# Methods of Statistical and Semantic Patent Analysis

Dmitriy Korobkin(✉), Sergey Fomenkov, Alla Kravets, and Sergey Kolesnikov

Volgograd State Technical University, Volgograd, Russia
dkorobkin80@mail.ru

**Abstract.** In the paper, authors proposed a methodology to solve the problem of prior art patent search, consists of a statistical and semantic analysis of patent documents, machine translation of patent application and calculation of semantic similarity between application and patents. The paper considers different variants of statistical analysis based on LDA method. On the step of the semantic analysis, authors applied a new method for building a semantic network on the base of Meaning-Text Theory. Prior art search also needs pre-translation of the patent application using machine translation tools. On the step of semantic similarity calculation, we compare the semantic trees for application and patent claims. We developed an automated system for the patent examination task, which is designed to reduce the time that an expert spends for the prior-art search and is adopted to deal with a large amount of patent information.

**Keywords:** Prior-art patent search · Patent examination · LDA · Semantic analysis · Natural language processing · Big data

## 1 Introduction

From year to year, the number of patent applications is increasing. Around 2.9 million patent applications were filed worldwide in 2015, up 7.8% from 2014. The escalating applications flow and more than 20 million World set of granted patents (from 1980 to 2015) increase the time that patent examiners have to spend to examine all incoming applications. Sometimes examiner has to make hundreds of search queries and to process thousands of existing patents manually during the examination procedure to make a decision: to approve the application or to reject it. The increasing workload of patent offices led to need for developing new approaches for patents prior-art retrieval on the base of statistical and semantic methods of natural language processing.

Among the most common commercial products in this area we can highlight services such as Thomson Reuters (Thomson Innovation), Questel (Orbit), GridLogics (PatSeer), VantagePoint, STN Analyze Plus, STN Anavist, Invention Machine (Knowlegist, Goldfire), etc. as well as many additional toolkits: Metheo Patent, TEMIS, TotalPatent, Wisdomain, PatBase, ArchPatent, PatentLens, PatentBuddy, PatentTools, FreePatentsOnline, Intellogist, PriorSmart, MaxVal, BizInt SmartCharts, Espacenet, AmberScope, Acclaim IP, Innography, IFI Claims, PatentInspiration. However, all the


A. Kravets et al. (Eds.): CIT&DS 2017, CCIS 754, pp. 48–61, 2017.
DOI: 10.1007/978-3-319-65551-2_4

above-mentioned products are conducting a search of the documents relevant to the application according to the request made by experts. So, they are not the direct counterparts of the developed system.

Many scientists tried to solve patent prior-art search task. The main research in patent retrieval started with annual tracks CLEF-IP 2009-2011, which were created to compare different approaches in different tasks related to the patent applications examination process, including prior-art search task. Magdy used an approach based on unigrams and bigrams [1], Verma's approach is based on keyphrase and citations extraction [2], Mahdabi used method based on a time-aware random walk on a weighted network of patent citations [3], Xue's approach considers an actual query as the basic unit and thus captures important query-level dependencies between words and phrases [4], D'hondt tried to compare flat classification with a two-step hierarchical system which models the IPC hierarchy [5], Bouadjenek used query with a full patent description in conjunction with generic query reduction methods [6], Kim proposed the method to suggest diverse queries that can cover multiple aspects of the query (patent) [7], Ferraro's approach consist in segmenting the patent claim, using a rule-based approach, and a conditional random field is trained to segment the components into clauses [8], Andersson used the techniques by addressing three different relation extraction applications: acronym extraction, hyponymy extraction and factoid entity relation extraction [9].

In this paper, we propose a novel approach, in which we tried to combine both statistical and semantic features to increase the accuracy of the prior-art search.

## 2 Methods of Statistical and Semantic Patent Analysis

### 2.1 Statistical Analysis

The patent statistical analysis can be performed by various methods (methods of terms extraction, storing patent databases, patents comparison). It is necessary to evaluate the effectiveness of methods of patent database statistical analysis through implementation of the following stages:

- patent document tokenization by three different methods (tokenization with removing stop-words, tokenization with replacing synonyms and tokenization based on N-grams);
- building a term-document matrix;
- clustering based on Latent Dirichlet Allocation (LDA) [10] model and using the constructed model to obtain the distribution of vectors by the clusters (unnamed topics);
- storing the obtained vectors by two different methods (storage in distributed file system HDFS, storing in PostgreSQL database);
- the comparison of the obtained vectors by four different methods (based on the standard deviation of the vectors, element-by-element comparison of the vectors, Cosine similarity, and comparison of the vector's lengths).

