DOI: 10.17323/1998-0663.2019.4.60.72

The design of the structure of the software system for processing text document corpus

Vladimir B. Barakhnin [a,b]
E-mail: bar@ict.nsc.ru

Olga Yu. Kozhemyakina [a]
E-mail: olgakozhemyakina@mail.ru

Ravil I. Mukhamediev [c,d,e]
E-mail: ravil.muhamedyev@gmail.com

Yulia S. Borzilova [a]
E-mail: i.borzilova@alumni.nsu.ru

Kirill O. Yakunin [c,d]
E-mail: yakunin.k@mail.ru

[a] Institute of Computational Technologies, Siberian Branch of the Russian Academy of Sciences
Address: 6, Academician M.A. Lavrentiev Avenue, Novosibirsk 630090, Russia

[b] Novosibirsk State University
Address: 1, Pirogova Street, Novosibirsk 630090, Russia

[c] Satbayev University
Address: 22a, Satbayev Street, Almaty 050013, Kazakhstan

[d] Institute of Information and Computational Technologies
Address: 125, Pushkin Street, Almaty 050010, Kazakhstan

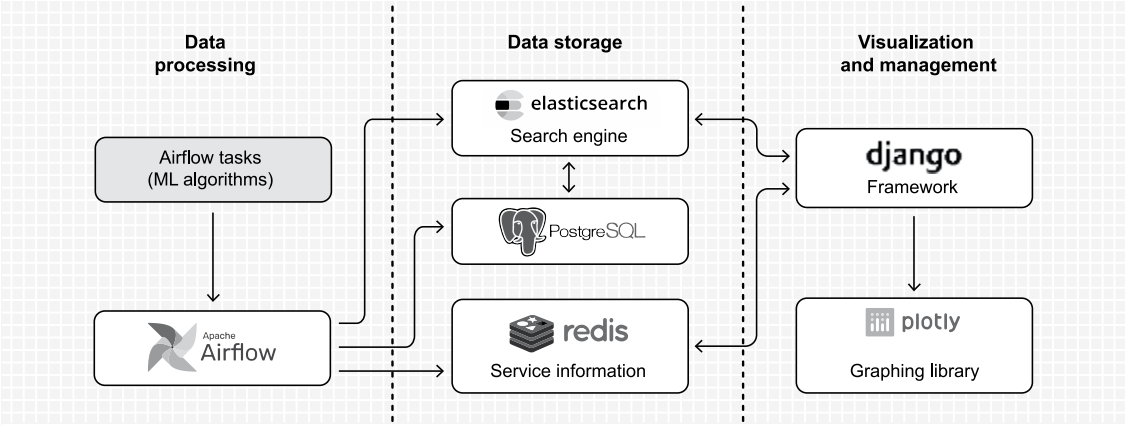
[e] ISMA University
Address: 1, Lomonosova Street, Riga LV-1019, Latvia

Abstract

One of the most difficult tasks in the field of data mining is the development of universal tools for the analysis of texts written in the literary and business styles. A popular path in the development of algorithms for processing text document corpus is the use of machine learning methods that allow one to solve NLP (natural language processing) tasks. The basis for research in the field of natural language

processing is to be found in the following factors: the specificity of the structure of literary and business style texts (all of which requires the formation of separate datasets and, in the case of machine learning methods, the additional feature selection) and the lack of complete systems of mass processing of text documents for the Russian language (in relation to the scientific community-in the commercial environment, there are some systems of smaller scale, which are solving highly specialized tasks, for example, the definition of the tonality of the text). The aim of the current study is to design and further develop the structure of a text document corpus processing system. The design took into account the requirements for large-scale systems: modularity, the ability to scale components, the conditional independence of components. The system we designed is a set of components, each of which is formed and used in the form of Docker-containers. The levels of the system are: the data processing level, the data storage level, the visualization and management of the results of data processing (visualization and management level). At the data processing level, the text documents (for example, news events) are collected (scrapped) and further processed using an ensemble of machine learning methods, each of which is implemented in the system as a separate Airflow-task. The results are placed for storage in a relational database; ElasticSearch is used to increase the speed of data search (more than 1 million units). The visualization of statistics which is obtained as a result of the algorithms is carried out using the Plotly plugin. The administration and the viewing of processed texts are available through a web-interface using the Django framework. The general scheme of the interaction of components is organized on the principle of ETL (extract, transform, load). Currently the system is used to analyze the corpus of news texts in order to identify information of a destructive nature. In the future, we expect to improve the system and to publish the components in the open repository GitHub for access by the scientific community.

Graphical abstract



Key words: natural language processing; streaming word processing; text analysis information system; development of a text corpus processing system.

