The Software for Formation of Technical Function Assessments Based on the Patent Analysis

Dmitriy Korobkin, Sergey Fomenkov, Alexander Zlobin, Dmitriy Shabanov, and Alexander Golovanchikov

Abstract Based on the developed method of construction of a database of technical functions carried out by physical effects, realized the software with following functions: (a) extraction of technical functions; (b) search of the physical effect description; (c) construction of term-document matrices "Patent - Technical Function" and "Patent - Physical Effect"; (d) construction based on the reduced term-document matrices the matrix "Physical Effect - Technical Function". The system consists of two main and independent parts: the patents repository and the semantic core. The repository implements a standard CRUD interface for creating, reading, updating, and deleting documents. The semantic core is a library that implements all text processing functionality. The correctness of algorithms work was estimated on a test sample, the method of extraction of technical functions showed the following criteria: precision—0.87, recall—0.77, F—0.82, the method of search of physical effect description showed precision—0.92.

Keywords Technical functions · Physical effects · Patents

1 Introduction

The chapter shows the developed method for the automated construction of a database of technical functions performed by physical effects. For the synthesis of the physical principle of operation of new technical systems [1] in several scientific approaches [2, 3], physical effects (PE) are used. PE implements technical functions, which in turn constitute the constructive functional structure of the technical system [4]. The authors have developed a method for extracting descriptions of physical effects and technical functions from US patent documents (USPTO) and Rospatent. The method of automatic creation of a table of technical functions performed by physical effects is based on identifying latent dependencies in term-document matrices «Physical Effects-Patents» and «Technical Functions-Patents» [5].

D. Korobkin (✉) · S. Fomenkov · A. Zlobin · D. Shabanov · A. Golovanchikov
Volgograd State Technical University, 28 Lenin Av, Volgograd 400005, Russia

A. G. Kravets et al. (eds.), *Cyber-Physical Systems*, Studies in Systems, Decision and Control 350, https://doi.org/10.1007/978-3-030-67892-0_12

The purpose of this work is to programmatically implement a formation system of the matrix of technical functions performed by physical effects based on patent array analysis.

2 The Methodology

The automated system (AS) should provide the following functions:

- Extraction of technical functions in the «Subject - Action - Object» (SAO) [6–8] format from the texts of patent documents;
- Search for a description of the physical effect in the text of patent documents;
- Construction of the term-documentary matrix «Patent - Technical function», the elements of which are the values of the frequency response TF-IDF [9] for the corresponding technical function and patent document;
- Construction of the term-documentary matrix «Patent - Physical effect», the elements of which are the values of the frequency response TF-IDF for the corresponding physical effect and the patent document;
- Reduction of the space of technical functions for the term-document matrix "Patent - Technical function» and the space of physical and technical effects for the term-document matrix «Patent - Physical effect»;
- Construction from the reduced term-document matrices «Patent - Physical effect» and «Patent - Technical function» of the matrix «Physical effect - Technical function» based on the method of cosines as a characteristic of the representations of the physical effect and technical function in the space of patent.

AS is implemented in python version 3.7.2. For morphological tagging the program TreeTagger [10, 11] is used, for syntactic parsing—UDPipe [12]. The information structure of the input XML documents must conform to the document type declaration (DTD) «us-patent-application», the version of the patent document must conform to «v4.4 2014-04-03», which meets the general description of the «ST.36» standard.

The AS consists of two main and independent parts (Fig. 1): the patent array storage and the semantic core.