The basic idea of LDA is that documents are represented as random mixtures of latent topics, where each topic is characterized by a distribution according to the words from documents array. Based on the LDA model can be used a patent database statistical analysis and distribution of patents by the unnamed topics.

The first stage of the statistical analysis is the tokenization of the patent text. In this case, the tokens will be individual words or N-grams from the patent text. After tokenization is necessary to make the lemmatization, i.e. converting the extracted words to their base form for the most accurate building of term-document matrix. Tokenization must implement three different techniques: tokenization with removing stop-words, tokenization with replacing synonyms, tokenization based on n-grams.

Tokenization with removing stop words (TokenRem) involves the removal stop-words from the text that do not carry meaning in this document: prepositions, interjections, etc. The text tokenization with removing stop words takes place via Apache Lucene library.

Tokenization with replacing synonyms (TokenSyn) also involves the removal of stop-words, but beyond that, all words-synonyms are present to one word (base alias). This approach allows us to build the most accurate term-document matrix and the LDA model but slow down compared to the usual tokenization.

Tokenization based on N-grams (TokenN) divides the text into phrases of any given length. This kind of tokenization allows you to build the LDA model based on keyphrases of a document that in theory should increase the effectiveness of the patent search. For the text tokenization based on N-grams, you need to clear the text from stop-words and extracted N-gram from the text by use of Apache Lucene library.

There are various methods of constructing a term-document matrix. To build the LDA model is sufficient to make a modified version of term-document matrix based on the TF [11]. TF (term frequency) indicates the importance of specific words within the document. First you need to get a dictionary of all the words a patent database $s = \{w_1, w_2, w_3, ..., w_n\}$, where $w_i$ is a unique word in the patent database. Then each $i$-th row of the term-document matrix represents the resulting dictionary, and each column - the number of occurrences of the word in $i$-th document.

The Big data processing framework Apache Spark and MLlib library for machine learning allow getting a dictionary of all words from the documents set and build a term-document matrix.

On the basis of the distribution vectors of patents by clusters on the base of LDA model is possible to make the selection of documents that match the query. The selection of relevant patents is made by comparing the distribution vectors. Since the comparison of multidimensional vectors can be produced based on different metrics of these vectors, and beforehand the most effective method are unknown, it is necessary to assess the effectiveness of the selected methods: element-by-element comparison (EE), calculate standard deviation (SD), compare vector size (VS), Cosine similarity (Cos).

After building the LDA model and obtain the vectors of topics per document distributions from the patent database these vectors must be stored in a data store for later retrieval and processing. In this case as a data repository for vectors can be either HDFS or PostgreSQL.

PostgreSQL allows you to store vectors of topics per document distributions in one table with two fields: ID and Vector (array of real values). Distributed file system

HDFS allows to effectively work with the patent database consisting of tens millions of documents. Framework Apache Spark allows you will store ID of the document and its vector in HDFS.

## 2.2 Machine Translation for Patent Examination

Patent examination on the base of prior-art retrieval from patent databases in other languages than application language needs pre-translation using machine translation tools. After analyzing was chosen statistical machine translation system Moses [12] (open-source project, licensed under the LGPL). Moses is a statistical machine translation system that allows you to automatically train translation models for any language pair. All you need is a collection of translated texts (parallel corpus). Once you have a trained model, an efficient search algorithm quickly finds the highest probability translation among the exponential number of choices. The Moses system can be integrated into any other system through XML-RPC protocol. For training Moses with GIZA ++ library used a set of several Russian-English corpora:

- training corpora based on patent families analysis (a set of patents registered in various countries (i.e. patent offices, including USPTO and Rospatent) to protect the same invention);
- fully parallel public-domain corpus consisting of 2100 United Nations General Assembly Resolutions with translations in the 6 official UN languages of the, including Russian and English;
- a collection of multilingual corpora (UMC) compiled at the Institute of Formal and Applied Linguistics (ÚFAL);
- fully parallel Russian and English corpuses from the website of Russian National Corpus;
- training corpora on the base of English and Russian version "The Art of Computer Programming" by Donald E. Knuth.

