



Article

Hybridization of Intelligent Solutions Architecture for Text Understanding and Text Generation

Anton Ivaschenko ^{1,*}, Arkadiy Krivosheev ¹, Anastasia Stolbova ² and Oleg Golovnin ^{2,*}

¹ Computer Science Department, Samara State Technical University, 443100 Samara, Russia; arkas19@gmail.com

² Information Systems and Technologies Department, Samara National Research University, 443100 Samara, Russia; stolbova.aa@ssau.ru

* Correspondence: anton.ivashenko@gmail.com (A.I.); golovnin@ssau.ru (O.G.)

Featured Application: Document workflow systems and enterprise content management systems that implement intelligent technologies for text understanding.

Abstract: This study proposes a new logical model for intelligent software architecture devoted to improving the efficiency of automated text understanding and text generation in industrial applications. The presented approach introduces a few patterns that provide a possibility to build adaptable and extensible solutions using machine learning technologies. The main idea is formalized by the concept of expounder hybridization. It summarizes an experience of document analysis and generation solutions development and social media analysis based on artificial neural networks' practical use. The results of solving the task by the best expounder were improved using the method of aggregating multiple exponents. The quality of exponents' combination can be further improved by introducing the pro-active competition between them on the basis of, e.g., auctioning algorithm, using several parameters including precision, solution performance and score. Analysis of the proposed approach was carried out using a dataset of legal documents including joint-stock company decision record sheets and protocols. The solution is implemented in an enterprise content management system and illustrated by an example of processing of legal documentation.

Keywords: text understanding; text generation; artificial neural networks; software architecture; auctioning; legal documents analysis



Citation: Ivaschenko, A.; Krivosheev, A.; Stolbova, A.; Golovnin, O.

Hybridization of Intelligent Solutions Architecture for Text Understanding and Text Generation. *Appl. Sci.* **2021**, *11*, 5179. <https://doi.org/10.3390/app11115179>

Academic Editor: Askhat Diveev

Received: 29 April 2021

Accepted: 29 May 2021

Published: 2 June 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Modern technologies of machine learning are widely used to improve the quality of text understanding—an engineering discipline that consists of extracting the explicit or implicit meanings from texts formed in natural languages. Another area of their application is text generation that provides new capabilities of documents' automatic creation using the knowledge base of content patterns, design samples and textual fragments. Both text understanding and text generation require a deep understanding of the problem domain, which makes it problematic to develop a universal software platform for industrial applications.

To solve this problem and make the transition from scientific research to technology that provides guaranteed results of machine learning application in practice, we propose unified software architecture in this paper. The presented approach introduces a few patterns that provide a possibility to build adaptable and extensible solutions using machine learning technologies. It summarizes an experience of document analysis and generation solutions' development and social media analysis based on artificial neural networks' implementation in practice.

The paper has the following structure. On the basis of the state-of-the-art overview, the formal concept of “exponents' hybridization” is presented, that introduces the main idea of neural networks' combination. The following section describes the details of its

implementation in the form of object-oriented patterns visualized by UML class diagrams. The results sections illustrate its implementation by the example of legal documents' processing. Finally, some evaluation results are provided that allow understanding the benefits of the proposed approach.

2. State-of-the-Art

The problem of automated understanding of text, as a separate domain of information analysis, emerged quite a long time ago, however, it was finally formed into a separate scientific direction by the 1980s [1]. Nowadays, intelligent text understanding is implemented to solve various problems [2]. For example, text analysis algorithms are used to monitor social media in order to identify and prevent various events [3,4]. Basic features of the implementation of the morphological analysis of informal texts on the Internet are considered in [5]. Data processing in natural language is also used in the problem of machine translation of texts or speech without human intervention, using the construction of deep neural networks [6].

The main challenge is concerned with understanding the semantics of the text [7]. To correctly determine the meaning of each passage, the reader and author must have a similar context formed by the background knowledge. Understanding the context is necessary, on the one hand, to improve the quality of text analysis, and on the other hand, to extract implicit information. Early approaches addressed this problem by defining a kind of knowledge structure, building a plan or scenario. However, later, general classes of inference were identified that do not depend on individual knowledge structures [8], which provided a transition to patterns for establishing connections between the concepts of the context.