The patent array storage implements the functionality of storing and issuing to the user the texts of patent documents and their meta-data, such as number, class according to IPC classification [13], and so on. The storage implements the standard interface for CRUD operations for creating, reading (receiving), updating, and deleting documents, as well as the functionality of writing queries to retrieve data, which allows you to filter documents by the value of certain fields. Such an implementation allows, if necessary, to replace this module with any NoSQL storage in the future. The document storage implementation stores the patent data and all the necessary metadata locally in a specified directory. If cluster computing is required, the data folder can be placed on any network file system that supports FUSE technology. From the program interface side, this storage is an interface that provides

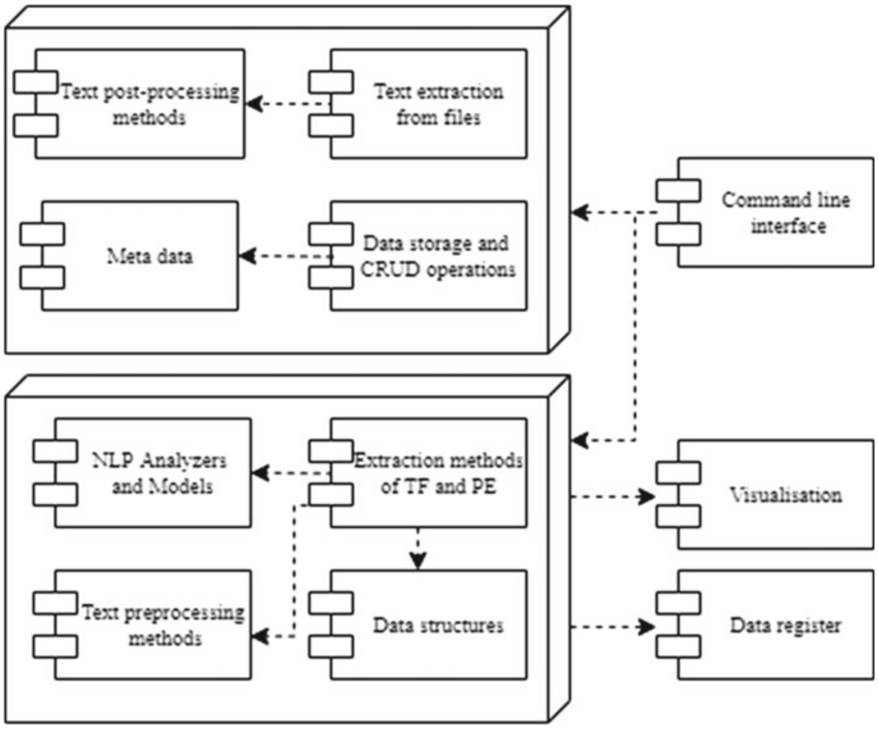

Text post-processing methods
Text extraction from files
Command line interface
Meta data
Data storage and CRUD operations
NLP Analyzers and Models
Extraction methods of TF and PE
Visualisation
Text preprocessing methods
Data structures
Data register

Fig. 1 Component diagram

a collection of documents in the form of a generator, which allows not to store the entire requested sample in memory, but to retrieve data only when accessing them.

All work on extracting data from files of patent documents is delegated to a separate library that implements the functionality of reading and extracting data for specific formats of patent documents. All communication between the store and the data extraction library is done through a set of data classes. The architecture of the block for working with the data warehouse is presented in the form of class diagrams (Fig. 2) and object diagrams (Fig. 3).

The functionality of the CRUD operation to the patent array is implemented in the «Bulk» class that works with objects that implement the «Document» interface—in this case, «Patent». The «DataFetcher» is responsible for extracting data and creating data document objects. For its operation, two objects of classes are needed that implement the «DocumentReader» and «DocumentParser» interfaces. «DocumentReader» extracts the text of patents from patent document files in their original formats (files from the USPTO database are contained in yourself several patents, archives, and other data at once). «DocumentParser» is responsible for the creation of document objects (class «Patent»). Since there are several dozen formats of input files and formats of the documents themselves, for simplicity, most of them are omitted in the diagram and replaced by the "…" sign, which symbolizes many

