Semantic patent analysis with Amazon Web Services

A Manukyan¹, D Korobkin¹, S Fomenkov¹ and S Kolesnikov¹
¹Volgograd State Technical University, Volgograd, Russia
E-mail: dkorobkin80@mail.ru

Abstract. Semantic analysis of the patent array allows us to solve several modern problems: (1) Clustering of the patent array. This information can be useful for identifying patent trends, key modern technologies, and predicting the demand for technologies in the future period; (2) Automation of the work of the patent office expert. Based on a full-text query (the text of a patent application), a search for analogous patents can be performed. This study describes a developed software that provides the possibility of clustering the patent array (topic modeling), identifying groups of related patents (not based on patent classification but on the basis of key terms/phrases extracted from the texts), and a search for patents using AWS technologies.

1. Introduction
One of the biggest problems of the twenty-first century related to patent law is the workload of the patent office. Only UPSTO (the US Patent and Trademark Office) reviews about 10 thousand patents a week. And each patent needs to be given special attention, to understand whether the patent rights of another patent are not affected, whether the patent is pseudoscientific, "obvious", and ultimately to make a complete classification of the patent with all the descriptions and references.

To solve the above, it is necessary to develop software for semantic analysis of the patent array. Semantic analysis of the patent array allows us to solve several modern problems:

1. Clustering of the patent array [1,2] (topic modeling) allows you to identify groups of related patents (not based on patent classification but on the basis of key terms/phrases extracted from the texts). This information can be useful for identifying patent trends, key modern technologies, and predicting the demand for technologies in the future period.

2. Automation of the work of the patent office expert [3]. On the basis of a full-text query (the text of a patent application), a search for analogous patents can be performed. In addition, the process of identifying key phrases in both the patent application text and the patent text can be automated.

In this regard, the aim of the work was to develop software for semantic analysis of the patent array for the tasks of patent search, clustering, and parsing.

2. Description of the process
The subject area of this article is closely related to the business processes taking place in the patent field. One of the problems of patent search is the hard human work of the patent office expert, or more precisely, the lack of automation of some parts of this work. For example, automating the keyword search process would save the initial stage of the patent office expert in identifying keywords, and clustering the patent array and identifying analog patents within a specific cluster would facilitate all other stages of patent analysis. If the developed software is used, the speed of the expert of the patent office will be increased by automating the process of identifying keywords in the patent application for further submission to the search engine, clustering the patent array-the search for analogous patents.
is narrowed by searching in a separate cluster. Currently, the patent office expert searches for keywords manually, and the search is made from extensive sources, such as the USPTO. In this regard, the task was set to create software to solve this problem.

The expert examines the application and uses software to perform a patent search. To conduct it, the expert must select from the patent application the keywords that, in his opinion, most accurately characterize the patent. Next, a request is formed, according to which software searches for analog patents from the available patent database. The expert should analyze all the issued patents—analogs and make a conclusion about the patentability of the invention. Each patent office examiner must go through these steps for each patent Fig. 1.

Figure 1. Methodology of the expert's work

As you can see in the presented algorithm, the patent office expert must manually search for key phrases in the patent, as well as make a selection from the entire patent database.

As described earlier, one of the problems of patent search is the hard human work of the patent office expert, or more precisely, the lack of automation of some parts of this work. For example, automating the keyword search process would save the initial stage of the patent office expert in identifying keywords, and clustering the patent array and identifying analog patents within a specific cluster would facilitate all other stages of patent analysis.

Let's look at how the expert Advisor will work with the implementation of the software in Fig. 2.

As you can see, since the software is used, the speed of the expert of the patent office will be increased by automating the process of identifying keywords in the patent application for further submission to the search engine, clustering the patent array—the search for analogous patents is narrowed by searching in a separate cluster.

3. Technical solutions

During the creation of the required program, you will need to parse patent documents and texts. A parser (parser) is a part of a program that converts input data (usually text) into a structured format that is necessary for the tasks of their subsequent analysis and use. Technically, the parser parses data (for example, text).