The classical approach to text understanding uses a semantic analyzer and two static semantic resources: combinatorial vocabulary and ontology [9]. The vocabulary contains a variety of information about words, and the ontology stores non-linguistic knowledge about concepts and serves as a meta-language for semantic description. Words and concepts are provided with semantic descriptions, which consist of definitions in a formal language.

In general, legal documentation analysis is one of the perspective areas of text understanding applications. Solutions based on the use of ontologies containing knowledge about the subject area are actively used to detect and eliminate errors in the consistency of the extracted data, including legal documents [10]. In the context of legal documentation processing, the automatic search for relevant legal provisions in the written description of events is a challenging problem [11]. The method for classifying legal texts is based on characteristic word analysis. To organize the connection between legal provisions and characteristic words, court decisions are taken as a training set of documents, since the corresponding provisions can be accurately extracted from them. The corresponding relationship is then established using the TF-IDF algorithm. Experiments show that the algorithm can extract keywords from various legal texts and classify them into the corresponding legal terms. A method for automatic classification of scientific texts [12] is based on data compression. Experiments have shown that the method correctly identifies topics of scientific texts with a probability of 75–95%.

The transition to patterns and knowledge bases in logical inference made it possible to use artificial intelligence technologies, e.g., neural networks. When using them, the mechanism of “attention” is highlighted, being a key component at the stage of training a neural network. The attention engine models how important each part of the original sentence is. To compute importance scores, the attention engine summarizes the information and then creates a context vector. Experimental results in [13] show that paying attention to words significantly improves the quality of the final solution.

Text features should also be considered. Short passages are not easy to process with conventional methods due to incorrect or abbreviated syntax and the lack of statistical signals [14]. Concept dependencies can be used to find the hidden semantics of short

texts. Neural networks with short-term memory or the forgetting effect can capture and memorize dependencies between terms, presenting good results.

Additional information helpful for text analysis can be obtained by clustering nouns and verbs following their meanings [15]. Applying this approach requires building a vocabulary containing word roots and endings that give a grammatical feature to words. Vocabulary is used to build a network of word meanings. The network reflects the relationship between lexemes of nouns and verbs, all of which are hierarchically combined into groups that have specific features and common properties.

Understanding the semantic correlations between the text and associated images, video and audio recordings, geotags, etc., has great potential for improving text understanding systems. However, understanding multimedia information still remains an unsolved research problem, although there are already some successes in this area [16]. Implementation of algorithms of big data and deep learning combined with enormous computational capabilities support the technical progress in the field of text understanding [17]. The essence of such approaches is to combine semantic and big data-based representations to more accurately identify the meaning. The evaluations of these approaches show that they significantly outperform traditional benchmarks.

The capabilities of automatic text understanding are also important for solving the problems of standard documents' development and processing. The authors of [18] show that neural networks easily generate texts, but the resulting passages cannot come close to human-created documentation. The use of templates, copying and modification solves this problem and significantly improves the generation results. Thus, most research in the field of text generation does not take into account the context and structure of the text that should be obtained. The research in [19] presents the architecture of a neural network, which includes the selection and planning of content and leads to the division of the generation problem into two stages: first, a content plan is created and it is highlighted which information should be mentioned and in what order, and then a document is already generated taking into account this plan.

A new text generation system is proposed in [20]. This system creates a new story based on a series of stories entered. Two options for generating new stories were considered: based on stories with different plots and characters, and based on the same stories, but from different volumes of the book, where the storyline is in the same context and the characters are similar. The results obtained by the system are analyzed according to the following parameters: the correctness of grammar, the relationship of events, the level of interest and uniqueness.

One of the tasks of text generation is to compare the content of existing documents. Content analysis is used to find recommended documents with similar semantic content. In [21], a new knowledge-based technique for calculating inter-document similarity is proposed, called semantic context analysis. Reference [22] presents an in-depth study of the technology for calculating the similarity of documents.