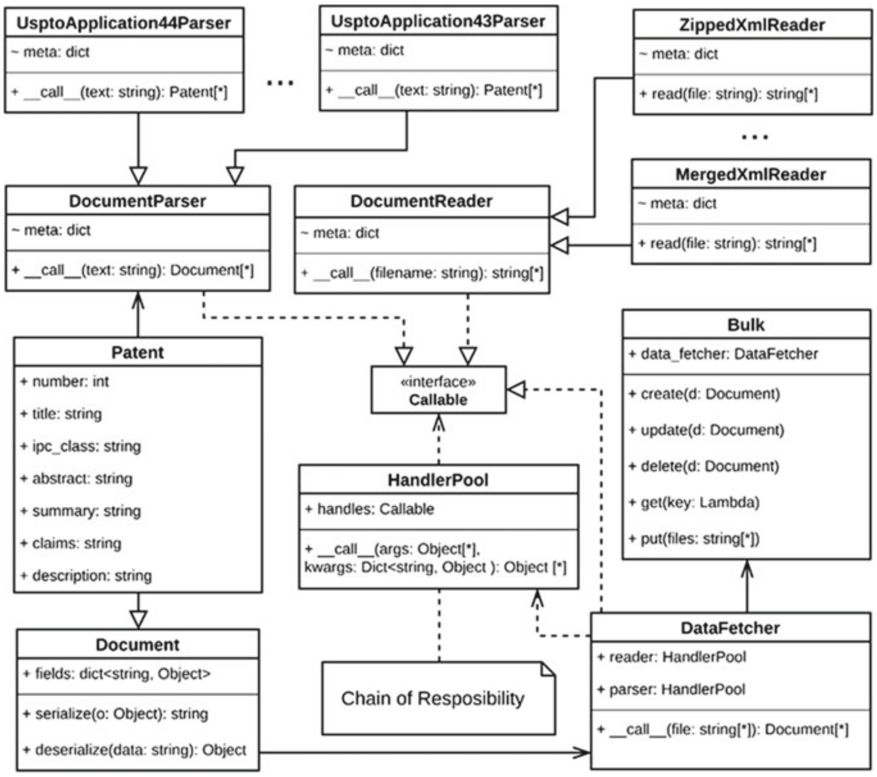


Fig. 2 Document storage class diagram

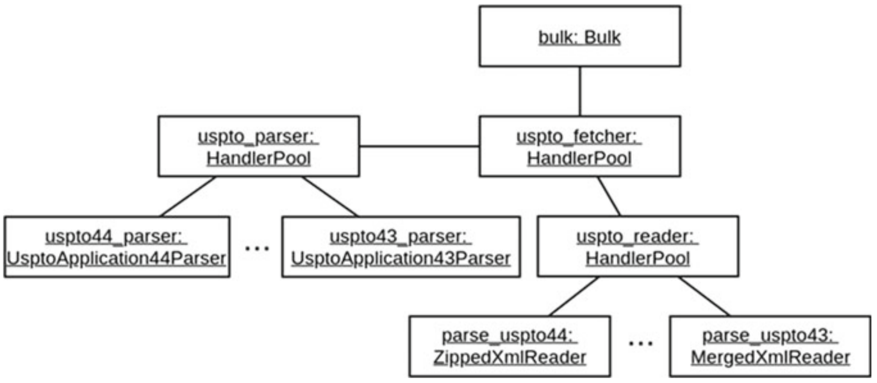

bulk: Bulk
uspto_parser: HandlerPool
uspto_fetcher: HandlerPool
uspto44_parser: UsptoApplication44Parser
uspto43_parser: UsptoApplication43Parser
uspto_reader: HandlerPool
parse_uspto44: ZippedXmlReader
parse_uspto43: MergedXmlReader

Fig. 3 Patent storage object diagram

different implementations for each format. For the same reasons, the «HandlerPool» class has been introduced, which is the standard «Chain-of-responsibility» design pattern. «HandlerPool» stores a list of all registered handlers and, when a request for processing arrives, delegates it to one of them, which makes it easy to add functionality for processing new input data formats with small code changes. The class object relationships are shown in Fig. 3.

The second part (semantic core) is a library that implements all the functionality for text processing:

- graphemic and lexical analysis, implemented in the form of text processors, as well as adapters to third-party word processing libraries such as NLTK [14];
- morphological and syntactic analysis, implemented in the form of adapters to third-party libraries TreeTagger, MaltParser [15, 16], UdPipe, and others;
- syntactic and semantic analysis—packages for working with PE and TF that programmatically implement the methods and data structures described in this work.

