Advanced Engineering Informatics 29 (2015) 431–439

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

A two-level parser for patent claim parsing ☆

Jingjing Wang [a,b,*], Wen Feng Lu [c], Han Tong Loh [d]

[a] School of Statistics, Jiangxi University of Finance and Economics, Nanchang 330013, China
[b] Research Center of Applied Statistics, Jiangxi University of Finance and Economics, Nanchang 330013, China
[c] Department of Mechanical Engineering, National University of Singapore, Singapore
[d] Singapore Institute of Technology, Singapore

ARTICLE INFO

Article history:
Received 8 April 2014
Received in revised form 25 January 2015
Accepted 28 January 2015
Available online 21 February 2015

Keywords:
Patent search
Product design
Patent claim
Parsing
Dependency syntax
Parser

ABSTRACT

Patent claim parsing can contribute in many patent-related applications, such as patent search, information extraction, machine translation and summarization. However, patent claim parsing is difficult due to the special structure of patent claims. To overcome this difficulty, the challenges facing the patent claim parsing were first investigated and the peculiarities of claim syntax that obstruct dependency parsing were highlighted. To handle these peculiarities, this study proposes a new two-level parser, in which a conventional parser is imbedded. A patent claim is pre-processed in order to remove peculiarities before passed to the conventional parser. The process is based on a new dependency-based syntax called Independent Claim Segment Dependency Syntax (ICSDS). This two-lever parser has demonstrated promising improvement for patent claim parsing on both effectiveness and efficiency over the conventional parser.

1. Introduction

Patent analysis approaches are generally classified as patent mining or visualization [1]. The rapid growth of published patents has made a call for more sophisticated patent analysis tools. Patent parsing is an essential operation in many patent mining approaches such as function-behaviour-state information extraction [2], concept-based patent search [3] and conceptual graph extraction [4]. However, patent claim parsing is considered very difficult [5]. The claim section of a patent defines the scope of Intellectual Property (IP) protection granted by the patent. In a patent, the claim section is the only part that are examined and conferred for IP protection. In contrast, other parts like the description section or drawings are used for understanding and interpreting the claims, but do not provide any IP protection themselves. Semantic patent claim analysis can examine patents for possible infringements and identify which needs to be manually perused [6]. Moreover, patent claims are important to the value of the patent [7], especially claims in essential patent i.e. those patents that are indispensable for designing and manufacturing products [8]. Besides, the number of claims a patent makes has significant effects on the duration that a patent is under consideration [9]. It is expected that the improvement on patent claim parsing can promote more sophisticated patent analysis and therefore tackle the challenges of the rapid growth of published patents.

The patent claim syntax follows exactly common English grammar but is peculiar [10]. These peculiarities are usually not considered when designing a conventional natural language parser. Therefore, conventional natural language parsers may fail to correctly parse patent claims [5].

To design a completely new data-driven parser, it is necessary to prepare both the training data and a data-driven model that can handle the peculiarities of patent claim syntax. In this way, existent natural language resources are discarded. In contrast, a smarter way is to utilize existent natural language resources by improving the adaptability of a conventional parser. This study applies the latter approach and proposes a two-level parser for patent claim parsing. At the top level, patent claims are pre-processed so that a conventional parser e.g., Stanford parser [11] can be more adaptable to them; while at the bottom level, the conventional parser is evoked to parse the pre-processed claims.

To build such a two-level parser, the peculiarities of the claim syntax that lead to challenges of claim parsing were investigated. A new dependency-based syntax, called Independent Claim Segment Dependency Syntax (ICSDS), was then proposed in order to address these challenges. The two-level parser was finally build based on the proposed dependency-based syntax.

☆ Handled by C.-H. Chen.

* Corresponding author at: School of Statistics, Jiangxi University of Finance and Economics, Nanchang 330013, China. Tel.: +86 0791 83816428.
E-mail addresses: wang_jingjing@jxufe.edu.cn (J. Wang), mpelwf@nus.edu.sg (W.F. Lu), HanTong.Loh@SingaporeTech.edu.sg (H.T. Loh).

http://dx.doi.org/10.1016/j.aei.2015.01.013
1474-0346/

The rest of this paper is organized as the following. The related works are reviewed in Section 2. The Section 3 highlights the challenges facing the patent claim parsing. The Section 4 introduces the proposed approach, including the new dependency-based syntax and the parser system. The Section 5 evaluates both the effectiveness and efficiency of the proposed parser. Lastly, the Section 6 addresses conclusions and gives recommendations for future work.

2. Related works

A complex knowledge-based natural language analysis approach was proposed [12] to capture both the structure and content of a claim text. The knowledge includes both shallow lexicon and predicate lexicon. The shallow lexicon is a word list which was automatically acquired from a corpus of five million words in US patents. The predicate lexicon for claims on apparatuses was manually acquired from a corpus of 1000 US patent claims. It was expected that the proposed claim parsing can be used in machine translation and improvement on the readability of patent claims. However, with regard to performance, this approach was not compared with other approaches.

Since not only structure but also content is useful in patent claim analysis, claim parsing in this study focuses on dependency parsing. Compared with phrase structure (or constituency) parsing, dependency parsing offers an easier way to extract the content of a claim. This is because dependency grammar [13] can explicitly express word-to-word relations. Further, the result of dependency parsing can be converted from that of phrase structure parsing [14]. The phrase structure grammars have a high proportion in formal grammatical systems. Thus, many existent natural language resources can be reused in dependency parsing.

Dependency parsing methods are generally classified into two categories: grammar-based parsing or data-driven parsing. The grammar-based parsing requires grammar or rules, e.g., context-free dependency grammar. In contrast, data-driven parsing does not need grammar or rules; it relies on models, which are learned from training data, to make decisions. The learned models can be classified into graph-based models [15,16], transition-based models [17,18], or hybrid models [19–21].