The total corpora volume was approximately 180,000 sentences (5,854,095 lexical units) in English and Russian.

The training process in Moses takes in the parallel data and uses occurrences of words and segments (known as phrases) to infer translation correspondences between the two languages of interest. Parallel Russian-English corpus was made on the basis of Patent families with using this algorithm:

- search an information about Patent Family in espacenet.com using the patent database in Russian;
- retrieval an English version of the patent in espacenet.com for this patent family;
- extract structured information such as claims from patents in English and Russian;
- proposals segmentation to obtain sentence-aligned data (parallel sentences) for the training process.

The data typically needs to be prepared before it is used in training, tokenizing the text and converting tokens to a standard case. Heuristics are used to remove sentence pairs which look to be misaligned, and long sentences are removed.

We used IRSTLM language model toolkit system to build a statistical language model. Experience with Moses revealed the difficulties with the usage of large paragraphs (more than 255 symbols) which must be segmented.

### 2.3 Semantic Analysis

On this step, we perform a semantic trees construction for application and patent claims. Building semantic trees and searching their intersections requires much more resources than statistical methods, which is a problem for the corpus of millions of patents. We build semantic trees using only patents claims because this part of the patent describes the invention and recites its limitations.

The text of patent claims has one feature that makes the effective use of existing solutions for building dependency trees is difficult. This feature is that the patent claims are written in one sentence, which sometimes includes hundreds of words. To solve this problem has been developed an algorithm of complex sentences segmentation [13]. Sentences are segmented on the base of transitional phrases of claims and special "marker" words such as "wherein", "such that", etc.

Then, we perform a morphological analysis of the patent text. For morphological analysis was chosen TreeTagger [14] for Russian and English language. The TreeTagger is a tool for annotating text with part-of-speech and lemma information. The TreeTagger has been successfully used to tag texts from basic world languages and is adaptable to other languages if a lexicon and a manually tagged training corpus are available.

After that, we used MaltParser [15] to perform semantic parsing (built dependency tree). Its main advantages are: it is open a source software, it allows to rich a moderate accuracy and it has pre-trained models for many languages, including Russian and English (Table 1).

Part-of-speech (POS) tags [16] are assigned to a single word according to its role in the sentence. Traditional grammar classifies words based on eight parts of speech: the verb (VB (base form), VBN (past participle), VBZ (3rd person singular present)), the noun (NN - single, NNS - plural), the Wh-determiner (WDT), the adjective (JJ), the preposition (IN), the determiner (DT), modal (MD), etc.

The Stanford typed dependencies [17] representation was designed to provide a simple description of the grammatical relationships in a sentence. It represents all sentence relationships uniformly as typed dependency relations. The current representation supports approximately 50 grammatical relations (in this example of patent dependence tree contains 11 grammatical relations: "amod" - adjectival modifier, "det" - determiner, "pobj" - object of a preposition, "nsubj" - nominal subject, "nsubjpass" - passive nominal subject, "aux" - auxiliary, "auxpass" - passive auxiliary, "nn" - noun compound modifier, "rcmod" - relative clause modifier, "prep" - prepositional modifier, "punct" - punctuation).

In the collapsed representation, dependencies involving prepositions, conjuncts, as well as information about the referent of relative clauses are collapsed to get direct dependencies between content words. We removed from dependence trees the grammatical relations such as "punct", "det", "prep", etc. The parent of removed node is transferred to the child node. So, we got dependency tree with Collapsed Stanford Dependencies (Table 2).