The architecture of the semantic core block is presented in the form of class diagrams (Figs. 4, 5) and object diagrams (Fig. 6).

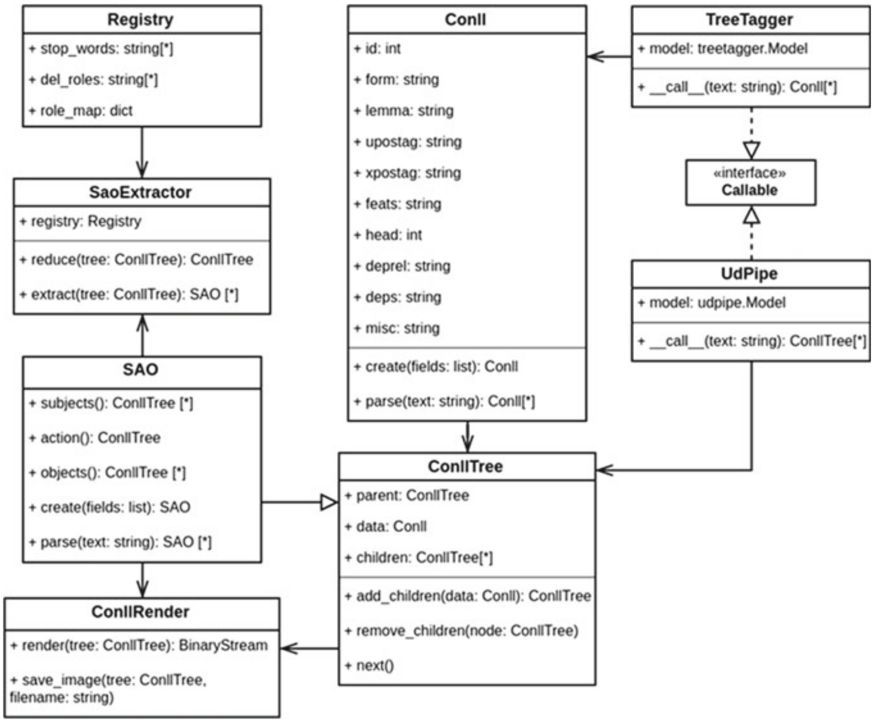

Registry
+ stop_words: string[*]
+ del_roles: string[*]
+ role_map: dict
SaoExtractor
+ registry: Registry
+ reduce(tree: ConllTree): ConllTree
+ extract(tree: ConllTree): SAO [*]
SAO
+ subjects(): ConllTree [*]
+ action(): ConllTree
+ objects(): ConllTree [*]
+ create(fields: list): SAO
+ parse(text: string): SAO [*]
ConllRender
+ render(tree: ConllTree): BinaryStream
+ save_image(tree: ConllTree, filename: string)
Conll
+ id: int
+ form: string
+ lemma: string
+ upostag: string
+ xpostag: string
+ feats: string
+ head: int
+ deprel: string
+ deps: string
+ misc: string
+ create(fields: list): Conll
+ parse(text: string): Conll[*]
ConllTree
+ parent: ConllTree
+ data: Conll
+ children: ConllTree[*]
+ add_children(data: Conll): ConllTree
+ remove_children(node: ConllTree)
+ next()
TreeTagger
+ model: treetagger.Model
+ __call__(text: string): Conll[*]
«interface»
Callable
UdPipe
+ model: udpipe.Model
+ __call__(text: string): ConllTree[*]

