Text Simplification of Patent Documents

Jeongwoo Kang(✉), Achille Souili, and Denis Cavallucci

CSIP/INSA Strasbourg, 24 Boulevard de la Victoire,
67084 Strasbourg Cedex, France
{jeongwoo.kang, achille.souili,
denis.cavallucci}@insa-strasbourg.fr

Abstract. This paper represents an automatic text simplification system for patent documents. The simplification system is embedded in the broader context of an information retrieval system which extracts IDM related knowledge from patent documents. Extracting elements of IDM ontology from patents involves training machine-learning model. However, an accuracy of the model is compromised when the given text is too long, hence the need of simplifying the texts to improve machine learning. There have been precedent studies on automatic text simplification based on hand-written rules or statistical approach. However, few researches addressed simplifying patent documents. Patent document has its particularity in its lengthy sentences and multiword expression terminology, which often hinder accurate parsing. Therefore, in this research, we present our method to automatically simplify texts of patent documents and scientific papers by analyzing their syntactic and lexical patterns.

Keywords: Inventive Design Method · TRIZ · Information extraction
Text simplification · Syntactic analysis · Text mining

1 Introduction

This paper represents an automatic text simplification system for patent documents. The simplification system is embedded in the broader context of an information retrieval system which extracts IDM (Inventive Design Method) related knowledge from patent documents. Patent documents are important source of information which contains a history of the evolution of the artifacts [1]. By looking into patents, one can learn progress of technologies over the course of history, and more importantly, technological challenges and their solutions that were invented by specialists and engineers in the field. In the light of the importance of patents as information resource, there have been a number of academic researches and activities in patent mining in recent years.

The research of Souili [1], which inspired our work of research, applied text mining approach for information retrieval from patents. He analyzed patent documents and automated extraction of IDM related knowledge from patents. The Inventive Design Method presents a framework for inventive problem solving process. The IDM was created by Cavallucci and Khomenko [2] and it was inspired by the Theory of Solving Inventive Problem (TRIZ), a theory invented by Altshuller [3]. Therefore, it is worth looking into TRIZ before we discuss the IDM in further detail.

TRIZ is “a problem-solving, analysis and forecasting tool derived from the study of patterns of invention in the global patent literature” [4]. In the research of Altshuller [3], he suggested a systematic methodology to find inventive solutions to technological problems. According to Altshuller [3], there are generalizable patterns in the nature of inventive solutions and problems. These patterns can be applied across industries and sciences, therefore, by defining principles of technological challenges and inventive solutions, one can reuse this strategy to solve a similar problem in different context. Altshuller [3] also argued that a number of technical problems which necessitate inventive solutions stem from technical contradiction. Accordingly, defining these contradictory situations and the way to solve the contradiction are dealt as a key factor in TRIZ methodology.

IDM was created to complement the limits of TRIZ. Classical TRIZ methods are not easy to comprehend due to a lack of formalized ontology, nor simple to make a computation design model upon its concepts [5]. Therefore, IDM presented its formalized ontology to formulate relevant ideas and concepts for inventive design. IDM ontology mainly consists of three concepts: problem, partial solution, and parameter. Parameters consist of two sub-categories, which are evaluation parameters and action parameters. Problems often refer to a contradictory situation where two features conflict with one another, and therefore, ameliorating one feature deteriorates the other. Solution is a way to solve this contradiction without compromising any of the features. On the other hand, a partial solution is an incomplete solution and while it provides a solution to one problem, it may bring about another problem. Action parameter is what a designer can make a design choice on, and evaluation parameter represents a value with which one can evaluate the result of the choice of a designer [6].