**Table 1.** Example of dependence tree in Conll'09

| Index word in a sentence | Word form itself | Word's lemma or stem | POS | Index of syntactic parent | Stanford typed dependencies |
|---|---|---|---|---|---|
| 1 | the | the | DT | 3 | det |
| 2 | sandwiched | sandwiched | VBN | 3 | amod |
| 3 | layers | layer | NNS | 6 | nsubjpass |
| 4 | can | can | MD | 6 | aux |
| 5 | be | be | VB | 6 | auxpass |
| 6 | rolled | roll | VBN | 0 | root |
| 7 | with | with | IN | 6 | prep |
| 8 | dielectric | dielectric | NN | 7 | pobj |
| 9 | into | into | IN | 6 | prep |
| 10 | a | a | DT | 12 | det |
| 11 | compact | compact | JJ | 12 | amod |
| 12 | form | form | NN | 9 | pobj |
| 13 | that | that | WDT | 14 | nsubj |
| 14 | looks | look | VBZ | 12 | rcmod |
| 15 | like | like | IN | 14 | det |
| 16 | rod | rod | NN | 15 | pobj |
| 17 | of | of | IN | 16 | prep |
| 18 | metal | metal | NN | 17 | nn |
| 19 | . | . | SENT | 6 | punct |

Then we use an approach based on the MTT (Meaning-Text Theory) [18]. Syntactic representations in MTT are implemented using dependency trees. According to MMT we merge collapsed Stanford Dependencies (SD) into the set of Deep syntactic relations. For sentence "The sandwiched layers can be rolled with dielectric into a compact form that looks like a rod of metal", the Stanford Dependencies, Collapsed Stanford Dependencies and Deep Syntactic Structure is representation in Table 2.

Actantial relations (I, II, III) are just numbered by increasing oblicity I for the most salient actant, II for the next, etc. We merge the following SD into actantial relation: "nsubj", "nsubjpass", "pobj", etc. The attributives relations (ATTR) cover all types of modifiers (circumstantials and attributes). We merge the following SD into relation ATTR: "amod", "rcmod", "nn", etc.

### 2.4 Semantic Similarity Calculation Between Application and Patents

On this step, we compare the semantic trees for application claims with trees from selected subset received on the step of the statistical analysis. We re-rank relevant patents from selected subset according to similarities between semantic trees.

In accordance with MTT trees at Deep syntactic representation level show dependency relations between terms (words) and look as networks with arrows running

**Table 2.** Transformation from dependence tree to semantic tree

| Stanford Dependencies | Collapsed Stanford Dependencies | Deep Syntactic Structure |
|---|---|---|
| det(layers-3, the-1) | amod(layers-3, sandwiched-2) | ATTR(layers-3, sandwiched-2 |
| amod(layers-3, sandwiched-2) | nsubjpass(rolled-6, layers-3) | I(rolled-6, layers-3) |
| nsubjpass(rolled-6, layers-3) | root(ROOT-0, rolled-6) | $OPER_2$(ROOT-0, rolled-6) |
| aux(rolled-6, can-4) | pobj(rolled-6, dielectric-8) | II(rolled-6, dielectric-8) |
| auxpass(rolled-6, be-5) | amod(form-12, compact-11) | ATTR(form-12, compact-11) |
| root(ROOT-0, rolled-6) | pobj(rolled-6, form-12) | III(rolled-6, form-12) |
| prep(rolled-6, with-7) | nsubj(looks-14, that-13) | I(looks-14, that-13) |
| pobj(with-7, dielectric-8) | rcmod(form-12, looks-14) | ATTR(form-12, looks-14) |
| prep(rolled-6, into-9) | pobj(looks-14, rod-16) | II(looks-14, rod-16) |
| det(form-12, a-10) | nn(rod-16, metal-18) | ATTR(rod-16, metal-18) |
| amod(form-12, compact-11) | | |
| pobj(into-9, form-12) | | |
| nsubj(looks-14, that-13) | | |
| rcmod(form-12, looks-14) | | |
| det(looks-14, like-15) | | |
| pobj(like-15, rod-16) | | |
| prep(rod-16, of-17) | | |
| nn(of-17, metal-18) | | |

from predicate nodes to argument nodes. The semantic tree is built on a base of sentence "The sandwiched layers can be rolled with insulating layer into a compact form that looks like parallel sheets of metal" has 4-level view presented in Fig. 1. At the null level of a semantic trees representation are the ROOTs, at the first level are the actant relations I, II, III, etc. according to the MTT, at the second level are the attributive relations, at the last fourth level are the expanded descriptions.