Fig. 4 Semantic core class diagram

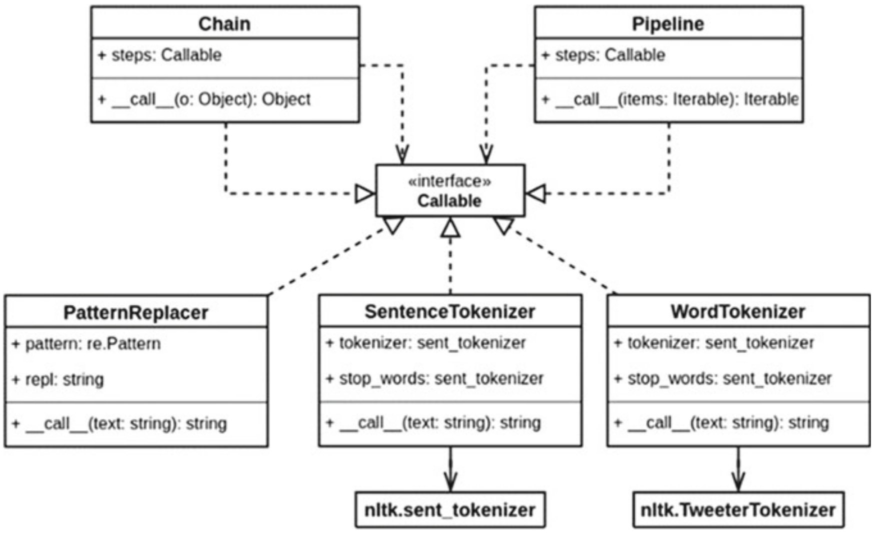

Chain
+ steps: Callable
+ __call__(o: Object): Object
Pipeline
+ steps: Callable
+ __call__(items: Iterable): Iterable
«interface»
Callable
PatternReplacer
+ pattern: re.Pattern
+ repl: string
+ __call__(text: string): string
SentenceTokenizer
+ tokenizer: sent_tokenizer
+ stop_words: sent_tokenizer
+ __call__(text: string): string
WordTokenizer
+ tokenizer: sent_tokenizer
+ stop_words: sent_tokenizer
+ __call__(text: string): string
nltk.sent_tokenizer
nltk.TweeterTokenizer

Fig. 5 Pipeline data processing class diagram

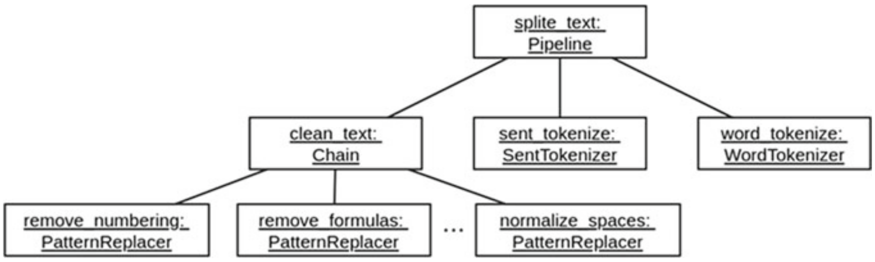


Fig. 6 Diagram of Pipeline data processing objects

In the approach when data is the root cause, the main emphasis is on: storage structures and work with different formats and notations (classes SAO, Conll, ConllTree, etc.) [17] and processors of this data, taking data in one format as input and producing morphological (class TreeTagger), syntactic (UdPipe class), semantic analysis, such as extracting technical functions (SaoExtractor class).

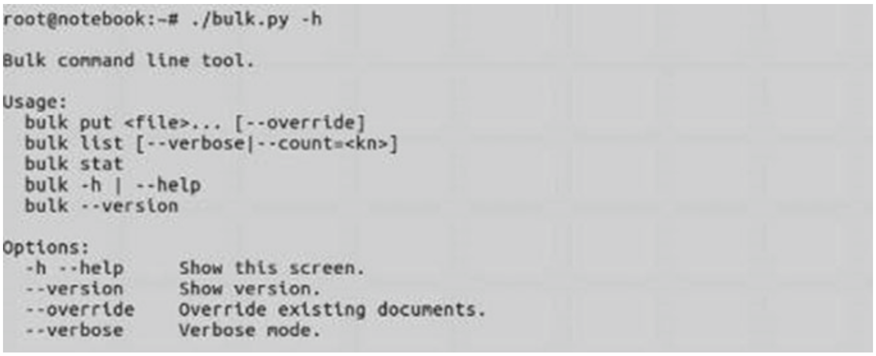
The main concept that is taken as a basis is the Data Pipeline. Most of the existing Batch Processing systems are based on this principle. In this work, we use several third-party libraries based on a data pipeline, which has its implementation specific to the data format they use (matrices, token sequences, etc.). It was decided not to use one of the existing pipelining systems due to their cumbersomeness and specificity, but to implement a simpler and more flexible implementation based on the «Iterator» design pattern and functional programming elements. The main idea is to build a chain of tasks, implemented based on the «Command» design pattern, which allows implementing delayed execution of a certain set of functions. Text processors are examples of this approach (Fig. 5). An illustrated principle has been applied to the entire system.