A simple way to realize domain adaptation is by correcting and enriching the training set of a data-driven parser with domain-specific data. In this way, previous work [22] had retrained a Bohnet's parser. However, the improvement on performance is not very promising. This is probably because the parser lacks a mechanism to handle the peculiarities of patent claim. Besides, effectiveness improvement should consider natural language processing. Most works focus on the detection of a restricted number of prominent verbal relations, including in particular is-a, has-part and cause, while deriving a large number of content relations relies on deep syntactic structures [23].

In parsing, the length of a sentence is the number of tokens, which are words and punctuations in the sentence. Similarly, the length of a claim can be defined as the number of tokens in a claim. The length of an independent claim is usually too long so that the claim cannot be parsed [10]. To improve the efficiency of patent claim parsing, a common strategy is segmentation. An approach [10] was proposed for reducing the length and the complexity of patent claim via claim decomposition. The decomposition can obviously improve the success rate of parsing with a Stanford parser. However, no evaluation results were given to show the accuracy of parsing after decomposition. A finite-state machine [4] was implemented to split a patent claim into a set of sub-sentences before passed to a Stanford parser. This finite-state machine was designed for handling two claim's forms. However, these two claim's forms can not cover all claims. Two segmentation tasks [24] were carried out. The first task segments a claim into three components: preamble, transition and body text; the second task further segment the claim into subordinate and coordinate clauses. The evaluation of these two tasks only focuses on claim segmentation rather than the effectiveness of claim parsing.

Briefly, little previous works had focused on patent claim parsing. It should be noted that the efficiency of patent claim parsing can be improved with claim segmentation. However, it is still a question whether claim segmentation will depress the effectiveness of patent claim parsing. Moreover, simply enriching training set of a data-driven parser does not make very promising improvement on performance. A mechanism is needed to handle the peculiarities of patent claim.

3. Peculiarities of claim syntax

Claim syntax follows exactly English grammar. However, compared to daily English usage, claim syntax is peculiar [10]. This study focuses on those peculiarities that may increase the difficulty of dependency parsing. These peculiarities are highlighted in the following sub-sections.

3.1. Claim template

There are some formal templates for starting a claim. For example, "file folder" is the patented product found in US Patent 7,954,694. The first independent claim starts with "*We claim*:" and a dependent claim starts with "*The file folder of claim 3, wherein*". It is necessary to use these templates for organizing multiple claims.

These templates do not offer any information pertaining to the patented product. In other words, removing these templates does not lead to useful information loss. However, the existence of these templates does increase the difficulty of dependency parsing.

3.2. Post attribute past participle

For regular verbs, the verb form of their past forms is the same as that of their past participles. In claims, the past forms are rarely used since the basic tense is present tense rather than past tense. On the other hand, it is frequent to use a complex noun phrase with post attributive present participle phrase or post attributive past participle phrase. However, given such a complex noun phrase with post attributive past participle phrase, a conventional parser is prone to mark the past participle as a past form. It is because a conventional parser usually treats the input text as a sentence rather than a noun phrase.

3.3. Parenthetical sentence

Parenthesis means that additional word, phrase or sentence is inserted into a passage which would be complete without it. Insertion of an independent sentence is rare in daily English usage, but it is frequent in claims. This increases the difficulty of dependency parsing because a conventional parser usually treats the input text as a single sentence.

3.4. Complex noun phrase as sentence

In claim, it is frequent that an independent sentence consists of only a complex noun phrase rather than has a subject–verb–object structure. Moreover, it is frequent that such a complex noun phrase is inserted into another sentence, i.e., parenthesis. Theoretically, noun phrase as an independent sentence is allowed by dependency grammar. A parser can correctly parse a noun phrase as a sentence

when the noun phrase is simple and stays alone; e.g., when it is a single noun, or when it is a very simple noun phrase structure such as a determiner plus a noun. However, if the noun phrase is complex and stay with other constituents, then the parser is prone to mark the noun phrase as a dependent of another constituent. In other words, the noun phrase is treated as a sentence's constituent rather than a sentence.

3.5. Recursion

Recursion is frequent in independent claims, especially when expressing the structure of the patented invention. In recursion, the predicates of the main sentence and sub-sentence express the same meaning. For instance, "*wherein the body includes a graphical region comprising an ornamental three dimensional sculpture*" (in US Patent 7,917,986) is best analysed as a main sentence "*wherein the body includes a graphical region*" having an embedded sentence "*a graphical region comprises an ornamental three dimensional sculpture*". The predicate of the main sentence is "includes", while the predicate of the sub-sentence is "comprises". The meaning of "includes" and "comprises" are the same when expressing inclusion relationship. This phenomenon increases the difficulty of dependency parsing.

3.6. Coordination

Coordination is frequent in claims since an invention usually includes several parts. In dependency grammar, coordination is defined trickily. For example, in sentence "*A camera comprises a lens and a body*", the head of both "lens" and "body" should be "comprises". However, according to dependency grammar, the head of "and" is assigned as "lens" and the dependency relation is assigned as "coordinator". At the same time, the head of "body" is also assigned as "lens" and the dependency relation is assigned as "conjunct". Therefore, additional step is needed to reveal the reasonable dependency relation. Although the reasonable dependency relation can be revealed, too many coordination increases the difficulty of dependency parsing.

In summary, peculiarities above lead to long claims. Some peculiarities, such as parenthetical sentence, complex noun phrase as sentence, recursion and coordination cause Long Distance Dependencies (LDD). The LDD is a classic critical problem in language modelling.