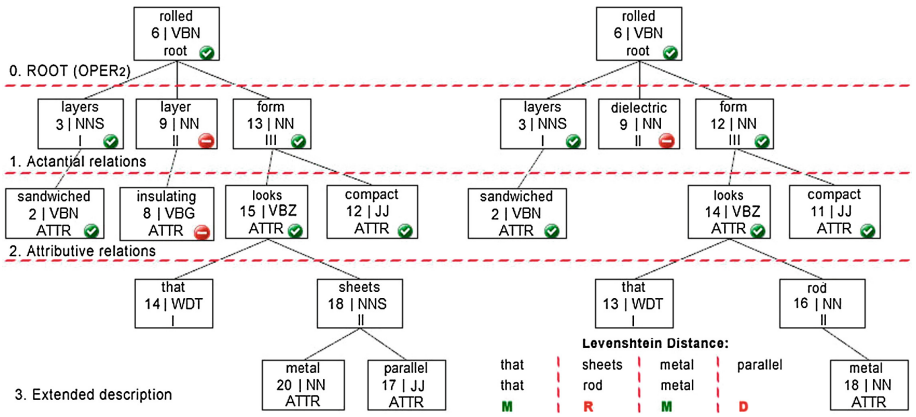


**Fig. 1.** Semantic trees of application (left view) and of the patent (right view).

After the stage of constructing semantic trees, the patent application is compared with each patent in the database. A comparison of the application with the i-th patent occurs by comparing each of the j-th semantic tree of the application with each k-th tree of the i-th patent.

**The First Stage of Semantic Trees Comparison**

According to the structure of the semantic tree, the tree's root (ROOT) is the verb (predicate). If the ROOTs of the application and the patent do not match further comparison of the trees is not performed and the comparison is made for the next semantic tree of the patent. We introduce the coefficient of two semantic trees similarity on the first 3 stages:

$$K^{j}_{\text{similarity}}(St_k, St_l) = \frac{\sum_{i=1}^{N_i} F_{\text{common}}(T_1, T_2)}{N_i}, \quad (1)$$

where $St_k$, $St_l$ are the semantic trees of $k$-sentence and $l$-sentence of an application claim and patent claim accordingly;

$j$ – number of semantic tree level;
$F_{common}(T_1, T_2)$ – MATCH function determines T1 и T2 terms of the compared semantic trees matches for the same parent terms;
$N_i$ – number of terms for semantic tree $St_k$ of application claim.

**The Second Stage of Semantic Trees Comparison**

If ROOTs (predicates) of trees are the same then the actant relations are compared. At this stage, each word (term) of the application is compared with each word of the corresponding patent. The tree similarity coefficient is calculated from the ratio of the number of matched words at the first level to the total number of words at the first level of the application. If the similarity coefficient (1) is less than 1/2 then a further comparison is not made.

If any term (word) is not matched on the first-third level, then the term is checked for significance. Testing the significance is based on a predetermined table that contains IDF [11] - inverse document frequency of terms in documents of patent databases. If the term's IDF is above a limit value then the term is not significant and is not taken into account of the similarity coefficient calculation.

In our example, the application semantic tree has three terms on the first level (Fig. 1). We produce a comparison: looking for matching terms with the identical actantial relations. The terms "layers" and "form" match in the application and the patent: +2/3. The term "layer" from the application did not match, check it for significance: the IDF coefficient is low than the limit value. So, the term "layer" must be taken into account in the calculation of the similarity coefficient that equal to 2/3 at the first level.

**The Third Stage of Semantic Trees Comparison**

At the third stage, semantic trees are compared at the 2nd level - the level of attributive relations. It is necessary that not only the application's term matches with the patent's term but also the parent terms are matched.

In the example (Fig. 1) the application has 4 attributes, at this level 3 application's terms match with the patent terms for identical parental terms. We check the non-matched term for significance, the IDF coefficient of the term "insulating" is less than the limit value, the word has a high significance, it must be taken into account when determining the similarity coefficient. Thus, the similarity coefficient for the 2nd level is 3/4.

**The Fourth Stage of Semantic Trees Comparison**

On the last stage compares the additional (extended) information of the third level of semantic trees.

The comparison is based on the Levenshtein distance [19]. This coefficient determines the minimum sequence of actions that need to be done to obtain another sequence of objects from the same. Actions are: insert (I), delete (D), replace (R), match (M). If the same terms from the level of attributive relations have child branches, these branches are decomposed into a linear sequence and the Levenshtein distance is determined (Fig. 1).