The diagram shows the text processors «PatternReplacer», «WordTokenizer» and «SentenceTokenizer» that process text at different levels: as a sequence of symbols, words, and sentences respectively. “PatternReplacer” is intended for replacing or deleting blocks of text by a pattern, instances of this class implement their specific functionality, for example, for deleting formulas and numbering paragraphs, as illustrated in Fig. 6. The classes «WordTokenizer» and «SentenceTokenizer» add functionality to similar classes of the NLTK library. «WordTokenizer» combines several words into one for named entities according to the input rules. «Sentence-Tokenizer» segments sentences. These classes are designed to eliminate from the text constructions that do not carry semantic significance in the framework of the problem being solved but negatively affect the correctness of the work of morphological and syntactic analyzers. The classes described above implement the standard «Command» design pattern, which allows you to create deferred objects and build task chains (the «Chain» class) or pipelines (the «Pipeline» class) for processing data. The «Chain» class represents a chain of execution, stores a list of handlers that implement the «Callable» interface, passes the input data (a single object) to the first, its result to the next, and so on along the chain. The Pipeline class implements similar functionality but works with a collection of objects.

3 Research Results

Command-line interfaces for working with the data storage (Fig. 7) and the semantic core (Fig. 8) were implemented.

By the physical effect model, each of its components is described by a regular expression, an example of a description is given in Table 1, for clarity, the pattern is



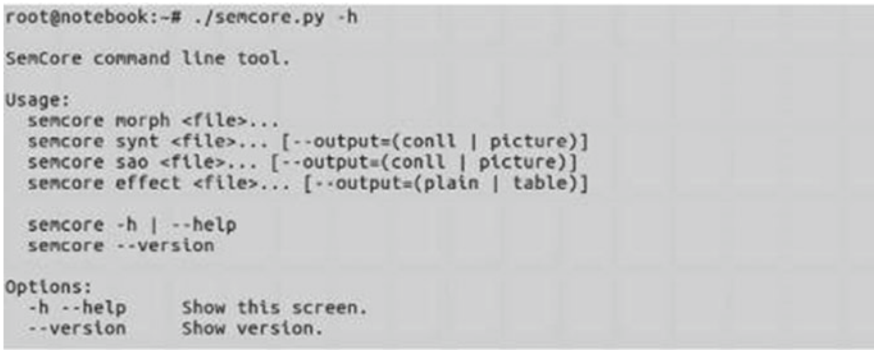
```
root@notebook:~# ./bulk.py -h

Bulk command line tool.

Usage:
  bulk put <file>... [--override]
  bulk list [--verbose|--count=<kn>]
  bulk stat
  bulk -h | --help
  bulk --version

Options:
  -h --help      Show this screen.
  --version      Show version.
  --override     Override existing documents.
  --verbose      Verbose mode.
```

Fig. 7 Data storage command-line interface



```
root@notebook:~# ./semcore.py -h

SemCore command line tool.

Usage:
  semcore morph <file>...
  semcore synt <file>... [--output=(conll | picture)]
  semcore sao <file>... [--output=(conll | picture)]
  semcore effect <file>... [--output=(plain | table)]

  semcore -h | --help
  semcore --version

Options:
  -h --help     Show this screen.
  --version     Show version.
```

Fig. 8 Semantic core command-line interface

Table 1 Description of the physical effect "Ohm's Law"