Citation: Barakhnin V.B., Kozhemyakina O.Yu., Mukhamediev R.I., Borzilova Yu.S., Yakunin K.O. (2019) The design of the structure of the software system for processing text document corpus. *Business Informatics*, vol. 13, no 4, pp. 60–72. DOI: 10.17323/1998-0663.2019.4.60.72.

Introduction

Modern methods of data mining allow us to process a large corpus of text documents (with a volume of more than one million documents) in order to identify certain properties of individual documents included in the corpus, as well as to identify rules characterizing their combination. Since these algorithms involve extracting a wide range of diverse characteristics from texts (which is often a complex task in itself, involving the use of complex but not always high-speed algorithms), it becomes necessary to store the extracted characteristics (along with the documents themselves) in a reference-and-information fund of the software system created. At the same time, the information model of the document repository and their characteristics will largely depend on the type of research text corpus and the nature of the tasks to be solved. For example, systems for processing news messages in order to identify destructive information [1] are significantly different from the systems for processing scientific information [2] and, especially, literary texts (prose and poetry) [3].

It should be noted that one of the difficult tasks is the development of universal tools for the analysis of texts of literary and business styles. As indicated in [4], "when the words have been recognized in a business text, the most significant factor is familiarity with the text (its subject, structure and most frequent words); the keywords and theme elements are recognized relatively well; the end of the text is predictable and well recognized. For a literary text, a large "accent" falls on the initial (preamble) and middle (plot development) compositional fragments and in different ways relates to the components of communicative and semantic division: with the topic for the preamble, with dialogue (especially keywords or rema) for the middle fragment. Thus, speaking about text structures and analysis procedures, we must take into account various types of context, in particular, the functional style, compositional structure and rhetorical connectivity of the text" [4].

Currently, text processing is an actively developing field of IT. A review of works in this area is available, for example, in [5–8]. Note that in the last decade, the main direction of the development of algorithms for processing text documents has been the use of machine learning methods (see, for example, [9–11]). In general, the following approaches can be classified for automatic text analysis [12]:

1. Rule-based with patterns. This approach uses such tools as part-of-speech-taggers and parsers. Another option is to use N-grams to define the frequently used combinations that merged into words. In particular, when solving problems of text sentiment analysis, these N-grams are assigned positive or negative estimates;

2. Unsupervised learning. The main difference from supervised learning is the lack of manual markup for model training. In the case of a statistical model of the corpus of texts, the most weight in the text belongs to such terms that are more often found in this text and at the same time are found in a small number of texts of all sets;

3. Supervised learning. The training set is manually marked up by qualified experts or dataset engineers. Then, the marked-up set is used to train various classifiers, among which the Naive Bayes Classifier [13], Support Vector Classifier (SVM) [14], as well as algorithm ensemble, for example, boosting [15], when several machine learning methods can be combined into an ensemble, in which each subsequent method is trained on the errors of the previous one, and artificial neural networks (ANN) of various configurations [16, 17];

4. Hybrid method. This approach can combine machine learning methods, as well as use rule templates;

5. A method based on graph-theoretic mod-

els. With this approach, the division of the corpus into words is used, with each word having its own weight. Such a weight is used, for example, in problems of sentiment analysis: some words have more weight and more strongly affect the sentiment of the text;

6. Pre-trained models based on deep neural networks (transfer learning), when a pre-trained model is retrained to solve specific problems, for example, the very popular BERT model [18].

In particular, the task of sentiment analysis has been repeatedly solved and is actively used in commercial developments. The latter include, for example, the system of linguistic text analysis of the modular type Eureka Engine, which allows us to extract new knowledge and facts from unstructured data of large volumes[1]. In addition to sentiment analysis, the system solves the problem of the definition of the subject of the text (i.e. classification) and named-entity recognition (NER). The module for automatic classification of texts TextClassifier is implemented on machine learning; there are also modules for automatic determination of named entities, normalization of words, and a morphoanalyzer. The internal structure of the system is not given. The system was used as a tool for sentiment analysis in the media regarding the same event [19]. We can also note work [20], which presents the results of a study of the method of sentiment analysis using the analysis of Twitter messages and reviews of the Kinopoisk portal as an example. The authors used machine learning algorithms as a toolkit: the SVM, the Naive Bayes classifier, and random forest methods. Additionally, the work is providing an overview of similar works in sentiment analysis problems.

A variety of algorithms for NLP suggests the possibility of their implementation in the form of an independent software product. Due to this, the structure of the created software system should be aimed at its interaction with both the end user and the other systems. This article formulates the requirements for the structure of the created software system and defines the role functions of users. After this, the structure we developed is described; this includes a data processing subsystem, a storage and a subsystem for constructing analytical reports.