PatExtractor is an information extraction system for patents, into which the simplification system presented in this paper is integrated. PatExtractor automatically retrieves IDM knowledge from patents with its machine learning algorithm. However, the accuracy of the system is compromised when the system analyses lengthy sentences, which are fairly common in patent documents. Overlong sentences in patents make it difficult for the machine learning algorithm to classify the texts accurately and extract relevant information from them. This is because the algorithm analyzes a sentence based on the words composing the sentence, and the algorithm can be distracted when there are too many words to analyze. It is even more so when the words in a sentence contain conflicting information. Moreover, a syntactically-complex sentence, which consists of more than one predicate, often has more than one piece of information to extract. For example, a sentence composed of a few predicates that are joined by concessive or contrastive conjunctions such as *yet*, *but*, *although*, *though* or an adverb *however* sometimes includes a problem and a partial solution at the same time in different predicates. Therefore, it is necessary to split this sentence into smaller pieces so that the classifier identifies each part of information and label them properly.

To handle the above-mentioned sentences and to improve accuracy of the classifier by doing so, this research contributes to making an automatic sentence simplification system. This simplification system detects complex sentences and simplify those sentences by splitting, dropping and modifying. This research addresses simplification on two levels: coarse simplification and fine simplification. The coarse simplification system is designed to improve the performance of the machine learning algorithm as mentioned above. The fine simplification system is, on the other hand, designed to improve the readability of the extracted data. PatExtractor not only extracts IDM knowledge from patents but also visualizes the results using an ontology-based graphic. As its name suggests, the fine simplification system simplifies given texts in more elaborate way after the coarse simplification. Further details on the two simplification systems will be discussed in methodology section. In the next section, we review precedent researches on automatic text simplification.

2 Literature Review

There have been a number of precedent researches on text simplification. Text simplification is a task that "aims to rewrite a sentence so as to reduce its complexity, while preserving its meaning and grammaticality" [7]. Complexity of sentences stem from either syntactic complexity or semantic complexity [8]. As its name suggests, syntactic complexity is derived from a syntactic structure of a sentence. Syntactically complex sentences consist of more than one predicate. These predicates are joined by conjunctions such as *but*, *and*, *for*, *until* or relative pronouns such as *who*, *which*, *that*, *whose* or relative adverbs such as *when* and *where*. Therefore, this syntactic complexity can be resolved by splitting a complex sentence into shorter phrases and then restructuring each phrase to complete it. Sentence 1 is an example of a syntactically complex sentence and Sentence 3 shows how its complexity is reduced by splitting. Semantic complexity, on the other hand, is derived from difficult vocabulary. Thus, this complexity can be reduced by paraphrasing, more particularly, replacing difficult words to understand with easier and more understandable ones. Sentence 4 is an example of semantically complex sentence and Sentence 5 shows how its complexity can be diminished by paraphrasing.

Sentence 1: I have a friend *whose* cat is annoying.
Sentence 2: I have a friend. Whose cat is annoying.
Sentence 3: I have a friend. *His* cat is annoying.
Sentence 4: The workers *acquiesced* to their boss' request.
Sentence 5: The workers *accepted* their boss' request.

Text simplification is mainly used in two contexts. First, it is used to help people with low-literacy or reading disability such as foreigners, kids or people suffering from dyslexia [8, 9]. In other context, it serves as a preprocessing tool for natural language processing (NLP) tasks [10]. Simplified texts are easier to parse or translate, thus text simplifications are often integrated into automatic text summarization [11] or machine translation system [12] as a preprocessing tool. Moreover, the desired results could vary depending on the context in which the simplification is used. For example, when text simplification is used as a supporting tool to aid people with low-literacy, both syntactic and semantic complexity should be reduced so that the texts are more readable and understandable for target readers. However, when text simplification is used to facilitate natural language processing, simple syntactic simplification could be enough to fulfil the objectives, depending on the context of its application.

There have been different approaches to achieve text simplification. At the early stage, most of researches were based on hand-crafted rules [10, 13, 14]. In most rule-based approaches, they use POS tagger or syntax parser to detect complex sentences, and then split these complex sentences into smaller pieces using predesigned rules [8, 12]. This step is often followed by phrase-reordering or restructuring. As shown above in Sentence 1 to 3, a syntactically complex sentence (Sentence 1) is firstly identified as a sentence to simplify by syntactic parsing, and then split into two or more phrases (Sentence 2) and finally, restructured (Sentence 3). To restructure the phrases, co-reference relations and grammar are taken into consideration and dependency parser is often used to achieve this. Dependency parser represents the words of a given sentence in its dependency relation with other words, so it enables to capture co-reference relations in a sentence, such as a relation between a relative noun (*whose* in Sentence 1) and its antecedent (*friend* in Sentence 1). However, this rule-based approach requires linguistic knowledge to craft the rules, and it is difficult to cover all the complex sentences of different structures with these rules.