Based on the analysis of the state-of-the-art and some experience in the area of semantic analysis and enterprise content management [23–25], we have identified a gap in intelligent text understanding and text generation application to practice. This gap consists of the lack of adaptive software architecture capable of combination and extension of artificial intelligent solutions according to the developing situations and changing problem statements, without retraining of the neural networks. To solve this problem, a new approach was developed for intelligent text analysis and text generation applications.

3. The Concept of Expounders Hybridization

The proposed approach is based on the concept of expounder—a new pattern that encapsulates the basic functionality of text pre-processing and post-processing and can involve intelligent algorithms of text understanding. Expounder provides an adaptation of the intelligent solution. In case the existing algorithms provide low efficiency, there can be new expounders generated in manual or automatic mode to improve the logic.

This process is described by expounder hybridization. Let $A = \{a_1, a_2, \dots, a_n\}$ be a sequence of work items (jobs) that form a pipeline for processing in FIFO order.

All elements of the conveyor $a \in A$ are subject to processing by expounder $e_i \in E$ with number i from the set of all available exponents E to obtain a complex solution S_i , combined of individual working units s_i^a :

$$\begin{aligned} \forall a \in A : e_i(a) &\rightarrow s_i^a, \\ S_i &= \bigcup_{a \in A} s_i^a, \\ s_i^a &= \langle NN_j, DS_k, CL \rangle, \end{aligned} \quad (1)$$

where

NN_j is the selected neural network with number j from the set of available neural networks $NN_j \in NN$.

DS_k is the selected dataset with number k for training the neural network from the set of all available datasets, $DS_k \in DS$.

CL is a list of commands, which is generated depending on the selected neural network and dataset: $CL = f : NN_j, DS_k \rightarrow \{C_1, C_2, \dots, C_m\}$, in which C_i is one atomic instruction.

In the case when processing of some job $a \in A$ is impossible, $s_i^a = \emptyset$, the expounder e_i is replaced with the next one in order $e_{i+1} \in E$, and then the processing process (1) is repeated.

In the case when several exponents are able to fulfill the job A and start competition for it, the system performs an auction, which allows the identification of the winner with the best solution (2). For this, the evaluation function f is introduced and the time period τ is designated, during which the evaluation of the expounder e_i for the performance of A is valid:

$$W_{e_i}^\tau(A) = \frac{1}{n} \sum_{a \in A} f([e_i(a)]), \quad (2)$$

where $W_{e_i}^\tau(A)$ is the value of expounder e_i performing job A , valid for time τ , and n is the number of units a_i in job A .

The estimate W_{e_i} remains valid during the interval τ , and must be recalculated after it expires. The implementation of such a restriction makes it possible to ensure the stable structure of the expounder and guarantee the result only for a given period of time while opening up opportunities for the development of the expounder if the decision to use it in this task is delayed for external reasons that inevitably arise when working in a multi-user, multi-threaded mode with a shared resource in a single information space.

The function of estimating $f(a)$ in (2) depends on the class of problems to be solved, e.g., it can be the probability of information recognition, accuracy (value, inverse error) or the time of operations execution.

In situations where none of the exponents $e \in E$ can finalize the entire job A , it requires hybridization. For this purpose, we represent each expounder $e \in E$ by a set of pairs of neural network NN and dataset DS :

$$e = \{ \langle NN_1, DS_1 \rangle, \langle NN_2, DS_2 \rangle, \dots, \langle NN_m, DS_m \rangle \} \quad (3)$$

Hybridization is carried out in one of the following ways (see Figure 1):

- Transition—replacement of one ineffective expounder element with an effective expounder element, for example, one neural network $e_i : NN'_i$ with another $e_j : NN''_j$ while saving the dataset. The borrowed element is replaced by another: $e_i : NN'_i \leftrightarrow e_j : NN''_j$.
- Transversion—replacement of several elements of the expounder with elements of another expounder, i.e., exchange of neural network–dataset pairs: $e_i : NN'_i, DS'_i \leftrightarrow e_j : \langle NN''_j, DS''_j \rangle$.