We introduce the coefficient of two semantic trees similarity on the 4 stage:

$$K^4_{\text{similarity}}(St_k, St_l) = \frac{LevLen}{N_i}, \tag{2}$$

where *LevLen* – the Levenshtein distance to compare trees.

At this level of the application's semantic tree, there are 3 terms, in the case of words similarity the coefficient increases by 1/4. In the case of any similarity coefficient at the 3rd level, the patent will participate in the ranking, the value of this coefficient affects only the position in the final relevant list. In the example the Levenshtein distance is 2 (MRMD), the similarity coefficient is 2/4.

The coefficient of similarity of semantic trees is summarized for each level and the total coefficient of the application and patent similarity is the sum of trees similarity coefficients.

The coefficient of application' and patent' semantic trees similarity:

$$K_{\text{similarity}} = \sum_{z=1}^{4} (K^z_{\text{similarity}} \times 10^{4-z}), \tag{3}$$

The coefficient of application and patent similarity:

$$K_S = \frac{\sum_{i=1}^{N_{st}} (\max_j (K_{\text{similarity}}(St_i, St_j)))}{\max_i (K_S)_i}, \tag{4}$$

where $St_i$, $St_j$ are the semantic trees of $i$-sentence and $k$-sentence of an application claim and patent claim accordingly;

$N_{st}$ – number of semantic trees of application claim.

## 3 Experiments and Results

The experiments are performed using a multiprocessor computer system with distributed memory (cluster) of the Volgograd State Technical University. The cluster uses the operating systems Linux Cent'OS 6.5, 6.7. The cluster entered the 22nd edition of the Top-50 rating of the Russian supercomputers (Table 3).

**Table 3.** Characteristics of cluster nodes

| Nodes | HDD | RAM | Cores |
|---|---|---|---|
| node21.cluster | 1 TiB/2.3 TiB | 7 GiB/15.6 GiB | 8 |
| node22.cluster | 439 GiB/2.3 TiB | 2 GiB/15.6 GiB | 8 |
| node47.cluster | 355.5 GiB/708.7 GiB | 2.4 GiB/62.8 GiB | 20 |
| node48.cluster | 355.5 GiB/708.7 GiB | 2.5 GiB/62.8 GiB | (40 with hyper-threading) |

The software was installed on the nodes of the cluster:

- Apache Spark - open-source cluster-computing framework, engine for large-scale data processing;
- Library for software implementation of the LDA method – Apache Spark MLlib;
- PostgreSQL.

Statistical and semantic portraits were formed for 990,000 Russian- and English-language patents and stored in the Document Storage on the basis of the HDFS file system.

The statistic analysis software [20] produces a patent search by different methods and evaluates the efficiency of patent search. The precision and recall basic measures used in evaluating search strategies. The recall is the ratio of the number of relevant patents retrieved to the total number of relevant patents in the database. Precision is the ratio of the number of relevant patents retrieved to the total number of irrelevant and relevant patents retrieved.

$$\text{Recall} = \frac{\text{a}}{\text{a}+\text{b}} \times 100\%, \tag{5}$$

$$\text{Precision} = \frac{\text{a}}{\text{a}+\text{c}} \times 100\%, \tag{6}$$

where a – number of relevant patents retrieved in the patent database; b – number of relevant patents not retrieved; a – number of irrelevant patents retrieved (Table 4).