To complement the limits of rule-based approaches, academic society shifted their focus to data driven approaches [7]. Some of these approaches use statistical machine translation system to achieve simplification [15, 16] and they are known to produce accurate results. This approach considers text simplification as a monolingual translation task. More particularly, they consider complex sentences as source texts and simplified sentences as target texts. These approaches use parallel corpus, which consists of original texts aligned with their simplified texts, to train their simplification model [17, 18]. This type of data is easily accessible on Simple English Wikipedia[1]. By aligning texts of Wikipedia and Simple English Wikipedia, one can obtain a parallel corpus needed to train their statistical text simplification model [15, 16]. To train this statistical simplification system, texts, syntactic parse trees or semantic parse trees [16] are often used as input data. This data driven method yields good results for lexical substitution and deletion, but is less effective for sentence splitting and sentence reordering [19, 20]. Moreover, parallel corpora to train this model are available for few languages.

Patents simplification has been studied in a few researches as a sub-category of general text simplification. Patent documents are known for its notoriously poor readability, and it is especially more so in the claims part. Poor legibility of patents stem from their abstract vocabulary, overlong sentences [17] and terminology. In patents, it is not uncommon to see a sentence with 200 words and there are even sentences with more than 500 words [1]. To make those documents more readable and comprehensible for readers, there have been a few, yet not many, researches on patent simplification [17, 18, 21]. In most researches on patents simplification, rule-based approaches are preferred to statistical approaches. This is not only because there are few parallel corpus on patent documents, but also because texts in patents have specific linguistic features such as long distance anaphoric references and repetitiveness [17]. Accordingly, hand-crafted rules which take these linguistic features of patents into consideration are often used for patent simplification tasks.

[1] It is an edition of Wikipedia geared towards students, children or adults with learning difficulties. It is therefore written in basic and easy English.

3 Our Methodology

3.1 PatExtractor and IDM Ontology

Our simplification system is integrated into an information extraction system, PatExtractor. PatExtractor takes patent documents (unstructured data) as input data, and extracts concepts of IDM ontology from the documents with NLP approach. As output, it creates a graphic (structured data), on which each node contains problem, partial solution or parameter.

In general contexts, ontology is an ensemble of relevant concepts in a specialized domain where each concept is interrelated to another. On the contrary, IDM ontology is a generic one and is not limited to a specialized domain, which makes it applicable across domains. IDM ontology is composed of concepts that are useful to formulate problematic situation and its solution during inventive design. Our simplification system is integrated to PatExtractor in order to aid the system with extracting main concepts of IDM ontology – problem, partial solution, action parameter and evaluation parameter – by simplifying the texts and thereby facilitating the information extraction process.

3.2 Text Simplification

Overall, our methodology for text simplification follows the precedent rule-based approaches. However, our method is distinguished from the previous methods in two aspects. First, we don't apply lexical simplification since our texts to simplify are patents and most of terminology in patent documents cannot be replaced with another lexicon. Moreover, our target readers are specialists who seek specialized knowledge from patents, thus we have to keep the precise information that is contained in the original terminology. Secondly, our simplification system consists of two different levels of simplification. Our system is integrated into PatExtractor, an information extraction system for patents, therefore, it should fit the objectives of PatExtractor. Integration of the simplification systems to PatExtractor is as shown in Fig. 1.

PatExtractor necessitates two levels of simplification for different goals. First simplification is to facilitate machine learning algorithm's identifying target concepts – concepts of IDM ontology – from texts. Thus, the objective of the simplification is to split sentences so that each phrase contains one idea, not several. Second simplification is designed to improve readability of the graph drawn with extracted information. PatExtractor identifies problems, partial solutions, and parameters, from a given text and then visualizes them as a graph[2]. Each node of the graph contains extracted knowledge. However, when the contents of each node are too wordy, it decreases readability of the graph, hence the necessity of simplifying texts. Desired outputs for each level of simplification will further be discussed in the following sections.