The main features of the developed system, distinguishing it from comparable systems, are:

1. Automatic thematic modeling, which allows us to identify trends in real time without manually generating a list of keywords or queries. This allows us to automatically identify relevant and socially significant topics online, all of which is critical in making managerial decisions in various fields;

2. Expert marking at the subject level allows us to reduce the volume of objects required for marking (by comparison with use of deep learning networks);

3. From the previous paragraph it follows that prompt and relatively inexpensive markup is possible according to an arbitrary set of criteria, not limited only by semantics. Criteria can be selected individually for the specific requirements of the client (for example, assessment of innovativeness, social significance, opposition, social trust, inflation expectations, etc.).

1. Statement of the problem

The process of text analysis in natural language is described according to the following steps, which analyze the characteristics of the text:

✦ initialization – the formation of the text corpus and its preprocessing for subsequent analysis;

[1] Eureka Engine: http://eurekaengine.ru/

✦ structural analysis (only for poetic texts) – determination of the low-level characteristics of the text (phonetics and metrorhythmics of the poem);

✦ semantic analysis – the definition of semantic constructions taking into account synonymy and named entity linking (NEL). Analysis of scientific texts is usually limited to this level;

✦ pragmatic analysis – definition of genre and style features for literary texts; constructions that determine the destructive impact for news messages, etc.

✦ synthesis of the obtained results – determination of the effect of lower levels on higher, as well as aggregation of results in a convenient form for perception and search.

Let's formulate requirements for the functionality of the system based on its purpose: scraping, storage, streaming analytics and the formation of analytical reports with visualization.

1. Reliable storage of texts corpus of large volumes, while the system must be configured to work with multi-style texts: scientific, journalistic and artistic;

2. Fast parallel access, filtering and aggregation of data for stream processing: preprocessing, building thematic models and classifiers, aggregation and uploading for real-time reports, etc.;

3. Flexibility of the system and the ability to store unstructured and weakly structured data to support storage and access arbitrary data structures for statistical analysis and various computational experiments based on modern text analysis methods.

The structure of the software system should allow us to solve large-scale problems consisting in storing corpus of volumes of several million texts and batch processing online of several thousand documents. Such, for example, is a real-time monitoring project of the Russian-language media of the Republic of Kazakhstan [21] designed to create the following types of reports:

1. The thematic structure of news publications in Kazakhstan Republic electronic media both at the level of major topics (economics, education, politics) and subtopics (pre-school education, a single state exam, higher education and science), and at the level of informational occasions (specific narrow topics that describe a specific event or group of closely related events);

2. Evaluation of individual publications, topics and the media on an arbitrary set of criteria. Such an assessment involves preliminary marking by an expert or a panel of experts;

3. Reports and alerts for identified anomalies. The anomalies are considered at two levels: at the level of dynamics (for example, a sharp increase in publications on a certain topic or a sharp increase in publications with a negative assessment according to the "semantic" criterion) and at the thematic level – the emergence of groups of publications with non-standard, "anomalous" topics that were not earlier met with (for example, the theme of cryptocurrencies, the theme of feminism in Kazakhstan Republic, etc.).

The first two types of reports can be obtained both dynamically and statically (for example, media assessment by certain criteria over the past year). The anomaly report involves an analysis of the dynamics with reference to publication time.

Conceptual design included the formation of the capabilities of the created software system. The created software system should have the following features:

1. Providing access to the texts corpus;

2. Automated processing of the texts corpus stored in the database;

3. Input the characteristics obtained into the repository (database);

4. Flexible planning for various data processing tasks;

5. Statistical processing of the characteristics obtained and their presentation in user-friendly form for the researcher;

6. Updating and improving the algorithms used to analyze the texts corpus.

In the current study the task was set to design the structure of a system for processing natural text corpuses. The scope of this system begins with the analysis of text corpus of a journalistic style. In the future, the scope of the system can be extended to literary texts, due to the modularity of the system and the flexibility of the technologies used.

The designed system consists of the following subsystems:

1. Data processing subsystem. A combination of hybrid methods is used (supervised learning and dictionaries);

2. Data store. To ensure quick user interaction, as well as reduce resource consumption, several types of storages are used;

3. Subsystem for building analytics based on the data obtained.

The information system should take into account the stages of text analysis. The structure of the system consists of the components listed in the description of the problem statement. At the preprocessing stage, the text is pre-processed for further analysis. The preprocessing methods used depend on the algorithm that works with this data (training and analysis). The following types of processing can be classified:

✦ giving as a result of "bag of words"; this type also includes the TF-IDF method;

✦ giving as a result of processing each semantic unit of the corpus (for example, news) its embedding, for example, distribution by tokens / words / phrases / sentences. In this case, it is possible to use recurrent neural networks (RNN);

✦ giving as a result of processing each semantic unit of the corpus one text embedding; for such preprocessing, standard classification methods may be used.