- Insertion—insertion by copying the effective elements of one expounder into another expounder $e_j : \langle NN'''_j, DS''_j \rangle \rightarrow e_i : \langle NN'''_j, DS''_j \rangle, e_j : \langle NN'''_j, DS''_j \rangle$.
- Deletion—exclusion of ineffective (unused) elements from the expounder $e_i : NN'_i, DS'_i \rightarrow e_i : \emptyset$.

The choice of the hybridization method is performed in ascending order of their number, which corresponds to the complexity and influence on the result: from minor permutations to significant changes in the composition of the expounder elements.

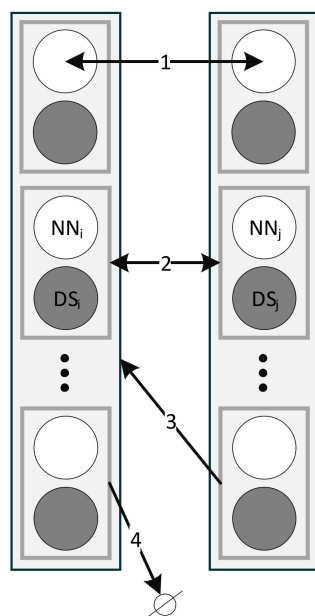


Figure 1. Expounder hybridization scheme.

The operation of deletion allows reducing a significant increase in the composition of expounders, which inevitably occurs during insertion. Hybridization is performed cyclically until the result is achieved—a hybrid expander is obtained that is able to most efficiently perform the entire job, A.

The choice of expounders for hybridization is carried out in descending order of estimate (4), i.e., first, an attempt is made to improve the most effective option at the expense of the next most effective option. The choice of specific elements of the expounder for their exchange or exclusion is based on estimates for these specific elements, similarly to (4).

In case hybridization of a set of expounders does not lead to a result, then two options are possible: in the first option, the best available one is taken as a satisfactory result, while the second approach requires the involvement of a data scientist. It is assumed that in the process of adapting a set of expounders, they will be developed to solve a set of problems in a specific subject area, which will reduce the participation of a data specialist to a minimum.

4. Implementation

The proposed approach can be implemented in the form of specifically designed patterns or as a part of the logic of a multi-agent system. The second option provides automatic generation of expounders and organization of their interaction in the process of text understanding. Both solutions require unified software architecture capable of supporting the process of expounder hybridization at the software level.

Below, a logical model of software architecture is introduced for intelligent solutions of text understanding and text generation based on the concept of expounders' hybridization, presented in the form of UML class diagrams.

The basic object that is analyzed in the system is a document (see Figure 2, where 1..* and 0..* identify cardinality in UML notation). The document can be uploaded to the system in scanned form (.pdf, .jpeg), as well as in the form of text (.doc). Depending on the type of corporate documentation, the content of the document is usually structured by sections. For example, such legal documents as contracts, protocols and explanatory notes contain the sections: title, preamble, introduction, subject of the contract, etc.

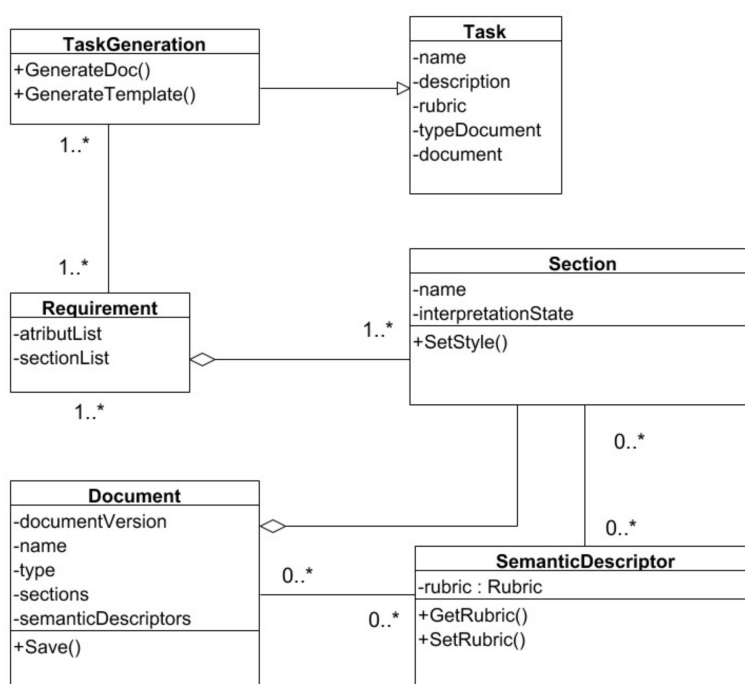


Figure 2. Text understanding and text generation basic entities.