The main advantages of parsing are:

- data collection is faster and in any mode, even around the clock;
- follow all the set parameters, even very subtle ones;
- avoiding mistakes from inattention or fatigue;
the ability to perform a regular check at a specified interval

presentation of the collected data in any required format without unnecessary effort;

This paper uses the XML parser format. XML parser is a program that extracts data from an XML source file from the format and saves or uses it for subsequent actions.

To solve the problems of clustering the patent array, identifying key phrases, and searching for similar patents, Amazon technologies [4,5] were used.

3.1 Amazon Comprehend Detects Key Phrases (ACDKP)
A key phrase is a string containing a noun phrase that describes a particular thing. It generally consists of a noun and the modifiers that distinguish it. For example, "day" is a noun; "a beautiful day" is a noun phrase that includes an article ("a") and an adjective ("beautiful"). Each key phrase includes a score that indicates the level of confidence that Amazon Comprehend has that the string is a noun phrase. You can use the score to determine if the detection has high enough confidence for your application. Detect key phrase operations can be performed using any of the primary languages supported by Amazon Comprehend. All documents must be in the same language.

3.2 Amazon Comprehend Topic Modeling (ACTM)
Amazon Comprehend uses a Latent Dirichlet Allocation-based learning model to determine the topics in a set of documents. It examines each document to determine the context and meaning of a word. The set of words that frequently belong to the same context across the entire document set make up a topic. A word is associated with a topic in a document based on how prevalent that topic is in a document and how much affinity the topic has to the word. The same word can be associated with different topics in different documents based on the topic distribution in a particular document. For example, the word "glucose" in an article that talks predominantly about sports can be assigned to the topic "sports," while the same word in an article about "medicine" will be assigned to the topic "medicine." Each word associated with a topic is given a weight that indicates how much the word helps define the topic. The weight is an indication of how many times the word occurs in the topic compared to other words in the topic, across the entire document set.

After Amazon Comprehend processes your document collection, it returns a compressed archive containing two files, topic-terms.csv, and doc-topics.csv. For more information about the output file, see OutputDataConfig. The first output file, topic-terms.csv, is a list of topics in the collection. For each topic, the list includes, by default, the top terms by topic according to their weight.

For example, if you give Amazon Comprehend a collection of newspaper articles, it might return the following to describe the first two topics in the collection: The weights represent a probability distribution over the words in a given topic. Since Amazon Comprehend returns only the top 10 words for each topic the weights won't sum to 1.0. In the rare cases where there are less than 10 words in a topic, the weights will sum to 1.0. The words are sorted by their discriminative power by looking at their occurrence across all topics. Typically this is the same as their weight, but in some cases, such as the words "play" and "yard" in the table, this results in an order that is not the same as the weight. You can specify the number of topics to return. For example, if you ask Amazon Comprehend to return 25 topics, it returns the 25 most prominent topics in the collection. Amazon Comprehend can detect up to 100 topics in a collection. Choose the number of topics based on your knowledge of the domain. It may take some experimentation to arrive at the correct number. The second file, doc-topics.csv, lists the documents associated with a topic and the proportion of the document that is concerned with the topic. If you specified ONE_DOC_PER_FILE the document is identified by the file name. If you specified ONE_DOC_PER_LINE the document is identified by the file name and the 0-indexed line number within the file.

3.3 Amazon Twinword Text Similarity (ATTS)
Evaluate the similarity of two words, sentences, paragraphs, or documents. Get a score of how similar or different two texts are. For an example of a real use case, this API was used in creating the first semantic keyword research tool that can sort by relevance. Keyword research involves skimming through long lists of keywords to find the most relevant ones. This API makes keyword research
quicker by auto sorting each keyword in the list by its similarity to a user-specified topic. This is one of the many places where sorting a long list semantically helps.

4. **Architecture of software**

The software was implemented in the Python programming language. The architecture of the developed software is shown in Fig. 3.