Structural analysis is used for literary style texts and can be performed by currently existing tools, for example, [22]. Semantic analysis can be performed both at the stage of text preprocessing (for example, lemmatization of words), and may not be performed at all — the chosen toolkit will depend on the methods of machine learning and may change over time. Pragmatic analysis in the system is carried out using a combination of machine learning algorithms and compiled frequency dictionaries. The synthesis of the results is ensured by aggregating the results in some storages and outputting these results in a form that will most accurately satisfy the needs of the user.

Based on the capabilities of the system described above, the following requirements for the developed system can be distinguished:

to ensure the operation of subsystems in the form of separate independent components, each of which can be quickly replaced if necessary;

✧ to organize parallelization of calculations, including on several machines;

✧ to implement automated processing of the texts corpus at the request of the user;

✧ monitor tasks in real time, including providing timely reporting of exceptions;

✧ to display data from the analysis of texts in the user interface;

✧ to update the algorithms used in the system to improve the quality of analysis and expand their scope.

2. The system's structure

All components of the system are organized in the Docker containers. All the containers have access to one virtual network, which provides the ability to exchange data using standard network protocols (TCP). Such an implementation ensures the operation of subsystems in the form of independent components, each of which can be replaced if necessary.

The interaction of the components, the subsystem for building analytics and the subsystem for data processing, is carried out using a storage system. The general scheme of the interaction of components is organized on the principle of ETL (extract, transform, load): the user receives a request for data in ElasticSearch (if data is rarely used) or in Redis (if data is often used). In addition, the processing subsystem uses the Airflow scheduler, which records in Redis information on the distribution of tasks by workers; they, in turn, report to Redis on the status of their tasks. During the design process, components can be used according to their intended purpose.

Visualization of the system structure is shown in the *Figure 1.*

Analysis of text corpus (at this stage – corpus of news messages in Kazakhstan's Russian-language Internet media in the amount of 1.5 million documents with constant replenishment) is carried out by loading "workers". New documents are uploaded to the data processing subsystem using a special parser: at this stage, the download is done manually at the user's request, in the future the receiving new news will be configured according to the schedule (jobs running). With a given frequency, reports will be generated that require a lot of compu-