[2] The graph is displayed on Finder (https://finder.inventivedesign.unistra.fr), a web application which visualizes extracted information with a graph.

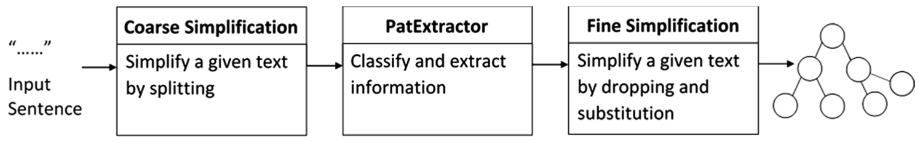
"……"
Input Sentence
Coarse Simplification
Simplify a given text by splitting
PatExtractor
Classify and extract information
Fine Simplification
Simplify a given text by dropping and substitution

Fig. 1. Integration of the simplification systems to PatExtractor

Before we look into different methodologies for two levels of simplifications, we address a general approach and tools that were used throughout our experiment as well as a development environment for our system. First, we adopted a rule-based approach to simplify texts. Based on our analysis on patent documents, we determined that not every complex sentence is our target for simplification. Texts need to be split only when the splitting helps the classifier to label them correctly. Otherwise, it can hurt the performance of the classifier. Accordingly, a rule-based approach with which we could specify our own rules for simplification was adopted.

Furthermore, we used a syntactic parser to identify a subordinate phrase and split the subordinate off from a matrix phrase. Among several syntactic parsers, we adopted the BLLIP parser [22] given its high accuracy and fast speed[3]. The BLLIP parser is a statistical natural language parser which creates several candidate syntax parses of a given sentence and ranks n-best among them. Another advantage of using the BLLIP parser is that it supports NLTK module [23], so one can easily manipulate syntactic trees created by the BLLIP parser. Thus, in our research, the BLLIP parser and NLTK modules are mainly used tools for simplification task. Lastly, our system is written in Python 3.5 and operates on Linux.

Coarse Simplification System. Our goal at the coarse simplification level is basically to split subordinate phrases from a matrix phrase so that information extraction algorithm treats shorter phrases and yields more accurate results. More specifically, our coarse system splits subordinate phrases that begin with a subordinate conjunction such as *if, until, so that*, *because* or relative adverbs such *as, when, where*, *why*, but it doesn't split relative clauses, which begin with a relative pronoun such as *who, that, which*. Relative clauses are handled at the fine simplification level, which is placed after classification task. Let's take an example of Sentence 1 to be clearer. When Sentence 1 is given as input, we expect to have two split phrases Sentence 2 and Sentence 3 as output after the coarse simplification.

Sentence 1: **If** there is any unmatched state, the signal A turns to H.
Sentence 2: If there is any unmatched state,
Sentence 3: the signal A turns to H.
Sentence 4: It is *advantageous* **if** all the permanent magnets are surrounded.

[3] According to the authors' experiment, compared to Standford Parser, which is one of the most commonly used parsers for syntactic analysis, BLLIPparser functions more accurately when analyzing sentences with multi-lexical terms, which are common in patents. As for parsing speed, BLLIPparser is slightly faster than Stanford Parser. NLTK syntax parser was not considered for our task since it requires users to write their own rules.

Sentence 5: A known *problem* with DC currents is that **when** opening a switch by separating the switch contacts a spark builds between the contacts.

To achieve this, our simplification system functions as shown in Table 1. First, for a given sentence, the system checks if the sentence contains a subordinate conjunction or a relative adverb with part of speech (POS) tagger. If the sentence contains these elements, it is analyzed by syntactic parser to create a syntax tree. The syntax tree is used not only to identify a scope of subordinate phrase, but also to double-check if it truly is our target structure. In this step, we filter out unwanted structures.