Component		Description	Pattern
Input	Impact	Electrical field	[weak] electric[al] field
	Characteristic of impact	weak	
	Physical quantity	electric field strength (V/m)	(electric[al] field) density ǀ pressure) ǀ voltage
Output	Impact	electricity	
	Characteristic of impact	Direct, alternating, electronic, mixed, ionic	([alternating ǀ direct ǀ ionic ǀ mixed] [electric[al]] current) ǀ AC ǀ DC
	Physical quantity	current density (A/m**2)	[electric[al]] current density
Object		conductor, semiconductor	[semi]conductor ǀ resistance ǀ resistor

presented in the following form: optional parts of the pattern are presented in square brackets, alternatives are listed, separated by the "ǀ" symbol. Examples of found descriptions of the physical effects are shown in Table 2.

Examples of found descriptions of technical functions in SAO format are shown in Table 3.

The correctness of the algorithms was proved on a test sample prepared manually. Technical functions were extracted from the «Summary of Invention» [18, 19] field of the document, and physical effects were searched for in the «Description» field [20]. The test sample was composed of 60 patent documents and includes 480 technical functions and a description of 20 physical effects, with one document describing only one physical effect.

TF extraction method: accuracy—0.87, completeness—0.77, and F-measure—0.82.

Search for PE description: accuracy—0.92.

Table 2 Examples of searching for descriptions of technical functions in the text of a patent

№	Determined PE	Input data
1	PE № 303 «Thermo-photoelectric effect»	Patent US6380534B1 The amplitude of the Brillouin peaks and the frequency shift of the Brillioun peaks compared with the Rayleigh peak is a measure of the voltage and temperature of the optical fiber at the point from which the light was backscattered
2	PE № 37 « Ohm's law»	Patent US2965301A Conductors, as indicated, are connected to the resistors for application thereto of factor—representing voltages and/or currents and for deriving therefrom an output, all as more fully explained hereinafter

Table 3 Examples of searching for technical functions in the text of a patent

№	Determined SAO	Input data
1	S: method and apparatus A: measure O: temperature and strain within a structure	Patent US6380534B1 …A method and apparatus for measuring the temperature and strain within a structure consists of having optical fibres incorporated in the structure, passing pulses of light down the fibre and detecting the backscattered light…
	S: optical fibres A: pass O: pulses of light down the fibre	
	S: optical fibres A: detect O: backscattered light	
2	S: object of the invention A: provide O: simple and reliable multiplier-divider computer unit of increased capacity.	Patent US2965301A …object of the invention is the provision of a simple and reliable multiplier-divider computer unit of increased capacity. Another object of the invention is the provision of a simple and reliable D.C. multiplier-divider computer unit…
	S: object of the invention A: provide O: simple and reliable D.C. multiplier-divider computer unit	

To test the method of constructing a database of technical functions performed by physical effects, a combination of test and design samples was made, measuring 60 and 10 thousand patent documents, respectively.

We will assume that for a specific document of the test sample, a correspondence between the PE and the TFs implemented by it has been determined, if at least 80% of the technical functions marked by experts in the document and correctly found at the stage of testing the method for extracting TF technical functions were marked in the TF-PE matrix for this PE. This threshold value is introduced in connection with the possible exclusion of technical functions at the stage of reducing their space.

According to the results of testing, the accuracy of extracting technical functions performed by physical effects was 0.78.

4 Discussion

The theoretical value of this work lies in the developed methodology for analyzing graphical representations of mathematical formulas to expand the description of scientific and technical effects and create an automated system on its basis.

5 Conclusion

The reported study was funded by RFBR (research project 18-07-01086), RFBR and Administration of the Volgograd region (projects 19-47-340007, 19-41-340016).