Numbers indicate data streams: 1 – Provision of patent texts from patent documents + parsing of parsed patents in the directory; 2 – Obtained from parsing patent documents - patent texts; 3 – Parsing of parsed patents in DynamoDB; 4 – Various connections and checks to DynamoDB; 5 – Filling the Amazon S3 storage with parsed patents from root directories; 6 – Connecting to Amazon technologies; 7 – Connecting to Amazon Comprehend technology; 8 – Connecting to Amazon Twinword technology; 9 – Clustering the patent array using Topic technology Modeling; 10-Identification of key phrases using DetectKeyPhrases technology; 11 - The result of Topic Modeling is placed in Amazon S3; 12 – The result of DetectKeyPhrases is placed in Amazon S3; 13 - Search for analog patents using Text Similarity technology; 14 – The result of Text Similarity is placed in the root directories; 0-The user has access to ALL threads to some extent.

![Figure 3. Software architecture](image)

During the implementation, the following functions were developed [6, 7]:

1. **Parsing of patent documents**

Description: the function is designed to divide one complete patent document into many separate patents, with which further transformations will be performed, and creates them in the selected root directory. Input data: the function input is provided with a patent document in the form of an XML file that stores patents. Output: The function outputs the number of patents to be processed in the future.

2. **Parsing patents to DynamoDB and the root directory**

Description: The patent parameters are parsed: publication date, title, classification, authors’ names, code, link to the patent; for further filling in DynamoDB. The patent descriptions are parsed, which will be processed in the future: the fields of abstractions, descriptions, and claims; they are placed in the form of txt files in the root directory for direct storage on the media, as well as for further filling the Amazon S3 storage. Input data: the function input is supplied with a patent in the form of an XML file, as well as parameters for connecting to DynamoDB. Output data: since the main role of the function is parsing to the database and root directories, there is no output data.

3. **Parsing of patents in Amazon S3**

Description: The Amazon S3 storage is being filled, for further transformations with Amazon technologies. Input data: the function input is provided with text files from the root directories, as well
as parameters for connecting to Amazon S3. Output data: since the main role of the function is to parse in the storage, there is no output.

4. Identification of key phrases in the text of a patent application, patent using Amazon Comprehend technology

Description: Using the Amazon Comprehend-Detect Key Phrases technology, the selected patent (or patents) contains keywords with their significance. Even at this stage, this makes the work of the patent office expert much easier, since the expert usually searches for such keywords for further search in search engines (for example, Google patents) manually. Input data: Text file from Amazon S3 storage, Amazon S3, and Amazon Comprehend connection parameters. Output: a file without a type (easily converted to text) with a complete list of keywords and their values

5. Patent array clustering (topic modeling)

Description: The entire database of more than 50 thousand patents is implicitly clustered into 10 topics (clusters) using the Amazon Comprehend – Topic Modeling technology. Each patent presented in the form of a processed txt file is related to the topics selected during the simulation in a certain proportion. Input data: the entire patent database stored in Amazon S3, connection parameters to Amazon S3 and Amazon Comprehend. Output: A bucket in Amazon S3 that contains two csv files. The first is a set of terms in topics, for understanding the logic of implicit clustering. The second is a patent array that has undergone clustering, in more than 190 thousand lines, including the name of the patent, its topic and the degree of belonging to this topic.

6. Search for analog patents based on a full-text query using Amazon Twinword technology

Description: A patent array that has undergone clustering is processed using Amazon Twinword technology to search for analog patents. This procedure follows different principles. For example, you can select a group of patents that belong to the topic - "001" and have more than 0.8 belonging to this topic. Next, one patent is selected from this group and checked with all the others to check the “strong" similarity of the texts. A reverse check can also be made, for example, patents belonging to different topics and having more than 0.8 belonging to these topics can be selected to check for “weak" similarity of texts. Input data: patent array that has undergone clustering, connection parameters to Amazon Twinword. Output: a json file with the significance of the patents being compared.

For structuring and storing data, it was decided to use the DynamoDB NoSQL database. The database stores objects of patent information (Patents in Fig. 4) extracted by the parser.