Table 1. Operation of coarse simplification system

Coarse simplification system

```
Identify complex sentences
If given text is a complex sentence
     ➔ Parse a complex sentence to obtain a syntactic tree
       If given syntactic tree is our target structure
            ➔ Split off subordinates from a matrix phrase
            ➔ Clean the result sentence
```

Let us recall that the goal of our coarse simplification is to aid the classifier in identifying problems, partial solutions and parameters from a given text. In case of Sentence 4 and 5, a phrase located before *if* and *when* can be an indicator telling the class of the following phrase. For example, the phrase '*It is advantageous*'[4] in Sentence 4 indicates that the following phrase has a positive connotation, and therefore, can be a partial solution. Likewise, the phrase '*a known problem with DC currents is that*' in Sentence 5 gives information that the following phrase may contain a problem. Therefore, splitting these phrases from its subordinates can hurt the performance of the classifier. To avoid this, we rule out sentences containing some keywords such as *problem*, *solution*, *appreciated, advantageous* off the list of sentences to simplify. We use the same, but filtered, keywords list that is used to extract problems from patent documents.

Once the given structure is identified as our target structure after the filtering, we split subordinate phrases, which is represented as SBAR on a syntactic tree, from a matrix phrase. As a result, a sentence is divided into two or more phrases. As a last step, we clean the simplified sentence. During the syntax parsing, a sentence is tokenized, which means that words are segmented and then some punctuation and symbol

[4] This is a common sentence structure in patents and they indicate that the current sentence includes a solution (or a problem). This type of sentences has a structure of '*It is* **[adjective]** *if/when/that* **[subordinate phrase]**.' As for adjectives, there are certain adjectives or past participles that are commonly used, such as *beneficial, advantageous, recommended, appreciated, accepted*.

characters, such as quotation marks and parenthesis, are replaced with different signs. Moreover, there can be left-out phrases or words during the simplification. Therefore, the system normalizes the modification made during the syntax parsing and simplification, and joins the left-out part to an appropriate location of the split sentence.

Fine Simplification. The objective of fine simplification system is improving a readability of a graph drawn by a web application Finder, which enables a visual representation of extracted information from patents. This can be accomplished by dropping supplementary information such as conjunctions (such as *if, when, although*), connective adverbs (such as *then, therefore*, *however, by contrast, that is to say*) and supplementary subordinates. Let us take an example of Sentence 5 to clarify the desired result of the fine simplification. When a sentence 5 is given, it is firstly split into two phrases as Sentence 6 and 7 by the coarse simplification system. Then conjunction 'if' in Sentence 6 and 'then' in Sentence 7 are deleted to make each of them a complete sentence. Afterwards, relative pronoun phrase '*which could interrupt the system*' of Sentence 7 is deleted. The last two simplification steps – dropping a conjunction and connective adverb; deleting a relative subordinate – are done by the fine simplification system. As a final output of two simplification systems, we obtain two sentences, Sentence 8 and 9.

Sentence 5: **If** this offset varies from test strip to test trip, **then** noise *which could interrupt the system* is added to the measurement.
Sentence 6: **If** this offset varies from test strip to test trip,
Sentence 7: **then** noise which could interrupt the system is added to the measurement.
Sentence 8: This offset varies from test strip to test trip.
Sentence 9: Noise is added to the measurement.

The general method of the fine simplification is similar to that of the coarse simplification in the way that it exploits syntactic tree to find subordinate phrases. In the fine simplification system, however, deletion is applied instead of simple split. Moreover, only certain types of relative subordinate clauses that are followed by *who*, *that*, *which, whose* are subject to be treated. Generally, relative pronoun subordinate carries supplementary information of its antecedent. This supplementary information does not affect a principle message of a sentence even when deleted. In Sentence 5, the relative clause '*which could interrupt the system*' carries additional information of its antecedent '*noise*.' However, even when it is deleted, it does not change the main idea, that is, a problematic situation to the stated system caused by added noise. Let us recall the final output of the entire system is a visual representation of extracted information, which, thus, should be concise and clear. Therefore, dropping additional information and conveying only principal ideas is the objective of the fine simplification system. Furthermore, original sentence before simplification is also provided to users on Finder, so that users can have access to the fuller information. This is to prevent information loss or distortion which could occur during the simplification process.