4. Independent Claim Segment Dependency Syntax

To improve the adaptability of a conventional parser, e.g. Stanford parser [11], a two-level parser for patent claim parsing is proposed. At the top level, the peculiarities of claim syntax are handled so that the conventional parser can be more adaptable to them. At the bottom level, the conventional parser is evoked to parse the pre-processed claims.

Specifically, the claim template peculiarity can be trimmed. The post attribute past participle peculiarity can be corrected with a post-POS tagging. Those peculiarities leading to Long Distance Dependencies (LDD) can be fixed with a segmentation and assembly approach.

The segmentation strategy limits the length of a claim in parsing and guarantees the efficiency of the proposed parser since previous works have shown its promising performance. The assembly process is expected to build segment-to-segment dependency relations (at cross-segment level) and further build cross-segment word-to-word dependency relations. Theoretically, the distance (in terms of segments) between two segments is much smaller than the distance (in terms of words) between two words within these two segments, respectively. Thus, the distance of the dependency becomes shorter and is easier to be captured. The conventional parser is only evoked when parsing word-to-word dependency relations in each claim segment (i.e., with-segment level).

To realize the whole idea discussed above, new dependency syntax is needed. Independent Claim Segment Dependency Syntax (ICSDS) is dependency-based syntax designed for parsing independent claims, which cannot be directly parsed well with conventional parsers. The ICSDS belongs to a class of modern syntactic theories that are all based on dependency relation.

Like other dependency grammar, ICSDS has flowing properties:

(1) Connectivity

All the words are connected with dependency relations. In other words, every word should be assigned at least one dependency relation.

(2) Single Head

Each word must have and can only have one head. In other words, every word should be assigned a head.

In ICSDS, a claim segment is a sequence of words after claim segmentation. Each claim segment (the dependent) must have and only have one head, which is another claim segment or ROOT, i.e., the root of the parsing tree. One word in the dependent must depend on and only depend on one word in the head. In other words, one segment-to-segment dependency relation can be converted into one word-to-word dependency relation.

The ICSDS only has Partial Planarity rather than Planarity. Planarity means that a dependency relation does not cross any other dependency relations when drawn above the words. The Partial Planarity relaxes the conditions by allowing cross drawing if the crossed dependency relations are connecting to ROOT. Planarity means the claim is formed by one sentence. In contrast, Partial Planarity allows a claim to include multiple sentences. This property makes ICSDS adaptable to the parenthetical sentence peculiarity of claim syntax.

The ICSDS follows Conditional Proximity Principle rather than Proximity Principle. Proximity Principle means that each dependent depends on the closest possible head. In ICSDS, a condition is added to the Proximity Principle, i.e., the dependency should satisfy all constraints.

The details of claim segment segmentation, claim segment feature recognition, claim segment parsing, and cross-segment word-to-word dependency construction are given in the following sub-sections.

4.1. Claim segment segmentation

Generally, the segmentation defined in ICSDS follows the natural separation of text. If two portions are linked with only space (punctuation), they are not separated. Two words separated by space (punctuation) usually have a local dependency relation. The design of segmentation mechanism is very important. The segmentation process precedes the assembly process. The mechanism of segmentation can affect that of assembly.

Briefly, any mark that helps separating an independent claim and making the meaning clear is considered as a separator. Known separators belong to three categories: HyperText Markup Language (HTML) element, sequential number, and punctuation mark. Generally, two separators belonging to the same category do not occur consecutively. In contrast, two or three separators belonging to different categories may occur consecutively.

Therefore, a delimiter, which is a mark fixing the boundary of a segment, is defined as a triple in the form of (HTML-element,

sequential-number, punctuation-mark) in ICSDS. For example, a part of the first independent claim of US Patent 4,027,510 is shown as follows:

> 1. *A forceps instrument comprising in combination,*
> *

a. an outer sleeve member,*
> *

b. a guiding viewing-tube support, tubular in shape, and mounted concentrically within said outer sleeve,*
> *

c. a tubular barrel mounted within said outer sleeve substantially concentrically around and axially slidable along said guiding viewing-tube support,*
> ...

The first segment here is "*A forceps instrument comprising in combination*", followed by the first delimiter ("br", "TypeE", ","). The first delimiter contains a HTML element i.e.,
 (formally
), a sequential number of type E, which is defined as an alphabetical sequential number followed with a period, and a punctuation mark i.e., a comma. The third segment is "*a guiding viewing-tube support*", followed by the third delimiter (-, -, ","). The third delimiter contains only a punctuation mark, i.e., a comma.

4.2. Claim segment feature recognition

The claim segment feature recognition step characterizes each claim segment with two features: a starting feature and an ending feature. This is because a segment is usually characterized by its starting portion and ending portion. The features are recognized from lower level elements. These elements include segment length (i.e., the number of tokens in a claim segment), lexicon, part-of-speech (POS) and some self-defined word classes e.g., indefinite article (IA).

Without large training dataset, the segment feature recognition uses a rule-based method. The form of a rule is: {has element 1, has element 2, ..., has element n} $\Rightarrow$ {has feature}. Both the starting feature and the ending feature have the same rule structure. A rule is stored in the form of string. The feature is put at the start of the string before all conditions. For example, a starting feature rule "NP,2,IA,!POS:adjective" means if a segment with length two, starting from an indefinite article, and the second token is not an adjective, then the segment should start from a noun phrase (NP).

A dataset (see Section 5.1 for details), which consists of 173 independent claims, is used for establishing feature rules. The fundamental method of establishing rules is generate and test (or trial and error). Firstly, claim segments are observed one by one. At the same time, feature and elements are manually annotated. With the annotation, each claim can generate two rules: one for starting feature, the other for ending feature. Secondly, the generated rules are sorted and checked. Redundant rules are discarded. If a specific rule can be replaced by a general rule, then the specific rule is discarded. Thirdly, the checked rules are test. Those claim segments whose features cannot be correctly recognized are used for generating new rules. The rule generation and test is repeated until the results of recognition are satisfying.

