Development of the patent array analysis module based on the “Problem-Solution” model

O A Kochura¹, D M Korobkin¹, S A Fomenkov¹ and S G Kolesnikov¹

¹ Volgograd State Technical University, 28, Lenin ave., Volgograd, 400005, Russia

E-mail: olegff134@gmail.com

Abstract. In the modern world, there is a description of many technical solutions in different types. A patent is one of these types. Analysis of the patent array based on the “Problem-Solution” model allows not only to analyze existing technical solutions, but also to improve the characteristics of individual elements of a technical object and, accordingly, the technical object (TO) itself. To do this, you need to detail the structural-functional structure of the TO, i.e. define the elements (components) of TO and the functions of the TO elements. To extract descriptions of functions of a technical object, we use the semantic construction "Subject-Action-Object" (SAO). If "Action" is an attributive verb, such as "reduce", "improve", "increase", etc., then "Subject" can affect "Object" as a problem or need. The SAO structure can be organized in the "problem-solution" format, where the "Object-action" (AO) forms the problem or need, and the "Subject" (S) sets the solution. Structured data will be extracted from natural language text using the Yandex Tomita-parser technology. Then, after all the procedures, it is advisable to fill the database with the information obtained for convenient analysis and search.

1. Introduction

In the modern world, there is a description of many technical solutions in different types. A patent is one of these types. Analysis of the patent array based on the “Problem-Solution” model allows not only to analyze existing technical solutions, but also to improve the characteristics of individual elements of a technical object and, accordingly, the technical object (TO) itself. To do this, you need to detail the structural-functional structure of the TO [1,2], i.e. define the elements (components) of TO and the functions of the TO elements.

Since patents are stored locally, a directory containing patent data in XML format is selected. After that, the program recursively starts checking each document for compliance with the correct structure. If the requirements are met, then the process of parsing from XML format to JSON takes place, preserving the entire structure and data that it originally had.

Since in our case the JSON file is a set of key-value data, you should list the keys whose values need to be extracted [3]:

- "B190" - code or other means of identifying the Agency or organization that made the publication;
- "B110" - patent number;
- "B130" - document type code;
- "B540" - the title of the document in Russian and English languages;
- "B510": classification index;
- "B700" - authors in both languages;
- "B100" - date of publication of the patent;
- "B560" - the list of citing patents.

After receiving the data, they are entered into the MongoDB database, so that subsequent blocks write the extracted AOC structures to them.

2. Materials and methods
2.1. Search in the patent text for problems solved by the device

Next, a search is carried out for information that reflects the purpose for which the invention was created, what problem it solves. This data is contained in the “description” key and stored in paragraphs. The difficulty in extracting is that the key-value contains a lot of unnecessary information at this stage, which can confuse the user. The dictionary of keyword constructions (“technical result”, “invention problem”, etc.) helps to avoid this problem, with the help of which the block filters out “unnecessary” paragraphs. The resulting paragraphs are saved to an intermediate text file. The procedure is repeated recursively until the files run out.

The saved files of the previous block will be needed for semantic analysis, and more specifically, to extract the very structures "Problem - solution". The extraction will take place using a third-party software tool - Yandex Tomita-parser technology. The parser receives at the input a text document that contains sentences in natural language, at the output - a solution, an example is shown in Figure 1. The output format of this debugging information is intended for the convenience of viewing and controlling the process [4].

![Figure 1](image1.png)

*Figure 1. Sample output file in .txt format.*

To work with the Tomita parser, you need to write a context-sensitive grammar that will allow you to extract the AOC structures necessary for the task. The grammar is shown in Figure 2.

```
#GRAMMAR_ROOT Directive points to the terminal that is the root of this grammar. The #encoding Directive specifies the encoding of the current grammar file.

Terminals used in grammar:
- Word - any word consisting of letters of the Russian or Latin alphabet;
- SimConjAnd – separate the word "and" from the part of speech;
- Noun – noun;
- Adj – adjective, ordinal numeral, participle, or pronominal adjective;
```

![Figure 2](image2.png)

*Figure 2. Triplet AOC extraction Grammar.*
Verb – verb;
Prep – preposition.
That is, to limit the terminal or non-terminal, litters are used, which are formatted as Word<litter>.

