, otherwise N_Value($N_i(R_k)$) = 0.
- R_Value(R_k): $\forall N_i \in R_k$, N_Value ($N_i(R_k)$) = 1 → R_Value(R_k) = 1, otherwise R_Value(R_k) = 0

Table 10 shows the precision of relation extraction for different types of relation classes. It turns out that the most difficult relation extraction belongs to relation class Verb because the average precision of the relation class R_k for Verb is only 70%, while the average precision of the relation class R_k for overall relation classes is about 79%. The last column in Table 10 shows the average precision for each individual neighbor of relation R_k.

Table 10. The precision of relation extraction for different relation classes.

Relation Class R_k	Precision for the relation class R_k	Neighbor$_i$	Precision for Neighbor$_i$ of relation class R_k
Verb	70%	Neighbor$_1$	88%
		Neighbor$_2$	82%
		Neighbor$_3$	90%
		Neighbor$_4$	95%
Contains	77%	Neighbor$_1$	88%
		Neighbor$_2$	87%
Prep	81%	Neighbor$_1$	86%
		Neighbor$_2$	92%
HasAtt	87%	Neighbor$_1$	87%
		Neighbor$_2$	94%
Average	**79%**		**89%**

7 Discussion

The system maps the content in a patent claim to ***Real*** concept class using Part-Of-Speech and the content of the sentence. In the performance evaluation for the concept and relation classes, the recalls of the concept classes for ***Manner*** and ***Attribute*** are much poorer than other concept classes while the recalls for all relation classes (such as ***Verb*** and ***Prep***) are relatively good. It is because that the adjectives and the adverbs are much easier to be tagged wrong than the verbs and the prepositions using the Part-Of-Speech Tagger. In a patent claim, there is a lot of VBG (Verb, gerund/present

participle) such as "deposition" and "comprising" to describe an action. Some of them are erroneously tagged as a Noun that affects the mappings for relation classes ***Verb*** and ***Contains***. Since "comprising" may be followed by "step", if it is erroneously tagged, "step" will also be tagged wrong and therefore it affects the mapping for concept class ***Step***. Some prepositions followed a verb may be erroneously tagged as an adverb that affects the mapping for relation class ***Prep***. There are 36 kinds of Part-Of-Speech generated from the corpus and 20 can be mapped to concepts according to the mapping in Table 7, 7 can not be processed and the rest are special symbols, conjunctions, determiners and punctuations that do need to map to a domain concept.

In the experiment, the precision of neighbor 2 of ***Contains*** is better than that of ***Verb.*** It is because most neighbor 2's of ***Contains*** are ***Step*** while most neighbor 2's of ***Verb*** are ***Entities*** whose precision on concept mapping is worse than those of ***Step***. The neighbor 1 of ***Prep*** is poorer than that of ***Verb*** and ***Contains***, because it is affected not only by the precision of conceptual mapping but also by that of neighbor 3, ***PrepHead***, of the ***Verb*** where ***PrepHead*** is supposed to link all ***Preps*** of the ***Verb***. The precision on neighbor 2 of ***Verb*** and ***Contains*** are heavily affected by the correctness of dependency tree. It is because most objects may have a lot of modifiers before them that cause ambiguities in parsing result. The precision on neighbors of ***HasAtt*** are all much better, because most triples of ***Attribute*** and ***Entity*** or triples of ***Attribute*** and ***Manner*** tend to be correct.

8 Conclusion and Future Work

We have successfully converted a patent claim to a conceptual graph based on the domain ontology. This paper is a pioneer to study on extracting a conceptual graph from a patent claim based on the dependency tree generated from a parser. In the performance evaluation, the average precision and recall for ***Real*** concept class mapping is 96% and 87% respectively and the average precision and recall for ***Real*** relation class mapping is 97% and 98% respectively. It means most terms in the patent claim can be correctly mapped to the domain ontology. The average precision of relation class ***Contains*** is 77% that means most structural information can be extracted correctly. The average precision of relation class ***HasAtt*** is 87% and the average precision of relation class ***Prep*** is 81% that mean the most attributes of an element can be extracted correctly. The average precision of relation class ***Verb*** is only 70% if all the neighbors must be extracted correctly as shown in Table 10 that means most processes or functions in a patent claim cannot be easily extracted correctly. The average precision of each individual neighbor of relation class ***Verb*** is 88% that means it is difficult to improve the precision of extracting relation ***Verb*** by improve the precision of extracting its individual neighbor. After examining the data, we found that most errors were due to the ambiguities generated by the NLP parser. We could design more regular expression patterns to identify more semantic patterns such as the range value ("*solids comprising about 1 to about 10 by weight clay abrasive particles*"), or assignment ("*chemical having a pH greater than 7*") to shorten a patent claim to reduce the ambiguities of applying the parser. In the future,

we will utilize the extracted conceptual graph to facilitate patent processing such as patent summarization and comparison to help the judgment of patent infringement.

References

1. Hobbs, J., Appelt, D., Bear, J., Israel, D., & Tyson, W. M.:FASTUS: A System for Extracting Information from Text, Proceedings, Human Language Technology, USA, (1993) 133-137
2. Sheremetyeva, S.:Natural Language Analysis of Patent Claims. Proceedings of the ACL-2003 workshop on Patent corpus processing Association for Computational Linguistics (2003)
3. Shinmori, A., Okumura, M., Marukawa, Y. & Iwayama, M.:Patent claim processing for readability: Structure analysis and term explanation. In Proceedings of the ACL-03 workshop on patent corpus processing (2003)
4. Sowa, J. F.:Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley (1984)
5. Amati, G. and Ounis, I.:Conceptual Graphs and First Order Logic. The Computer Journal. 43 (2000) 1-12.
6. Marneffe, M., MacCartney, B., & Manning, D. C.: Generating Typed De-pendency Parses from Phrase Structure Parses. Inter-national Conference On Lerc Language Resources and Evaluation (2006)
7. United States Patent and Trademark Office. http://www.uspto.gov/.