Table 2. Fine simplification system

```
Fine simplification system
Drop connective adverbs and conjunctions
Identify sentences containing a relative subordinate
If given text has a relative subordinate clause
    ➔ Parse the sentence to obtain a syntactic tree
      If given syntactic tree is our target structure
          ➔ Delete subordinates from a matrix phrase
          ➔ Clean the result sentence
```

Table 2 shows how the fine simplification system functions. Overall, it roughly follows the same process as the coarse simplification. However, one difference is dropping connective adverbs and conjunctions at the beginning of the system. Let us recall the fine simplification system takes place after the coarse simplification. That is, some sentences are already split into more than two phrases such as in Sentence 6 and 7. In this case, we need to edit these sentences to make them complete. For example, in case of Sentence 6, it is grammatically not complete because of conjunction 'if.' Thus, we drop this conjunction to make it a complete sentence. Moreover, sentence 7 does not need its connective adverb 'then' anymore since it does not have the preceding phrase. Therefore, we drop this adverb.

Dropping connective adverbs (not conjunctions) are applied not only to complex sentences which were split by the coarse simplification system, but also to the simple sentences which were not treated by the coarse simplification system. Connective adverbs (or phrases) such as *however, by contrast, that is to say, in addition, moreover* present the semantic relation with its preceding sentence. However, our information extraction functions on a sentence unit, which means that each sentence extracted by the system is represented without its context – its preceding and following sentence. Thus, connective adverbs which show the links between a current phrase and its preceding or following sentence are dropped to keep the extracted information simple.

4 Discussion

The evaluation on our simplification systems is twofold given that each level of simplification has different objectives. We integrate the coarse simplification system to the information extraction system, PatExtractor. Afterwards, we evaluate the simplification system by evaluating the performance of classification algorithm. We compare the performance of the algorithm before and after the integration of the coarse simplification system. On the other hand, the fine simplification is evaluated by readability score of the result sentences since the fine simplification system is designed to improve legibility of a graph, on which extracted information is displayed.

Evaluation on the Coarse Simplification System. One of the most commonly used measures to evaluate a classification algorithm are precision and recall. Recall and precision score of PatExtractor V 2.0 for partial solution and problem is as shown in the Table 3. To evaluate the system, we randomly chose 250 patent documents with different topics and extracted information from them. During our experiment, however, we did not observe a significant improvement on precision score after the integration of the coarse simplification system. The precision remained roughly the same (its improvement rate was less than 0.1%) and recall was not measured because given the structure of the entire system, our simplification does not affect the recall of the algorithm. Nevertheless, more interesting observation could have been possible if more test data, especially annotated data were available. Given the fact that only a small number of phrases are selected to classify and simplify from one patent text, the phrases that were finally used for evaluation (from 250 corpora) were not large in number. Furthermore, for lack of annotated data, the evaluation was largely based on the author's judgement.

Table 3. Recall and precision of PatExtractor V 2.0

	Recall	Precision
Partial solution	47.87%	74.99%
Problem	43.83%	88.14%

Evaluation on the Fine Simplification System. To evaluate the fine simplification system, we used Flesch Reading Ease score [24]. It is one of the most commonly used scoring formula to evaluate readability of a text. The scale of the score is from 1 to 100 and the higher the score is, the more readable the text is. To evaluate the fine simplification system, we chose three sets of corpus: copurs1 consisting of 10 patent documents treating a topic in automatic transfer switch; corpus2 consisting of 10 documents covering a topic in automobile; corpus3 consisting of 10 documents addressing a topic of batteries. The score reflects not only the result of the fine simplification system but also that of the coarse simplification system. This is because the fine simplification takes place consecutively after the coarse simplification step where complex sentences are firstly split into shorter phrases. As shown in the Table 4, after applying the fine simplification system, we obtained higher readability score by 0.8 point with Corpus1, which is equal to 2.1% of improvement rate. With corpus 2, we obtained 1.6 higher readability score after the simplification, that is, 4.83% of the improvement rate. With corpus3, we obtained higher readability score by 1.3 point, which is equal to 3.5% of improvement rate. The average improvement rate is therefore 3.37%.