Each section of the document, as a rule, contains a text typical for a given type of documentation and a set of attributes that are unique for a particular document. For example, the attributes include the date of the contract, the name of the customer, the name of the contractor and the cost of work.

- The problem of text understanding is solved by the following stages:
- Optical document recognition,
- Classification,
- Markup,
- Data extraction.

If the original document is presented as an image, it becomes necessary to extract text from it. Different methods of data extraction are used to work with different types of documents.

Expounder is a basic component that implements the logic related to document processing within the framework of text understanding. The logical model is presented in Figure 3. Expounder implementation contains a queue of elements available for processing in a pipeline. The elements of the pipeline are work units and jobs, for the processing of which solutions are created with the necessary set of neural networks, datasets and commands to be executed. If the solution succeeds, then the corresponding expounder gets a new item from the queue. In the event that the job has not been fulfilled, the expounder tries to find a more suitable expounder or creates a new one.

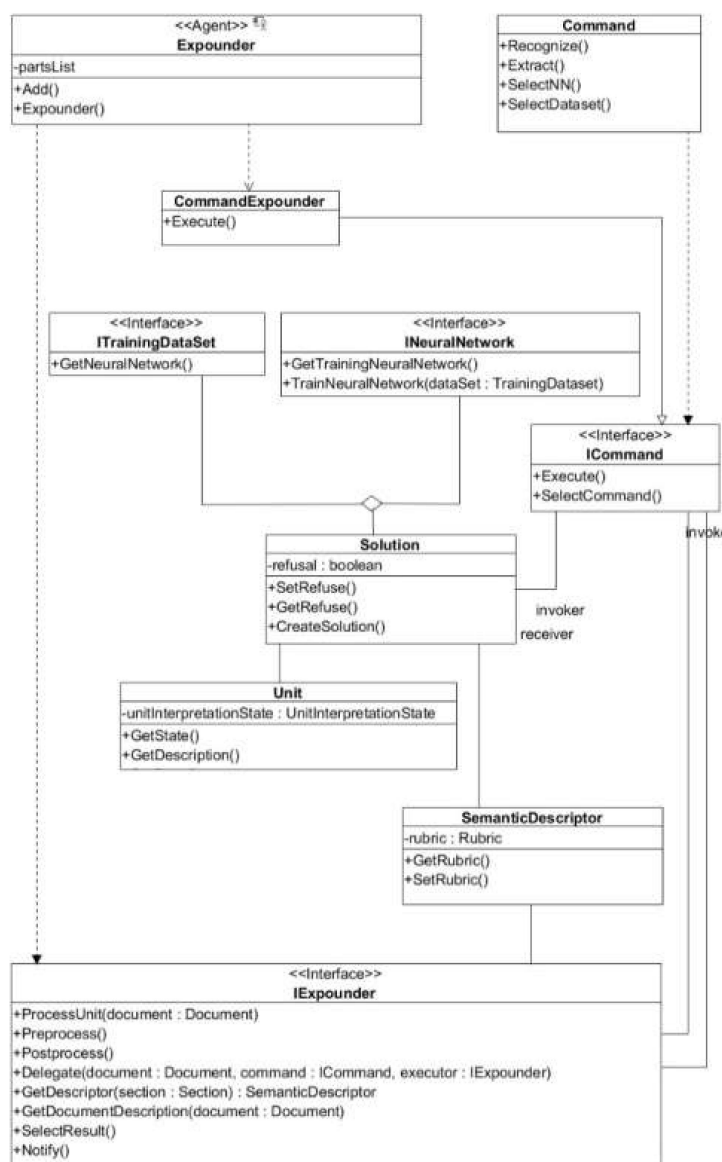


Figure 3. Expounder pattern implementation.