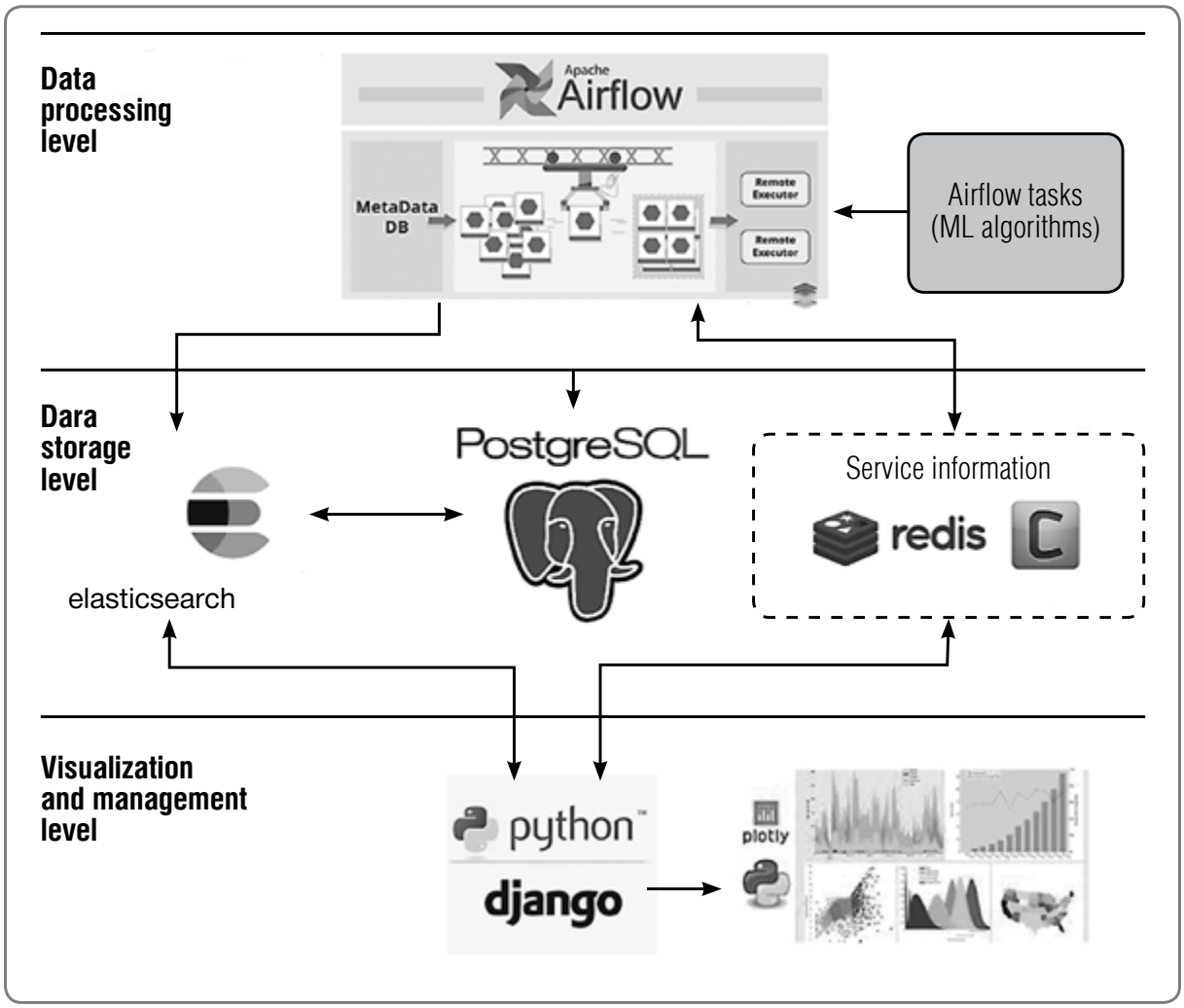


Fig. 1. Structure of the system developed

tational time; the results will be placed in the repository (this approach will reduce the waiting time for results from the data processing subsystem). Based on the data collected, additional model training will be performed (1–2 times per month), which will include recalculation of the set of key characteristics of the text corpus. In the event that additional training of the model does not lead to an increase in the accuracy index (for example, in the task of semantic analysis), the use of other ML-algorithms or their combination is provided.

The role system includes the following roles:

1. Custom user – has access to the basic functionality of the system: search, filtering, digital information panels (dashboards);

2. Advanced user – has access to custom reports, automatic alerts about "hot topics", the ability to filter by named entities (for example, person, organization, region) in articles. Such separation of users is due to subsequent use of the system by government bodies;

3. Developer – has access to the Airflow admin panel and to the repository where the Airflow DAG is stored. He can add and change his tasks, run and track their implementation;

4. Administrator – super-user, has a full set of rights to work with the system.

The role model is shown in *Figure 2*.

The following subsections describe the selected tools for each of the listed subsystems in more detail.

2.1. The data processing subsystem

During the analysis, Apache Airflow, an open source software platform, was chosen to these needs. The main components of this platform:

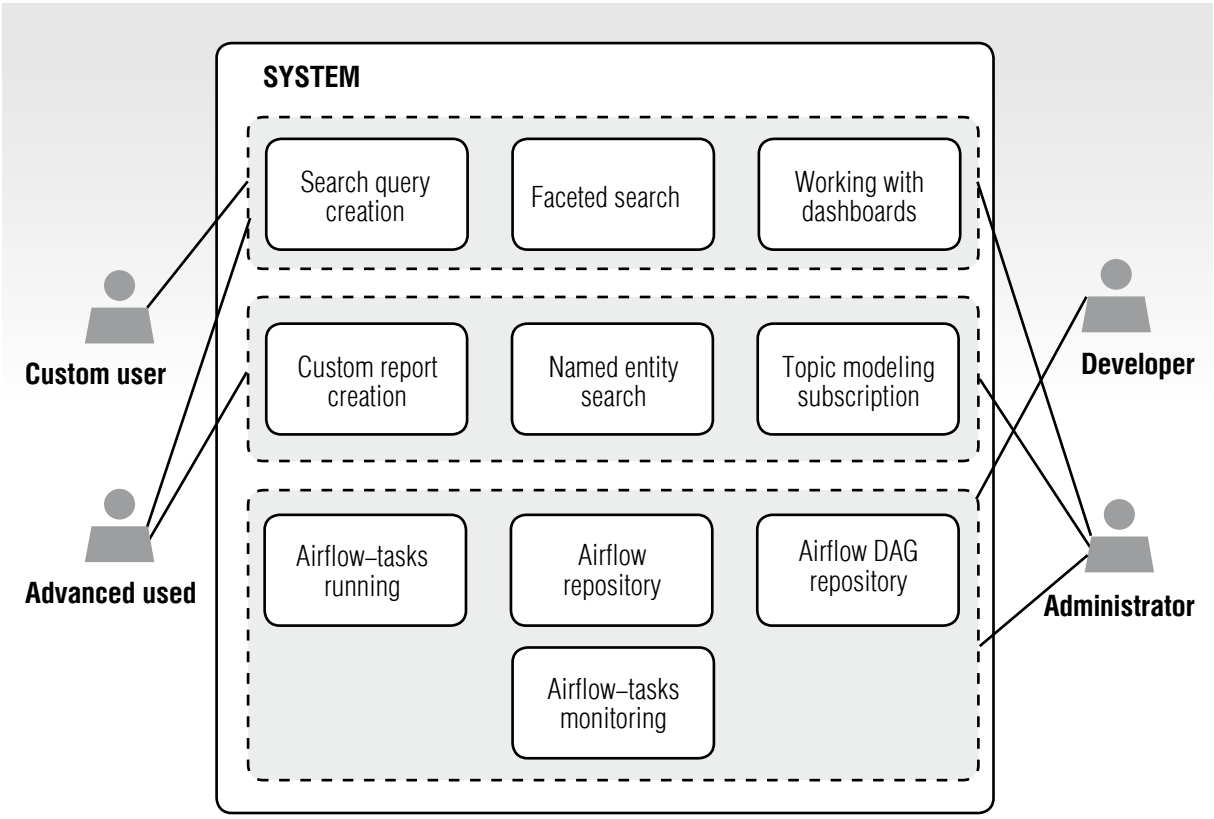


Fig. 2. The role model of the system

1. Airflow-worker – the main component that performs data processing. It can be scaled horizontally, including to individual servers or cloud virtual machines. In the current version of the architecture, the necessary dependencies are built into the Airflow-worker container image in advance. However, in principle, the process of dependency injection can occur in various ways, including by dynamically obtaining Docker containers from public or private repositories;

2. Airflow-scheduler – a component responsible for assigning tasks to Airflow-workers in the order defined by Airflow DAG. Airflow DAG is a non-cyclic directed graph that describes the order in which certain tasks are performed, and also contains information about the schedule, priorities, behavior in case of exceptions, etc.;

3. Airflow web server – a web interface that allows us to monitor and control the progress in tasks.

The machine learning algorithms are implemented in the system as separate Airflow tasks.

2.2. The data storage

There are three storage types in the system provided:

1. PostgreSQL – acts as a persistent storage for structured data. Its use is due to the wide capabilities of this relational database (among freeware) and interaction with a wide range of tools. The main data types stored in this database:

✦ news and metadata;

processed data at the level of different basic units of analysis (token / word / phrase / sentence / text), including vectorization, results of lemmatization, cleaning, etc.;

✦ the results of thematic modeling;

✦ the results of the classification of news on various grounds (semantic, politicization, social significance, etc.).

2. ElasticSearch – in-memory NoSQL storage designed for storing unstructured or weakly structured data, as well as quick search (including full-text) and filtering and streaming access. Compared with other NoSQL databases for storing documents with an arbitrary structure, such as MongoDB and CouchDB, ElasticSearch stands out with advanced tools for indexing text, which allow for full-text search in large volumes of documents in almost real time. Also, due to the possibility of constructing advanced indexes for data, it is possible to perform complex aggregations in the database itself, including distributed. ElasticSearch performs several functions:

✦ main storage for access, retrieval and filtering of data by the end user;

✦ main storage for ETL (extract, transform. Load) data processing processes, including the recording of any intermediate results in free form;

✦ storage for caching certain calculation results necessary for building dashboards and reports in the system;

ElasticSearch duplicates data stored in PostgreSQL as persistent storage, since ElasticSearch is an in-memory database without guarantees regarding data persistence and integrity.

3. Redis – a fast key-value storage used to cache individual pages and items, as well as to cache authorization sessions. Redis stores service data, as well as a cache of pages and items that are accessed frequently.

All three primary storages of the system can be easily scaled to several separate computers. Both horizontal scaling and replication are supported, with ElasticSearch and Redis showing near-linear increases in horizontal scaling performance.

A separate PostgreSQL cluster is used to store service data, such as task execution states. To run and track the progress of tasks, a bunch of Celery + Redis is used.

2.3. The subsystem for constructing analytical reports

The interface of the subsystem presents as an HTML + CSS + JS website with access via HTTP. The choice of the HTML + CSS + JS technology stack for the interface is justified by the fact that it is the web interfaces that are the most common and universally supported technology for building user interfaces, with the ability to access from any devices and operating systems from anywhere in the world, provided there is a web browser and Internet connection.

The web application is realized in the Django framework (Python), Gunicorn acts as the web server; the reverse proxy is Nginx. The web application has access to both the PostgreSQL persistent repository and ElasticSearch. Django has a built-in Cache Framework that allows us to cache pages and page elements in Redis. For example, if it is assumed that the page will be visited frequently, and it takes a long time for reading (for example, three seconds), then it is better to cache such a page in Redis, which will speed up access to the necessary data.