4.3. Claim segment parsing

With claim segment features, the claim segment parsing step builds segment-to-segment dependencies. If a claim segment relies on another segment to form a sentence, then there exists a dependency relation between these two segments, while the former is called dependent and the latter is called head. If a claim segment does not rely on any other segment to form a sentence, then its head is the ROOT. Therefore, the major task for segment dependency relation recognition is to find a segment's head and their dependency relation.

The segment dependency relation recognition also uses a rule-based method. Two major elements of the rule-based method are dependency rule and dependency constraint. The dependency constraints are used with the dependency constraints together to support segment dependency relation recognition. A dependency rule describes the features of both the dependent and its possible head. The adopted elements in dependency rules include relative position, relative distance, starting feature, ending feature, and punctuation. Moreover, a dependency rule can describe inheritance. A dependent may inherit another dependent's head, according to the inheritance. On the other hand, dependency constraints provide additional requirements on dependent. The dependency constraints are an important mechanism for realizing syntax properties and handling the peculiarities of the claim syntax.

For two segments, if a rule is applicable and all constraints are satisfied, then the dependency relation between the two segments is as the one defined in the rule. If no rule is applicable, "ROOT" will be assigned as the head. For example, four dependency rules and a claim are given in Fig. 1. Four segment features are "NP-S", "NEW", "NP" and "AND". Segment feature "NP" means noun phrase. Segment feature "NP-S" means the first noun phrase in a sentence. Segment feature "NEW" means a start of a new sentence. Segment feature "AND" means "and". The claim consists of two independent sentences: a sentence consisting of segment {1, 2, 7, 8, 9} and a sentence consisting of segment {3, 4, 5, 6}. A dependency relation between two segments is depicted via an arc with an arrowhead towards the head. An arc with solid line means a parsed correct dependency; an arc with dot line means a correct dependency waiting for parsing.

According to the rule "NP $\leftarrow$ NEW" and the Proximity Principle, Segment 7 should depend on Segment 3. Next, according to the rule "AND $\leftarrow$ NEW" and the Proximity Principle, Segment 8 should also depend on Segment 3. A dependency constraint on coordinating conjunction can reject the first incorrect parsing. Briefly, the constraint is that a head cannot accept dependent if its last two dependents are starting with "AND" and "NP", respectively. Thus, Segment 7 will depend on Segment 1, according to the rule "NP $\leftarrow$ NP-S". Consequently, the Partial Planarity property can reject the second incorrect parsing. The search for the head of Segment 8 will omit all segments before it but Segment 1.

A left-to-right parsing algorithm is designed to read the entire segmented claim, and then identify the head of each segment from the left side of the claim to the right side. The pseudo-code is shown as below.

Algorithm: PARSE

```
01 indexOfHead ← Ø
02 foreach current segment s_c in S do
03   getHead ← false;
04   indexOfHead[s_c] ← 0;
05   rule ← PICKRULE(Rules, GETTYPE(s_c));
06   foreach segment s_i that i < c (or i > c) in terms of rule do
07     if EXAMINE(s_i) then
08       if MATCH(rule, GETTYPE(s_i), GETTYPE(s_c)) then
09         getHead ← true;
10         head ← GETHEAD(r);
11         indexOfHead[c] ← index;
12         break;
13 return indexOfHead
```

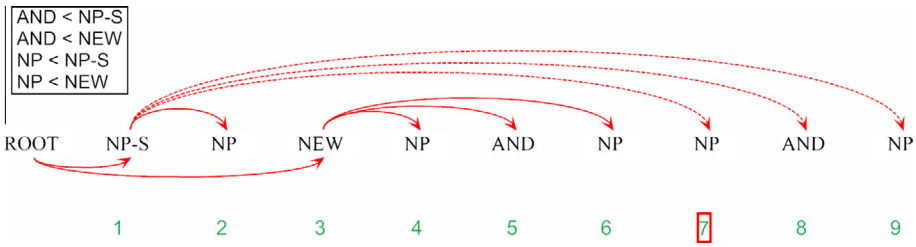

Fig. 1. An example for explaining dependency rules and constraints.

When a segment is in the process of head identification, it is called current segment. The head of current segment is assigned as "ROOT" initially (in Line 04). In the following head search process, a rule corresponding to current segment is picked (in Line 05). According to this picked rule, either the leftward segments or the rightward segments are examined one by one. For each segment under examination, the algorithm first examines dependency constraints (in Line 07). If the examined segment is feasible and it together with current segment can match the picked rule (in Line 08), the head in the rule (in Line 10) and its actual index (in Line 11) will be assigned to current segment.

The establishment of dependency rule and dependency constrain is similar to establishment of feature rule. The 173 independent claims, is used for establishing rules and dependency constraints. The fundamental method of establishing rules and constrains is generated and tested. Firstly, claim segment and features are observed one by one. At the same time, dependencies are manually annotated. With the annotation, each dependency can generate one dependency rule. Secondly, the generated rules are sorted and checked. Redundant rules are discarded. Thirdly, the checked rules are tested. Those dependency cannot be correctly parsed are used for generating additional rules and constraints. The dependency constraints are manually generated when one rule interferes with another rule. The generation and test is repeated until parsing results are satisfying.

4.4. Assembling

Given segment-to-segment dependencies, word-to-word dependencies within each segment, the assembling builds word-to-word dependencies crossing segments and finally returns all word-to-word dependencies. This task can be simply described as a process of the replacement of heads. The head of a word in the dependent segment is replaced with a word in the head segment.