![Figure 4. DynamoDB NoSQL database structure](image)

A key phrase is a string containing a noun phrase that describes a particular thing. It generally consists of a noun and the modifiers that distinguish it. For example, "day" is a noun; "a beautiful day" is a noun phrase that includes an article ("a") and an adjective ("beautiful"). Each key phrase includes a score that indicates the level of confidence that Amazon Comprehend has that the string is a noun phrase. Amazon Comprehend approves that you can use the score to determine if the detection has high enough confidence for your application. Detect key phrases operations can be performed using any of the primary languages supported by Amazon Comprehend. All documents must be in the same language.
On Fig. 5 you can see text input and on Fig. 6 you can see Amazon Comprehend Key Phrases result.

Figure 5. Amazon Comprehend Text Analysis

Figure 6. Amazon Comprehend Key Phrases result

Amazon Comprehend uses a Latent Dirichlet Allocation-based learning model to determine the topics in a set of documents. It examines each document to determine the context and meaning of a word. The set of words that frequently belong to the same context across the entire document set make up a topic.

A word is associated to a topic in a document based on how prevalent that topic is in a document and how much affinity the topic has to the word. The same word can be associated with different topics in different documents based on the topic distribution in a particular document. For example, the word "glucose" in an article that talks predominantly about sports can be assigned to the topic "sports," while the same word in an article about "medicine" will be assigned to the topic "medicine." Each word associated with a topic is given a weight that indicates how much the word helps define the topic. The weight is an indication of how many times the word occurs in the topic compared to other words in the topic, across the entire document set.

Amazon Comprehend Topic Modeling Result on Fig. 7,8. Evaluate the similarity of two words, sentences, paragraphs, or documents. Get a score of how similar or different two texts are. For example Amazon says about a real use case, this API was used in creating the first semantic keyword research tool that can sort by relevance. Keyword research involves skimming through long lists of keywords to find the most relevant ones. Amazon Twinword Text Similarity result on Fig. 9.
5. Conclusion

As a result of this work, a software was developed that provides the possibility of clustering the patent array (topic modeling) and allows identifying groups of related patents (not based on patent classification, but on the basis of key terms/phrases extracted from the texts) using ACTM technologies. Also, on the basis of a full-text query (the text of the patent application), a search for analog patents was carried out using ATTS technologies. The process of identifying key phrases in both the patent application text and the patent text was automated using ACDKP technologies.

In this regard, the purpose of the work, which was to develop software to cluster the patent array and speed up the work of the patent office expert by searching for key phrases and analogous patents, I believe is fully fulfilled.

The main direction of improvement and further development of the developed software is to increase the speed of the program, improve the algorithms and logic of parsing.

The output of the program should be a report on the persons found. It must contain the following data: the name of the person, the time interval where the software detected the person. The output data is written to the application database for later use.

6. Acknowledgments

The reported study was funded by RFBR and the Administration of the Volgograd region according to the research projects 19-47-340007, 19-41-340016.

References

[1] Korobkin D M, Fomenkov S A and Golovanchikov A B 2018 Method of identification of patent trends based on descriptions of technical functions Journal of Physics: Conference Series 1015 032065

[2] Kochura O A, Korobkin D M, Fomenkov S A and Kolesnikov S G 2021 Development of the patent array analysis module based on the "Problem-Solution" model Journal of Physics: Conference Series 1801 012014

[3] Kozina S A., Korobkin D M, Fomenkov S A and Vasiliev S S 2021 The formation of a unified
database on physical subjects *Journal of Physics: Conference Series* **1801** 012015

[4] Joe Baron, Hisham Baz, Tim Bixler, Biff Gaut, Kevin E. Kelly, Sean Senior, John Stamper 2016, *AWS Certified Solutions Architect Official Study Guide: Associate Exam*. [Online]. – Available; https://www.pdfdrive.com/aws-certified-solutions-architect-official-study-guide-associate-exam-e38558089.html

[5] Kees Varekamp 2018, *Are Classes Clusters?*. [Online]. – Available; https://arxiv.org/abs/2104.07840

[6] Larry Jeffus 2016, *Welding: Principles and Applications*. [Online]. – Available; https://www.pdfdrive.com/welding-principles-and-applications-e187528412.html

[7] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, Ting Wang 2018, *TextBugger: Generating Adversarial Text Against Real-world Applications*. [Online]. – Available; https://arxiv.org/abs/1812.05271