References

1. Orloff, M.: Inventive Thinking Through TRIZ: A Practical Guide, p. 352. Springer, Heidelberg (2006). https://doi.org/10.1007/978-3-540-33223-7
2. Vayngolts, I., Korobkin, D., Fomenkov, S., Golovanchikov, A.: Synthesis of the physical operation principles of technical system. In: Kravets, A., Shcherbakov, M., Kultsova, M., Groumpos, P. (eds.) CIT&DS 2017. CCIS, vol. 754, pp. 575–588. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65551-2_42
3. Korobkin, D., Fomenkov, S., Kravets, A.: Methods for extracting the descriptions of sci-tech effects and morphological features of technical systems from patents. In: IISA 2018 (2018). https://ieeexplore.ieee.org/document/8633624
4. Davydova, S., Korobkin, D., Fomenkov, S., Kolesnikov, S.: Modeling of new technical systems using cause-effect relationships. In: IISA 2018 (2018). https://ieeexplore.ieee.org/document/8633683
5. Korobkin, D., Shabanov, D., Fomenkov, S., Golovanchikov, A.: Construction of a Matrix «Physical Effects – Technical Functions» on the Base of Patent Corpus Analysis. Creativity in Intelligent Technologies and Data Science (CIT&DS 2019), pp. 52–68. (Ser. Communications in Computer and Information Science (CCIS); Volume 1084) (2019)
6. Park, H., Yoon, J., Kim, K.: Identifying patent infringement using SAO based semantic technological similarities. Scientometrics **90**, 515 (2012)
7. Manning, C., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)
8. Yufeng, D., Duo, J., Lixue, J.: Patent Similarity Measure Based on SAO Structure. Chin. Sentence Clause Text Inf. Process. **30**(1), 30–36 (2016)
9. Vasilyev, S., Korobkin, D., Kravets, A., Fomenkov, S., Kolesnikov, S.: Extraction of cyber-physical systems inventions' structural elements of Russian-language patents. Cyber-Physical Systems: Advances in Design & Modelling, pp. 55–68. https://link.springer.com/book/10.1007/978-3-030-32579-4#toc (Book ser. Studies in Systems, Decision and Control (SSDC); vol. 259) (2020)

10. Guo, J., Wang, X., Li, Q., Zhu, D.: Subject–action–object-based morphology analysis for determining the direction of technological change. Technol. Forecast. Soc. Change **105**, 27–40 (2016)
11. Lee, J., Kim, C., Shin, J.: Technology opportunity discovery to R&D planning: key technological performance analysis. Technol. Forecast. Soc. Change **119**, 53–63 (2017)
12. Moehrle, M.G., Walter, L., Geritz, A., Muller, S.: Patent-based inventor profiles as a basis for human resource decisions in research and development. R&D Manag. **35**(5), 513–524 (2005)
13. No, H.J., Lim, H.: Exploration of nanobiotechnologies using patent data. J. Intellect. Prop. **4**(3), 109–129 (2009)
14. Wang, X., Wang, Z., Huang, Y., Liu, Y., Zhang, J., Heng, X., et al.: Identifying R&D partners through subject–action–object semantic analysis in a problem & solution pattern. Technol. Anal. Strateg. Manag. **29**, 1–14 (2017)
15. Wich, Y., Warschat, J., Spath, D., Ardilio, A., König-Urban, K., Uhlmann, E.: Using a text mining tool for patent analyses: development of a new method for the repairing of gas turbines. In: 2013 Proceedings of PICMET 2013 Technology Management in the IT-Driven Services (PICMET), pp. 1010–1016. IEEE, 2013, July
16. Yoon, J., Kim, K.: Identifying rapidly evolving technological trends for R&D planning using SAO-based semantic patent networks. Scientometrics **88**(1), 213–228 (2011)
17. Yoon, J., Kim, K.: Detecting signals of new technological opportunities using semantic patent analysis and outlier detection. Scientometrics **90**(2), 445–461 (2012)
18. Yoon, B., Park, I., Coh, B.Y.: Exploring technological opportunities by linking technology and products: Application of morphology analysis and text mining. Technol. Forecast. Soc. Change **86**, 287–303 (2014)
19. Zhang, Y., Zhou, X., Porter, A.L., Gomila, J.M.V.: How to combine term clumping and technology roadmapping for newly emerging science & technology competitive intelligence: "problem & solution" pattern based semantic TRIZ tool and case study. Scientometrics **101**(2), 1375–1389 (2014)
20. Mel'čuk, I.: Dependency Syntax Theory and Practice. SUNY, New York (1988)