Current implementation only considers two types of cross-segment word-to-word dependencies: verb–noun relation and adjective–noun relation. This is because these two relations are the most common and critical relations. Especially, verb–noun relation is usually LDD. The relation between dependency type and word POS is given in Table 1.

4.5. The two-level parser system

To test the proposed approach, an ICSDS-based parser was designed and built. It is based on Stanford parser. The entire flowchart of the parser system is given in Fig. 2. Since loading a trained Stanford parser requires many seconds, the parser processes claims in a batch.

Table 1
Relation between dependency type and word POS.

Head word	Dependent word	Dependency type
Noun	Noun	Adjective–noun
Noun	Adjective	Adjective–noun
Noun	Verb	Verb–noun
Verb	Noun	Verb–noun

The ICSDS-based parser mainly includes nine modules: trimming, tokenization, POS tagging, POS correction, Stanford parsing, claim segment segmentation, claim segment feature recognition, claim segment parsing and assembling. The trimming is used for handling claim templates. The tokenization is completed by the Stanford tokenizer, which is based on Penn Treebank; while the POS tagging is completed by the Stanford POS tagger. In this way, the errors caused by using different tokenization method or POS tagging method are minimized. The POS correction is used for handling post attribute past participle. The details of last four modules are explained in sub-sections above.

5. Evaluation

Comparison experiments were conducted to test both the efficiency and effectiveness of the proposed approach. Two parsers were used: Stanford parser and ICSDS-based parser.

5.1. Dataset

A product patent dataset is built manually for the tests. In this dataset, there are a total of 273 product patents, which were downloaded from the United States Patent and Trademark Office (USPTO). Each patent is a utility patent and describes a whole product. There are ten product types, including toothbrush, digital camera, razor, lighter, forceps, file folder, mobile phone, surgical scalpel, hypodermic needle and paper punch.

The first independent claims (referring as claim in the rest of this paper) were extracted from the dataset. The length of a claim is defined as the number of tokens it contains. The statistical result is shown in Fig. 3. The frequency of a claim length is the number of times a claim length occurs in the dataset. It is observed that the length of most claims is more than 100 tokens. At the extreme, the length of a claim may exceed 800 tokens.

The whole dataset is separated into training set and test set. The training set consists of 173 patents, while the test set consists of 100 patents. Manual annotation was carried out to label every claim with standard tags.

5.2. Efficiency evaluation

The efficiency of parsing was evaluated through memory use and parsing time. Stanford parser requires a lot of memory. With 2 GB Java memory, only 174 of 273 claims can be directly parsed by the Stanford parser. The range of claim length is from 26 to 210. The ICSDS parser requires less memory than the Stanford parser, because its segmentation strategy reduces the maximum length of a claim in parsing. All 273 claims can be parsed under a computer with up to 1.4 GB Java memory.

To test the parsing time, 174 claims in the dataset that can be parsed with both ICSDS parser and Stanford parser were used. The comparison of parsing time is shown in Fig. 4.