The Django framework was chosen for the following reasons:

1. The ability to quickly Agile-develop a web interface and data storage model. The development speed with Django is significantly higher than with competing products like Spring (Java), Yii (PHP) and Node.js (JavaScript);

2. Due to the project's involving the analysis of data and the construction of machine learning models, including for NLP, Python is the best choice, since most of the "state-of-the-art" models and ML/AI and NLP methods are developed by the community specifically in Python;

3. Django ORM works better with a PostgreSQL database.

The web application implements a series of pages for filtering, searching and accessing various dashboards and reports. At the first stage of the implementation of the system, dashboards are calculated in advance manually. With further development of the system, faceted search from Elastic Search will be used. The Plotly data visualization library will be used to create the graphs.

Examples of information that graphs can display are:

Dynamics by tonality (as well as manipulativeness, politicization, etc.), topics, number of views and comments, filtered by media, topics, authors, tags (including full-text search);

Distribution of topics, tonality values, etc. in statics, with filtering and search;

Identification of anomalies for analytical reports (the hottest topics, etc.).

Conclusion

This article formulates the requirements for the structure of a software system designed to process large (over one million units) corpus of text documents, including, in particular, the implementation of automated processing of corpus of texts, the ability to parallelize calculations and the preparation of analytical reports. The user role functions are defined. The structure of the software system was developed, including a data processing subsystem based on the Apache Airflow service, several types of storages that provide quick access to system components, and a subsystem for building analytical reports, which is generated in the Python Django application using the Plotly visualization library. The flexibility of the system allows us to select a different ensemble of machine learning algorithms, providing an increase in the quality and accuracy of the analysis of corpus of text documents.

Currently, the system is used to analyze news text corpus for the purpose of comparative analysis of news media corps in the Republic of Kazakhstan. ■

Acknowledgments

This work was funded by grant No BR05236839 of the Ministry of Education and Science of the Republic of Kazakhstan, by the Russian Fund of Basic Research, project No 19-31-27001 and within the framework of the state task theme No AAAA-A17-117120670141-7 (No 0316-2018-0009).

References

1. Barakhnin V.B., Kuchin Ya.I., Muhamedyev R.I. (2018). On the problem of identification of fake news and of the algorithms for monitoring them. Proceedings of the *III International Conference on Informatics and Applied Mathematics, Almaty, Kazakhstan, 26–29 September 2018*, pp.113–118 (in Russian).
2. Shokin Yu.I., Fedotov A.M., Barakhnin V.B. (2010) Technologies for construction of processing software systems dealing with semistructured documents aimed at information support of scientific activity. *Computational Technologies*, vol. 15, no 6, pp. 111–125 (in Russian).
3. Barakhnin V.B., Kozhemyakina O.Yu., Borzilova Yu.S. (2019) The development of the information system of the representation of the complex analysis results for the poetic texts. *Vestnik NSU. Series: Information Technologies*, vol. 17, no 1, pp. 5–17 (in Russian). DOI: 10.25205/1818-7900-2019-17-1-5-17.
4. Bolshakova E.I., Klishinskii E.S., Lande D.V., Noskov A.A., Peskova O.V., Yagunova E.V. (2011) *Automatic natural language text processing and computer linguistics.* Moscow: MIEM (in Russian).
5. Pang B., Lee L., Vaithyanathan S. (2002) Thumbs up? Sentiment classification using machine learning techniques. Proceedings of the *2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002), Philadelphia, PA, USA, 6–7 July 2002*, pp. 79–86. DOI: 10.3115/1118693.1118704.
6. Choi Y., Cardie Cl., Riloff E., Patwardhan S. (2005) Identifying sources of opinions with conditional random fields and extraction patterns. Proceedings of the *Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT 2005). Vancouver, British Columbia, Canada, 6–8 October 2005*, pp. 355–362.
7. Manning C.D. (2011) Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? Proceedings of the *12th International Conference "Computational Linguistics and Intelligent T ext Processing" (CICLing 2011), Tokyo, Japan, 20–26 February 2011*, pp. 171–189.
8. Mukhamedyev R., et al. (2020) Assessment of the dynamics of publication activity in the field of natural language processing and deep learning. Proceedings of the *4th International Conference on Digital Transformation and Global Society, St. Petersburg, Russia, 19–21 June 2019*. Springer, 2020 (in press).
9. Tarasov D.S. (2015) Deep recurrent neural networks for multiple language aspect-based sentiment analysis. *Computational Linguistics and Intellectual Technologies: Proceedings of Annual International Conference "Dialogue–2015"*, no 14 (21), vol. 2, pp. 65–74.
10. Garcia-Moya L., Anaya-Sanchez H., Berlanga-Llavori R. (2013) Retrieving product features and opinions from customer reviews. *IEEE Intelligent Systems*, vol. 28, no 3, pp. 19–27. DOI: 10.1109/MIS.2013.37.
11. Mavljutov R.R., Ostapuk N.A. (2013) Using basic syntactic relations for sentiment analysis. Proceedings of the *International Conference "Dialogue 2013", Bekasovo, Russia, 29 May – 2 June 2013*, pp. 101–110.
12. Prabowo R., Thelwall M. (2009) Sentiment analysis: A combined approach. *Journal of Informetrics*, vol. 3, no 2, pp. 143–157. DOI: 10.1016/j.joi.2009.01.003.
13. Dai W., Xue G.-R., Yang Q., Yu Y. (2007) Transferring naive Bayes classifiers for text classification. Proceedings of the *22nd National Conference on Artificial intelligence (AAAI 07). Vancouver, British Columbia, Canada, 26–27 July 2007*, vol. 1, pp. 540–545.
14. Cortes C., Vapnik V. (1995) Support-vector networks. *Machine Learning*, vol. 20, no 3, pp. 273–297. DOI: 10.1023/A:1022627411411.