Table 4. Readability score before and after simplification

	Before simplification	After simplification	Improvement rate
Corpus1	37.1	37.9	2.1%
Corpus2	33.1	34.7	4.83%
Corpus3	36.7	38.0	3.5%
Average	35.6	36.8	3.37%

Let us take examples of sentences produced by PatExtractor v.3.0 to see the result in detail. Sentence 10 is produced by the previous version of PatExtractor, that is, version 2.0. which does not have the text simplification systems. The former part of the phrase '*if the water-absorbing ability exceeds over 800 g/g*' is a cause of a problem and the latter '*the paste may be gelled and become difficult to be uniformly coated on the conductive substrate*' is a resulted problem. Among these two ideas, only the latter is what our focus goes on since it concerns a problem, which is one of the IDM ontology elements we would like to extract. The previous version of PatExtractor extracts the full sentence and labeled the entire sentence as a problem. On the contrary, PatExtractor v.3.0 with the text simplification systems integrated extracts only the latter part and label it as a problem separately from its former phrase (Sentence 11). This is different from the previous version of PatExtractor which treats the entire sentence as a whole.

Sentence 12 is another example of extracted information by the previous version of PatExtracotr. Sentence 13 and 14 are extracted by the current PatExtractor v.3.0. Sentence 12 delivers two main ideas: *magnetic flux control means are controlled individually; the flow of the magnetic flux can be controlled more flexibly*. Both information contains important concepts which correspond to partial solution of an IDM ontology. The updated version of PatExtractor treats these two concepts separately (classifiy separately) and returns two results. This makes the final graph of extracted information more concise. However, as we discussed above, we observed that the integration of simplification systems does not improve the accuracy of the classification significantly. Sentence 12 corresponds to a partial solution but is labeled as a problem by PatExtractor v.2.0. This incorrect labeling remains unchanged in PatExtractort v.3.0 as seen in Sentence 13 and 14.

Sentence 10: *If the water-absorbing ability exceeds over 800 g/g*, the paste may be gelled and become difficult to be uniformly coated on the conductive substrate. **[problem]**

Sentence 11: The paste may be gelled and become difficult to be uniformly coated on the conductive substrate. **[problem]**

Sentence 12: When the magnetic flux control means are controlled individually, the flow of the magnetic flux can be controlled more flexibly. **[problem]**

Sentence 13: The magnetic flux control means are controlled individually. [**problem**]

Sentence 14: The flow of the magnetic flux can be controlled more flexibly. [**problem**]

5 Conclusion and Future Work

Even though integration of the simplification systems did not lead to a significant improvement of the classification algorithm, it improved a readability of a graph created from extracted information to a certain degree. Furthermore, deleting supplementary information made it possible to extract information with focusing only on important part of information, which corresponds to elements of the IDM ontology. Moreover, simplification system enabled PatExtractor to treat different concepts and ideas in one sentence separately, contrary to the previous version which treated them as a whole. For the future work, phrase restructuring function could be added to the simplification system since, while splitting off sentences into several phrases, anaphoric relations between a pronoun and its antecedent could be lost. Moreover, it would be interesting to evaluate the precision of the upgraded version of PatExtractor with more corpus that are annotated by experts.