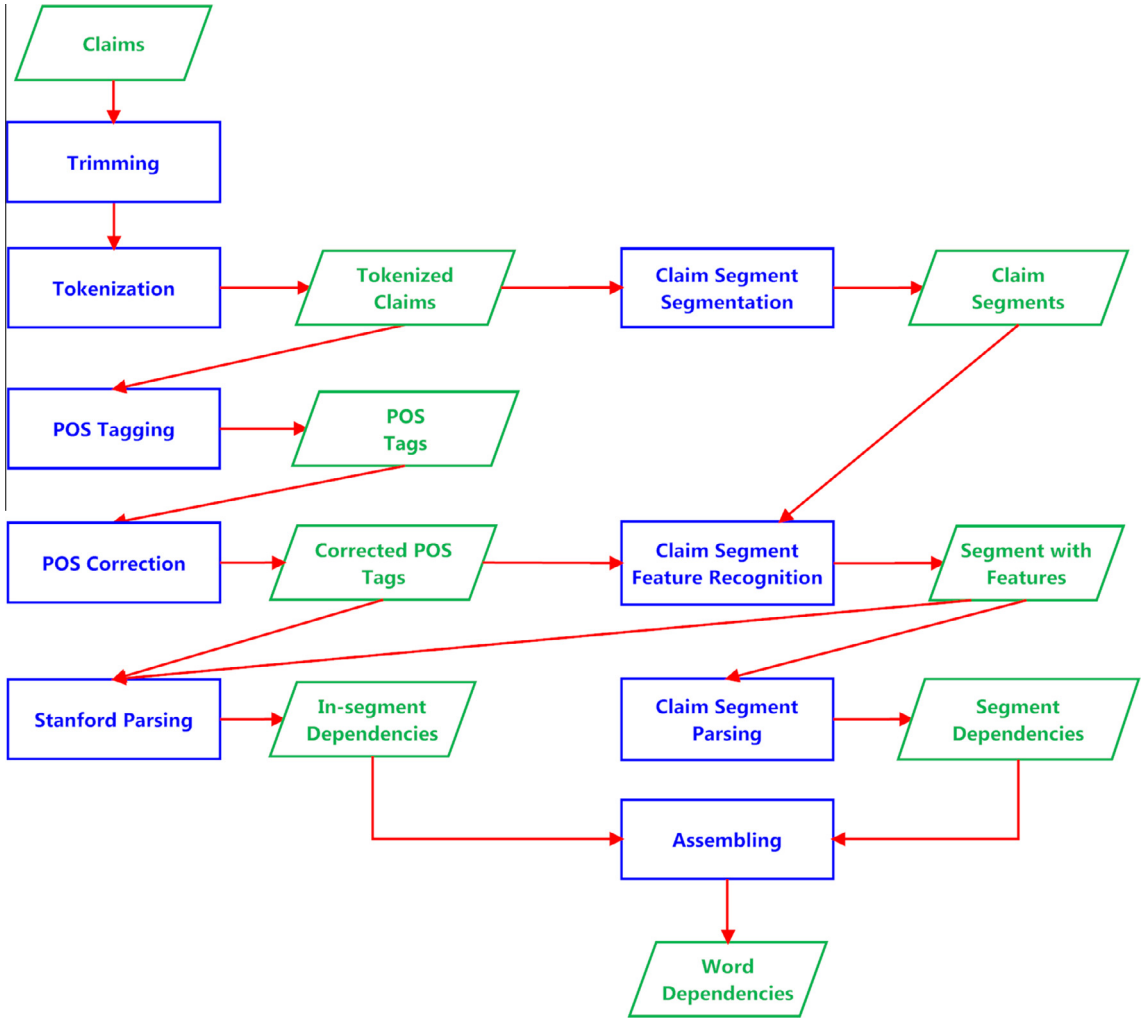

Fig. 2. Flowchart of the ICSDS-based parser.

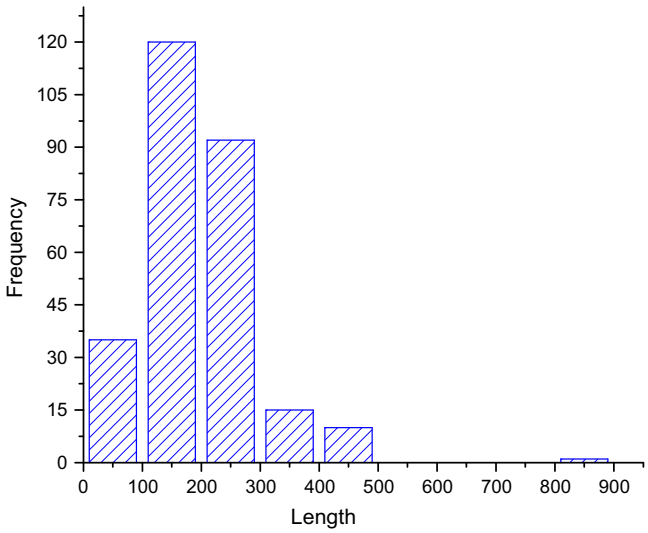

Fig. 3. The frequency of claim length.

It was observed that generally the parsing time of Stanford parser is monotonically increased with the increase of claim length. When the claim length is less than 50, the increase of parsing time is not significant. Parsing a claim with 50 tokens requires about five second. However, the parsing time increases sharply when the claim length is more than 100. Parsing a 140 long claim needs more than one minute; parsing a 170 long claim needs two minutes; while parsing a 200 long claim needs three minutes.

Apart from the shortest claim, ICSDS parser is faster than Stanford parser. Moreover, the variation of parsing time with ICSDS parser is small. The range of parsing time is from 1 to 31 s. The parsing time with ICSDS parser is almost independent from the claim length when the claim length is no more than 210.

5.3. Effectiveness evaluation

The effectiveness of parsing was tested on a structure model extraction problem. A structure model is a tree (graph model) that can describe the structure of a patented invention. In a structure model, as shown in Fig. 5, a root node is labelled with the name of the patented invention; a branch node is labelled with a unit name; and a leaf node is labelled with a part name.

For example, an independent claim consisting of four sentences (number in the second column) and 10 segments (number in the first column) is given below.

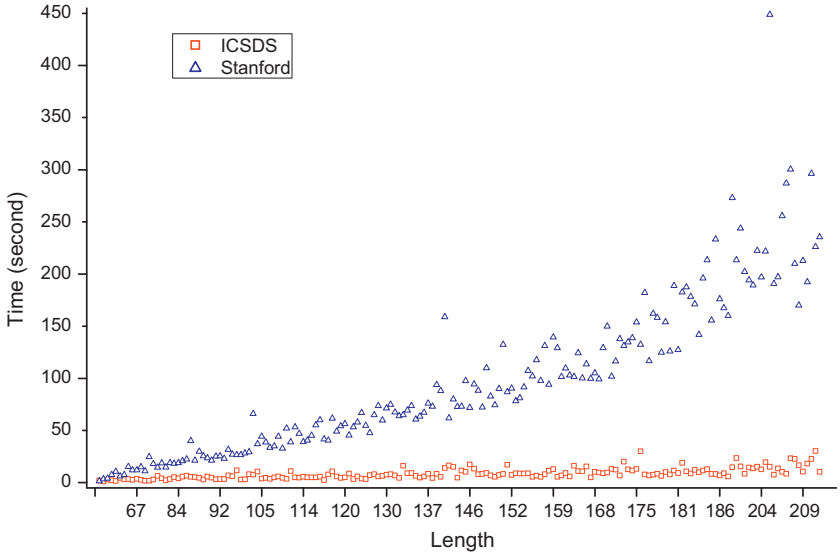

Fig. 4. Comparison of parsing time.

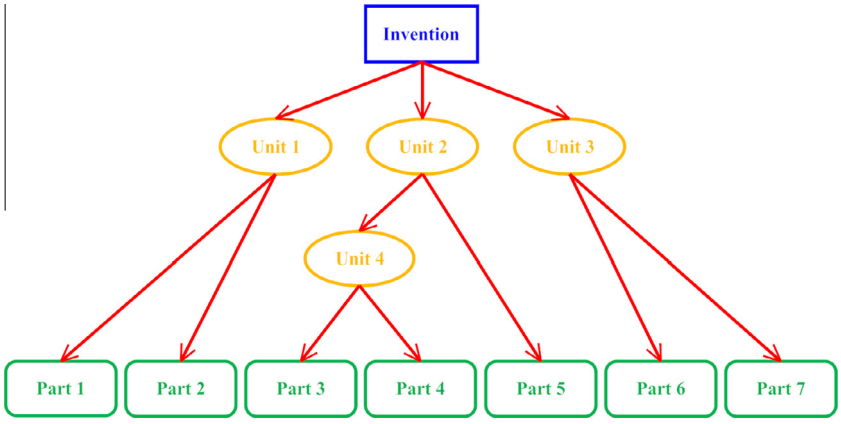

Fig. 5. Structure model.