15. Friedman J.H. (2001) Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, vol. 29, no 5, pp. 1189–1232.

16. Zhang G.P. (2000) Neural networks for classification: A survey. *IEEE Transactions on Systems, Man, and Cybernetics. Part C (Applications and Reviews)*, vol. 30, no 4, pp. 451–462.

17. Schmidhuber J. (2015) Deep learning in neural networks: An overview. *Neural Networks*, no 61, pp. 85–117. DOI: 10.1016/j.neunet.2014.09.003.

18. Devlin J., Chang M.-W., Lee K., Toutanova K. (2018) BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805.*

19. Vladimirova T.N., Vinogradova M.V., Vlasov A.I., Shatsky A.A. (2019) Assessment of news items objectivity in mass media of countries with intelligence systems: The Brexit case. *Media Watch*, vol. 10, no 3, pp. 471–483. DOI: 10.15655/mw/2019/v10i3/49680.

20. Romanov A.S., Vasilieva M.I., Kurtukova A.V., Meshcheryakov R.V. (2018) Sentiment analysis of text using machine learning techniques. Proceedings of the *2nd International Conference " R. Piotrowski's Readings in Language Engineering and Applied Linguistics (Saint-Petersburg, 2017)*, pp. 86–95 (in Russian).

21. Barakhnin V.B., Mukhamedyev R.I., Mussabaev R.R., Kozhemyakina O.Yu., Issayeva A., Kuchin Ya.I., Murzakhmetov S.B., Yakunin K.O. (2019) Methods to identify the destructive information. *Journal of Physics: Conference Series*, vol. 1405, no 1. DOI: 10.1088/1742-6596/1405/1/012004.

22. Barakhnin V.B., Kozhemyakina O.Y., Zabaykin A.V. (2014) The algorithms of complex analysis of Russian poetic texts for the purpose of automation of the process of creation of metric reference books and concordances. *CEUR Workshop Proceedings*, vol. 1536, pp. 138–143.

About the authors

Vladimir B. Barakhnin

Dr. Sci. (Tech.), Associate Professor;

Leader Researcher, Institute of Computational Technologies,
Siberian Branch of the Russian Academy of Sciences,
6, Academician M.A. Lavrentiev Avenue, Novosibirsk 630090, Russia;

Professor, Faculty of Information Technologies, Novosibirsk State University,
1, Pirogova Street, Novosibirsk 630090, Russia;

E-mail: bar@ict.nsc.ru

ORCID: 0000-0003-3299-0507

Olga Yu. Kozhemyakina

Cand. Sci. (Philol.);

Senior Researcher, Institute of Computational Technologies,
Siberian Branch of the Russian Academy of Sciences,
6, Academician M.A. Lavrentiev Avenue, Novosibirsk 630090, Russia;

E-mail: olgakozhemyakina@mail.ru

ORCID: 0000-0003-3619-1120

Ravil I. Mukhamediev

Dr. Sci. (Eng.);

Professor, Satbayev University, 22a, Satbayev Street, Almaty 050013, Kazakhstan;

Leader Researcher, Institute of Information and Computational Technologies,
125, Pushkin Street, Almaty 050010, Kazakhstan;

Professor, ISMA University, 1, Lomonosova Street, Riga LV-1019, Latvia;

E-mail: ravil.muhamedyev@gmail.com

ORCID: 0000-0002-3727-043X

Yulia S. Borzilova
Doctoral Student, Institute of Computational Technologies,
Siberian Branch of the Russian Academy of Sciences,
6, Academician M.A. Lavrentiev Avenue, Novosibirsk 630090, Russia;
E-mail: i.borzilova@alumni.nsu.ru
ORCID: 0000-0002-8265-9356

Kirill O. Yakunin
Doctoral Student, Satbayev University, 22a, Satbayev Street, Almaty 050013, Kazakhstan;
Developer Engineer, Institute of Information and Computational Technologies,
125, Pushkin Street, Almaty 050010, Kazakhstan;
E-mail: yakunin.k@mail.ru
ORCID: 0000-0002-7378-9212