References

1. Souili, W.M.A.: Contribution à la méthode de conception inventive par l'extraction automatique de connaissances des textes de brevets d'invention (2015)
2. Cavallucci, D., Khomenko, N.: From TRIZ to OTSM-TRIZ: addressing complexity challenges in inventive design. Int. J. Prod. Dev. **4**, 4–21 (2006)
3. Altshuller, G.: And suddenly the inventor appeared: TRIZ, the theory of inventive problem solving. Technical Innovation Center, Inc. (1996)
4. Hua, Z., Yang, J., Coulibaly, S., Zhang, B.: Integration TRIZ with problem-solving tools: a literature review from 1995 to 2006. Int. J. Bus. Innov. Res. **1**, 111–128 (2006)
5. Souili, A., Cavallucci, D.: Toward an automatic extraction of IDM concepts from patents. In: Chakrabarti, A. (ed.) CIRP Design 2012, pp. 115–124. Springer, Heidelberg (2013). https://doi.org/10.1007/978-1-4471-4507-3_12
6. Cavallucci, D., Rousselot, F.: Structuring Knowledge Use in Inventive Design. Springer, New York (2007)
7. Lee, J., Don, J.B.K.P.: Splitting complex English sentences. In: Proceedings of the 15th International Conference on Parsing Technologies, pp. 50–55 (2017)
8. Carroll, J., Minnen, G., Pearce, D., Canning, Y., Devlin, S., Tait, J.: Simplifying text for language-impaired readers. In: Ninth Conference of the European Chapter of the Association for Computational Linguistics (1999)
9. Inui, K., Fujita, A., Takahashi, T., Iida, R., Iwakura, T.: Text simplification for reading assistance: a project note. In: Proceedings of the Second International Workshop on Paraphrasing, vol. 16, pp. 9–16. Association for Computational Linguistics (2003)
10. Chandrasekar, R., Doran, C., Srinivas, B.: Motivations and methods for text simplification. In: Proceedings of the 16th Conference on Computational Linguistics, vol. 2, pp. 1041–1044. Association for Computational Linguistics (1996)
11. Knight, K., Marcu, D.: Statistics-based summarization-step one: sentence compression. In: AAAI/IAAI, pp. 703–710 (2000)
12. Poornima, C., Dhanalakshmi, V., Anand, K.M., Soman, K.P.: Rule based sentence simplification for english to tamil machine translation system. Int. J. Comput. Appl. **25**, 38–42 (2011)

13. Siddharthan, A.: An architecture for a text simplification system. In: 2002 Proceedings of Language Engineering Conference, pp. 64–71. IEEE (2002)
14. Siddharthan, A.: Text simplification using typed dependencies: a comparison of the robustness of different generation strategies. In: Proceedings of the 13th European Workshop on Natural Language Generation, pp. 2–11. Association for Computational Linguistics (2011)
15. Zhu, Z., Bernhard, D., Gurevych, I.: A monolingual tree-based translation model for sentence simplification. In: Proceedings of the 23rd International Conference on Computational Linguistics, pp. 1353–1361. Association for Computational Linguistics (2010)
16. Narayan, S., Gardent, C.: Hybrid simplification using deep semantics and machine translation. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (vol. 1: Long Papers), pp. 435–445 (2014)
17. Mille, S., Wanner, L.: Making text resources accessible to the reader: the case of patent claims. In: LREC (2008)
18. Sheremetyeva, S.: Automatic text simplification for handling intellectual property (the case of multiple patent claims). In: Proceedings of the Workshop on Automatic Text Simplification-Methods and Applications in the Multilingual Society (ATS-MA 2014), pp. 41–52 (2014)
19. Bott, S., Saggion, H., Figueroa, D.: A hybrid system for spanish text simplification. In: Proceedings of the Third Workshop on Speech and Language Processing for Assistive Technologies, pp. 75–84. Association for Computational Linguistics (2012)
20. Siddharthan, A.: A survey of research on text simplification. ITL-Int. J. Appl. Linguist. **165**, 259–298 (2014)
21. Shinmori, A., Okumura, M., Marukawa, Y., Iwayama, M.: Patent claim processing for readability: structure analysis and term explanation. In: Proceedings of the ACL-2003 Workshop on Patent Corpus Processing, vol. 20, pp. 56–65. Association for Computational Linguistics (2003)
22. Charniak, E., Johnson, M.: Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pp. 173–180. Association for Computational Linguistics (2005)
23. Bird, S., Loper, E.: NLTK: the natural language toolkit. In: Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions, p. 31. Association for Computational Linguistics (2004)
24. Kincaid, J.P., Fishburne Jr., R.P., Rogers, R.L., Chissom, B.S.: Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel (1975)