01 1 *A mobile phone,*
02 1 *comprising:*
03 1 *a body having a ground portion;*
04 1 *a metallic cover detachably coupled to the body,*
05 2 *the metallic cover forming an exterior surface of the mobile phone;*
06 1 *and a grounding unit configured to electrically connect the ground portion of the body to the metallic cover when the metallic cover is coupled to the body,*
07 3 *the grounding unit being disposed on one of facing surfaces of the body and the metallic cover,*
08 4 *wherein the grounding unit includes:*
09 4 *an attachment portion located on an inner surface of the metallic cover facing the body;*
10 4 *and an elastic extension portion extending from the attachment portion towards the body.*

In the first sentence, a mobile phone (in Segment 1) comprises (in Segment 2) a body (in Segment 3), a metallic cover (in Segment 4) and a grounding unit (in Segment 6). The second sentence further elaborates the metallic cover (in Segment 5). The third sentence further elaborates the grounding unit (in Segment 7) and it includes (in Segment 8) an attachment portion (in Segment 9) and an elastic extension portion (in Segment 10). Thus, its structure model is shown in Fig. 6.

In the structure model extraction problem, an independent claim in the test set is parsed and a structure model is extracted from the parsing tree. The extracted structure model is compared with the standard structure model, which is previously manually

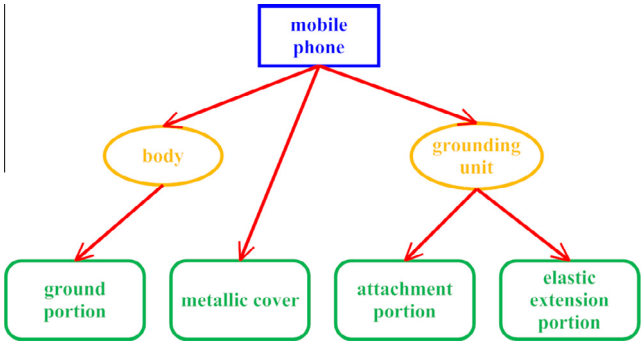

Fig. 6. An example of structure model.

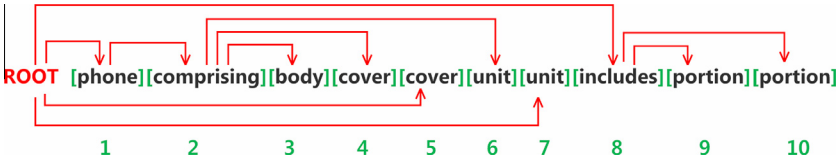

Fig. 7. An example of ICSDS parsing.

built. If two structure models are the same, then the parsing is considered as accurate.

The parsing result of ICSDS parser, where the word-to-word dependencies given by Stanford parser are omitted, is shown in Fig. 7. It can be observed that the core word in every segment is found correctly. For example, the core word in Segment 1 is "phone", which represents a mobile phone; while the core word in Segment 4 is "cover", which represents a metallic cover. Moreover, their dependency relations are also found correctly. For example, the phone comprises a body, a cover and a unit. The unit further comprises a portion and another portion. With this parsing result, the structure of patented invention can be easily described. This structure is just the major content of the claim. Therefore, the proposed parser had successful analysed the major content of the claim. In contrast, a conventional parser is hardly to figure out this structure and messes everything up due to LDD.

To evaluate the effectiveness, accurate rate is used as the evaluation measures. Accurate rate is the ratio of the number of accurate parsed claims to that of all parsed claims. Both Stanford parser and ICSDS parser were tested with the test set. The evaluation result showed that the accurate rate of Stanford parser was 14%, while the accurate rate of ICSDS parser was 68%. Although 68% is not very high, it is much higher than 14%, which is the accurate rate of Stanford parser. The improvement is close to 400%. The reasons of errors are various. Briefly, the rules discovered in the training set are incomplete to cover all situations.

6. Conclusions

In this study, six claim syntax peculiarities that increase the difficulty of parsing are highlighted. They are (1) claim template, (2) post attribute past participle, (3) parenthetical sentence, (4) complex noun phrase as sentence, (5) recursion, and, (6) coordination. These peculiarities cause long claims. Especially, the last four peculiarities lead to Long Distance Dependencies. A new two-level parser is proposed for patent claim parsing. It is designed to improve the adaptability of a conventional parser, e.g. Stanford parser. The conventional parser (in the first level) is evoked by a higher-level (in the second level) parser, which can handle the peculiarities of claim syntax. With respect to peculiarity (1), a trimming process is adopted to filter non-informative content. With respect to peculiarity (2), a POS correction process is adopted to change past form into past participle. With respect to last four peculiarities, a new dependency syntax called Independent Claim Segment Dependency Syntax (ICSDS) is proposed. To guarantee the efficiency of the proposed parser, a segmentation strategy is adopted. The segmentation and consequent assembly is executed by the ICSDS-based parser. The conventional parser is only evoked when processing each claim segment. Theoretically, the distance (in terms of segments) between two segments is much smaller than the distance (in terms of words) between two words within these two segments, respectively. Thus, the distance of the dependency becomes shorter and is easier to be captured.

The evaluation results show that ICSDS-based parser is much effective than Stanford parser. This is because its segmentation strategy. Moreover, ICSDS-based parser is much efficient than Stanford parser in a model extraction problem. This is because ICSDS-based parser can parse correctly more LDD relations.