**Table 4.** Statistic analysis software testing

| No | Tokenization | Vectors storage | Vectors comparison | Precision % | Recall % | Time search, c |
|---|---|---|---|---|---|---|
| 1 | TokenRem | HDFS | Cos | 89 | 65 | 1.4 |
| 2 | TokenRem | HDFS | VS | 87 | 63 | 1.5 |
| 3 | TokenRem | HDFS | SD | 85 | 61 | 1.3 |
| 4 | TokenRem | HDFS | EE | 86 | 63 | 1.3 |
| 5 | TokenRem | PostgreSQL | Cos | 89 | 65 | 2.8 |
| 6 | TokenRem | PostgreSQL | VS | 87 | 63 | 2.9 |
| 7 | TokenRem | PostgreSQL | SD | 85 | 61 | 3.0 |
| 8 | TokenRem | PostgreSQL | EE | 86 | 63 | 2.8 |
| 9 | TokenSyn | HDFS | Cos | 99 | 81 | 2.3 |
| 10 | TokenSyn | HDFS | VS | 92 | 78 | 2.4 |
| 11 | TokenSyn | HDFS | SD | 94 | 77 | 2.5 |
| 12 | TokenSyn | HDFS | EE | 95 | 74 | 2.5 |
| 13 | TokenSyn | PostgreSQL | Cos | 99 | 81 | 3.2 |
| 14 | TokenSyn | PostgreSQL | VS | 92 | 78 | 3.4 |
| 15 | TokenSyn | PostgreSQL | SD | 94 | 77 | 3.5 |
| 16 | TokenSyn | PostgreSQL | EE | 95 | 74 | 3.5 |
| 17 | TokenN | HDFS | Cos | 92 | 77 | 3.7 |
| 18 | TokenN | HDFS | VS | 88 | 75 | 3.8 |
| 19 | TokenN | HDFS | SD | 87 | 74 | 4.0 |
| 20 | TokenN | HDFS | EE | 87 | 73 | 3.9 |
| 21 | TokenN | PostgreSQL | Cos | 92 | 77 | 4.1 |
| 22 | TokenN | PostgreSQL | VS | 88 | 75 | 4.4 |
| 23 | TokenN | PostgreSQL | SD | 87 | 74 | 4.7 |
| 24 | TokenN | PostgreSQL | EE | 87 | 73 | 4.5 |

On the base of statistical analysis, testing results can be concluded: the most effective method for vectors storage is storing in HDFS, comparing vectors on base Cosine similarity, patent tokenization with replacing synonyms.

We chose the coefficient “Recall” for sets of the top 50, 100, 150, 200, 250, 500 most relevant patents retrieved as a criterion of the semantic analysis effectiveness. In the tests, the Recall will be 100% when the required patent is included in the set and 0% - not included in the set. The tables indicate the average Recall value for 20 tests.

To determine the maximum software effectiveness it is necessary to perform tests with various variations of semantic analysis methods (Table 5).

Verification the term significance increases the Recall. This is due to a more accurate ranking of the trees similarity, since insignificant, commonly used words do not affect the patents ranking.

To test multilingual versions was selected a method with verification of the term’s significance (Table 6).

**Table 5.** Semantic analysis w/ & w/o verification of term's significance

| Feature | Recall@500 | Recall@250 | Recall@200 | Recall@150 | Recall@100 | Recall@50 |
|---|---|---|---|---|---|---|
| With verification of term's significance | 96 | 95 | 93 | 92 | 88 | 81 |
| Without verification of term's significance | 89 | 85 | 84 | 80 | 76 | 72 |

**Table 6.** Semantic analysis for English and Russian patents

| Feature | Recall@500 | Recall@250 | Recall@200 | Recall@150 | Recall@100 | Recall@50 |
|---|---|---|---|---|---|---|
| Patents on English | 96 | 95 | 93 | 92 | 88 | 81 |
| Patents on Russian | 89 | 87 | 81 | 77 | 72 | 64 |

The results of checking the Russian- and English-language version of the semantic analyzer showed a lower Recall value for Russian patents - this is due to the complexity of Russian grammar.

## 4 Conclusion

For experiments with system prototype in the knowledge base were loaded the 990,000 patents from a different domain such as "Electricity", "Physics", "Mechanics" of the Russian Federation and United States patent databases.

On the first step, we compared the different variants of statistical analysis based on LDA method. Patent examination on the base of prior-art retrieval from patent databases in other languages than application language needs pre-translation using machine translation tools. On the step of the semantic analysis, we applied a new method for building a semantic network on the base of Stanford Dependencies and Meaning-Text Theory. On the step of semantic similarity calculation, we compare the semantic trees for application and patent claims.

Developed automated system prototype for the patent examination task, which is designed to help examiners in the examination process, significantly reduced search time and increased such criteria of search effectiveness as "Precision" and "Recall".

**Acknowledgement.** This research was partially supported by the Russian Foundation of Basic Research (grants No. 15-07-09142 A, No. 15-07-06254 A, No. 16-07-00534 A).