Marks used in grammar with their definitions [5]:
- kwtype – character restriction by the specified dictionary, in our case, the improve_verbs dictionary;
- rt – indicates that this nonterminal is a vertex of the syntax group;
- gram – checks the values of grammatical characteristics separately for each homonym;
- gn-agr – checks the match in gender and number;
- GU – same features as gram, plus the ability to check grammatical values for all homonyms at the same time.

The following grammemes are also used in grammar:
- ins – instrumental case;
- abl – ablative, original case.

Rule chains are connected by unary operators:
- operator | used to abbreviate rules with the same left side;
- operator * after a terminal or nonterminal means that the character is repeated zero or more times;
- operator + after terminal or nonterminal means that the character is repeated one or more times.

Figure 3. Example of added AOC structures.

The words interp are used to output data to output files. For interpretation, the word interp is written, then the name of the fact and the name of the field inside this fact are indicated in brackets [5].

The function of the last block is to record the extracted structures obtained using the previous block in the database to the corresponding patents. The initial action does not differ from the previous blocks – the file obtained at the stage of extracting facts is taken from the local directory. XML produces AOC structures that are written to the MongoDB database. An example of a saved file is shown in Figure 3.

3. Results and discussion

To implement the program, it was decided to develop the following algorithms:
- an algorithm for extracting information for identifying a patent from patent data and converting them into JSON format (Figure 4);
- an algorithm for extracting constructions by the Tomita-parser (Figure 5);
an algorithm for augmenting MongoDB with AOC constructs (Figure 6).

To obtain patent documents of semantic structures “Problem-solution” necessary to increase the efficiency of synthesis of new technical solutions using morphological search, and filling the database, a software was developed, shown in Figure 8.

**Figure 5.** Algorithm for extracting information for identifying a patent.

**Figure 6.** Algorithm for extracting constructions by Tomita-parser.
Algorithm for augmenting MongoDB with AOC constructs.

The software consists of the following program modules:
- parsing input files;
- information extraction for patent identification;
- fact extracting;
- filling the database.

4. Conclusion
In the course of this work, the following tasks were performed:
- the block extracting patent description elements from a local folder was implemented;
- an algorithm for extracting information for patent identification from patent data in JSON format has been developed;
- an algorithm for extracting facts from a text file using a Tomita parser has been developed;
- an algorithm has been developed for extracting AOC structures from intermediate XML files and saving them to a database;
- implemented a block for parsing patent data in JSON format;
- implemented block extracting semantic structures "Problem-solution" with the help of the Tomita-parser;
- implemented a block for filling the database with elements of the patent description;
- the module is implemented for Linux systems.

5. Acknowledgments
The reported study was funded by RFBR (research projects 18-07-01086), RFBR and Administration of the Volgograd region (projects 19-47-340007, 19-41-340016).
References
[1] Vasiliev S S, Korobkin D M, Kravets A G, Fomenkov S A and Kolesnikov S G 2020 *Extraction of Cyber-Physical Systems Inventions’ Structural Elements of Russian-Language Patents* Studies in Systems, Decision and Control 259 55-68
[2] Kharitonov A, Korobkin D M, Kravets A G, Fomenkov S A and Kolesnikov S G 2020 *Extraction of Morphological Features of Technical Systems from Russian Patent* CEUR Workshop Proceedings 2475 205-213
[3] Architectonics of classification indices Retrieved from: https://www.fips.ru/elektronnye-servisy/klassifikatory/mezdunarodnaya-patentnaya-klassifikatsiya/ii-arkhitektonika-klassifikatsionnykh-indekov.php
[4] Developer’s Guide (Yandex). Retrieved from: https://yandex.ru/dev/tomita/doc/dg/concept/run-parser-docpage.
[5] Litter-restrictions (Yandex) Retrieved from: https://github.com/yandex/tomita-parser/blob/master/docs/ru/dg/labels-limits.md.