To our best knowledge this work is a pioneer work in claim parsing. Compared to previous works, this study has shown an obvious improvement on effectiveness of patent claim parsing. With the ICSDS-based parser, patent claims can be automatically processed in a larger number and in a short time. This research contributes to patent users who are looking for technological details. For example, it saves time for product designers and allows them to focus on creative work. This research also contributes general patent users. It can support and enhance many patent-related applications, such as patent valuation, technology relatedness and competitor analysis, patent strategy development, and technology management.

In the future, ICSDS can be expanded by defining more dependency relationships between segments. Current ICSDS only focuses on verb–noun relation and adjective–noun relation. This is because these two relations are the most common and critical relations. However, for completeness, other relations such as preposition–noun, verb–preposition and adverb–verb should also be defined. Moreover, an evaluation with larger scale is expected to be carried out for testing the effectiveness and the efficiency of the enhanced ICSDS parser.

References

[1] A. Abbas, L. Zhang, S.U. Khan, A literature review on the state-of-the-art in patent analysis, World Pat. Inf. 37 (2014) 3–13.
[2] G. Fantoni, R. Apreda, F. Dell'Orletta, M. Monge, Automatic extraction of function-behaviour-state information from patents, Adv. Eng. Inform. 27 (2013) 317–334.
[3] T. Montecchi, D. Russo, Y. Liu, Searching in cooperative patent classification: comparison between keyword and concept-based search, Adv. Eng. Inform. 27 (2013) 335–345.
[4] S.-Y. Yang, V.-W. Soo, Extract conceptual graphs from plain texts in patent claims, Eng. Appl. Artif. Intell. 25 (2012) 874–887.
[5] S. Verberne, E. D'hondt, N. Oostdijk, Quantifying the challenges in parsing patent claims, in: the 1st International Workshop on Advances in Patent Information Retrieval (AsPIRe'10), March 2010, Milton Keynes, UK.
[6] C. Lee, B. Song, Y. Park, How to assess patent infringement risks: a semantic patent claim analysis using dependency relationships, Technol. Anal. Strategic Manage. 25 (1) (2013) 23–38.
[7] E.J. Walsh, H.J. DiPietrantonio, Getting the most value from patent claims, Intell. Property Technol. Law J. 23 (7) (2011) 9–12.
[8] R. Bekkers, R. Bongard, A. Nuvolari, An empirical study on the determinants of essential patent claims in compatibility standards, Res. Policy 40 (7) (2011) 1001–1015.
[9] Y. Xie, D.E. Giles, A survival analysis of the approval of US patent applications, Appl. Econ. 43 (11) (2011) 1375–1384.
[10] P. Parapatics, M. Dittenbach, Patent claim decomposition for improved information extraction, in: M. Lupu et al. (Eds.), Current Challenges in Patent Information Retrieval, Springer-Verlag, Berlin Heidelberg, 2011, pp. 197–216.
[11] S.-Y. Yang., V.-W. Soo, Comparing the conceptual graphs extracted from patent claims, in: The IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC 2008), June 2008, Taichung, Taiwan.
[12] S. Sheremetyeva, Natural language analysis of patent claims, in: The ACL-2003 Workshop on Patent Corpus Processing, July 2003, Sapporo, Japan.
[13] J. Nivre, Dependency Grammar and Dependency Parsing. Technical Report. Växjö University, 2005.
[14] M. Marneffe, B. MacCartney, C.D. Manning, Generating typed dependency parses from phrase structure parses, in: The Fifth International Conference on Language Resources and Evaluation (LREC 2006), May 2006, Genoa, Italy.

[15] Q.I. Wang, D. Lin, D. Schuurmans, Simple training of dependency parsers via structured boosting, in: The International Joint Conference on Artificial Intelligence (IJCAI-07), January 2007, Hyderabad, India.
[16] J. Eisner, three new probabilistic models for dependency parsing: an exploration, in: The 16th International Conference on Computational Linguistics (COLING 1996), August 1996, Center for Sprogteknologi, Copenhagen, Denmark.
[17] J. Nivre, M. Scholz, Deterministic dependency parsing of English text, in: The 20th international Conference on Computational Linguistics (COLING 2004), August 2004, University of Geneva, Switzerland.
[18] H. Yamada, Y. Matsumoto, Statistical dependency analysis with support vector machines, in: The International Conference on Parsing Technologies (IWPT 2003), April 2003, Nancy, France.
[19] Y. Zhang, S. Clark, A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search, in: The conference on Empirical Methods in Natural Language Processing (EMNLP 2008), October 2008, Waikiki, Honolulu, Hawaii, USA.
[20] J. Nivre, R. McDonald, Integrating graph-based and transition-based dependency parsers, in: The 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT 2008), June 2008, Columbus, Ohio, USA.
[21] K. Sagae, A. Lavie, Parser combination by reparsing, in: Human Language Technology Conference – North American chapter of the Association for Computational Linguistics annual meeting (HLT-NAACL 2006), June 2006, New York City, USA.
[22] A. Burga, J. Codina, G. Ferraro, H. Saggion, L. Wanner, The challenge of syntactic dependency parsing adaptation for the patent domain, in: The ESSLI 2013 Workshop on Extrinsic Parse Improvement (EPI), 5–16 August 2013, Düsseldorf, Germany.
[23] G. Ferraro, L. Wanner, Towards the derivation of verbal content relations from patent claims using deep syntactic structures, Knowl.-Based Syst. 24 (8) (2011) 1233–1244.
[24] G. Ferraro, H. Suominen, J. Nualart, Segmentation of patent claims for improving their readability, in: The 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR), April 2014, Gothenburg, Sweden, pp. 66–73.