## References


1. Magdy, W., Jones, G.J.F.: Applying the KISS principle for the CLEF-IP 2010 prior art candidate patent search task. In: Workshop of the Cross-Language Evaluation Forum, LABs and Workshops, Notebook Papers (2010)
2. Verma, M., Varma, V.: Exploring keyphrase extraction and IPC classification vectors for prior art search. In: CLEF Notebook Papers/Labs/Workshop (2011)
3. Mahdabi, P., Crestani, F.: Query-driven mining of citation networks for patent citation retrieval and recommendation. In: ACM International Conference on Information and Knowledge Management (CIKM) (2014)
4. Xue, X., Croft, W.B.: Modeling reformulation using query distributions. J. ACM Trans. Inf. Syst. **31**(2) (2013). ACM, New York
5. D'hondt, E., Verberne, S., Oostdijk, N., Boves, L.: Patent classification on subgroup level using balanced winnow. In: Lupu, M., Mayer, K., Kando, N., Trippe, A. (eds.) Current Challenges in Patent Information Retrieval. TIRS, vol. 37, pp. 299–324. Springer, Heidelberg (2017). doi:10.1007/978-3-662-53817-3_11
6. Bouadjenek, M., Sanner, S., Ferraro, G.: A study of query reformulation of patent prior art search with partial patent applications. In: 15th International Conference on Artificial Intelligence and Law (ICAIL 2015), pp. 1–11. Association for Computing Machinery (ACM), USA (2015)
7. Kim, Y., Croft, W.B.: Diversifying query suggestions based on query documents. In: Proceedings of the SIGIR 2014 (2014)
8. Ferraro, G., Suominen, H., Nualart, J.: Segmentation of patent claims for improving their readability. In: 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR). Stroudsburg, PA 18360, USA, pp. 66–73 (2014)
9. Andersson, L., Hanbury, A., Rauber, A.: The portability of three types of text mining techniques into the patent text genre. In: Lupu, M., Mayer, K., Kando, N., Trippe, A. (eds.) Current Challenges in Patent Information Retrieval. TIRS, vol. 37, pp. 241–280. Springer, Heidelberg (2017). doi:10.1007/978-3-662-53817-3_9
10. Blei, D.M.: Latent Dirichlet allocation. J. Mach. Learn. Res. **3**(4–5), 993–1022 (2003)
11. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Inf. Process. Manag. **24**(5), 513–523 (1988)
12. Durrani, N., Sajjad, H., Hoang, H., Koehn, P.: Integrating an unsupervised transliteration model into statistical machine translation. In: Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL), Gothenburg, Sweden (2014)
13. Korobkin, D., Fomenkov, S., Kravets, A., Kolesnikov, S., Dykov, M.: Three-steps methodology for patents prior-art retrieval and structured physical knowledge extracting. In: Kravets, A., Shcherbakov, M., Kultsova, M., Shabalina, O. (eds.) Creativity in Intelligent Technologies and Data Science. CCIS, vol. 535, pp. 124–136. Springer, Cham (2015). doi:10.1007/978-3-319-23766-4_10
14. Toutanova, K., Manning, C.D.: Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In: Proceeding EMNLP 2000, vol. 13, Hong Kong, pp. 63–70 (2000)
15. Hall, J.: MaltParser – An Architecture for Inductive Labeled Dependency Parsing, p. 92. University of Colorado, Boulder (2006)
16. Haverinen, K., Viljanen, T., Laippala, V., Kohonen, S., Ginter, F., Salakoski, T.: Treebanking finnish. In: Proceedings of the Ninth International Workshop on Treebanks and Linguistic Theories (TLT) (2010)

17. de Marneffe, M.-C., Manning, C.D.: Stanford typed dependencies manual (2016)
18. Mel'čuk, I.A.: Dependency Syntax Theory and Practice. SUNY Publ, Albany (1988)
19. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Sov. Phys. Dokl. **10**(8), 707–710 (1966)
20. Korobkin, D.M., Fomenkov, S.A., Kravets, A.G., Golovanchikov, A.B.: Patent data analysis system for information extraction tasks. In: 13th International Conference on Applied Computing (AC) 2016, pp. 215–219 (2016)