In order to provide the dynamical process of expounders' collaboration, task understanding is not performed in real-time. For each new job, there is a new instance of Command created, which is assigned to a corresponding Expounder. Expounder, in turn, generates a Solution that involves a neural network trained by a specific Dataset. In case the Solution does not cover the requirements of the initial Command, a new Command is generated and assigned to a new Expounder.

The logic of expounder hybridization is implemented by the builder presented in Figure 4. The expounder can act as a classifier or recognizer. While the process of document analysis proceeds, efficient expounders remain active in the system while those worked out are reduced.

The problem of text generating is solved using content analysis. Documents are generated based on templates consisting of a set of working units. Each work unit in the system has a status. The following relationships apply between the templates:

- Inheritance—templates can inherit each other.
- Implementation—a document can implement several templates in different sections.

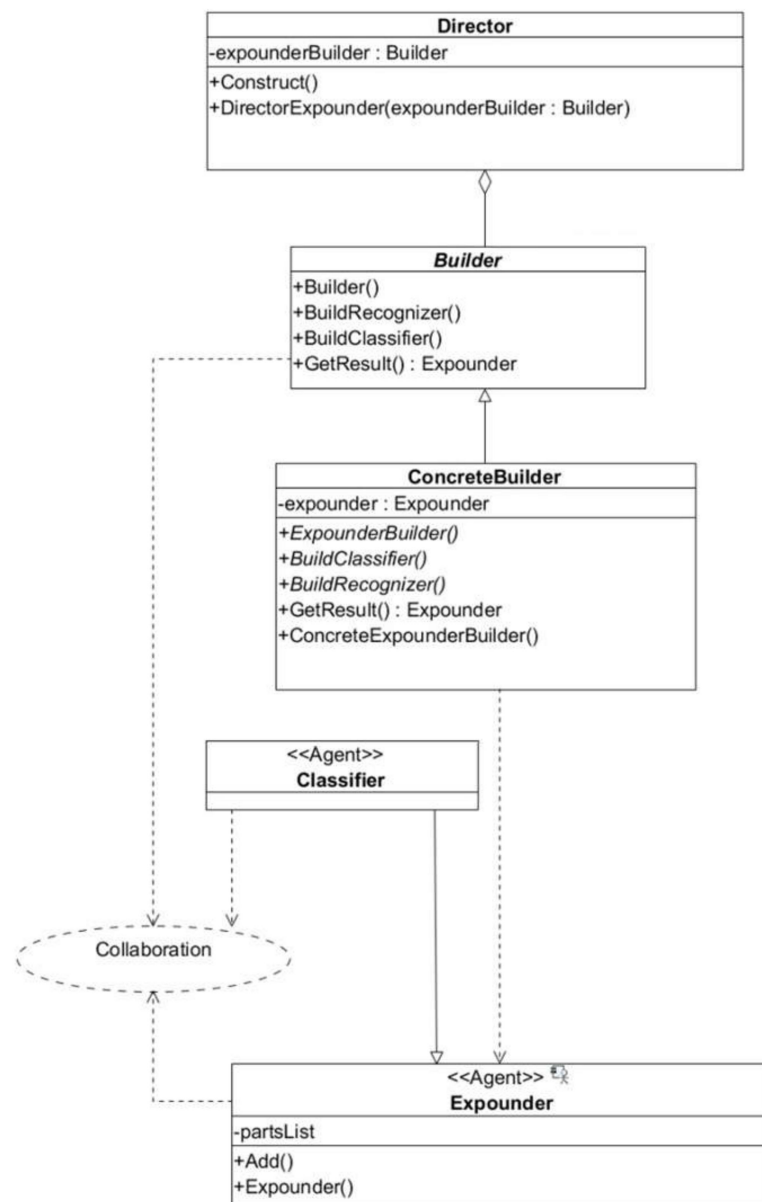


Figure 4. Expounder hybridization patterns.

The logical model supporting text generation is presented in Figure 5. The aggregator is a component that implements the creation of new documents based on existing templates.

A new document can be generated either based on another document (by equivalent) in the system or using the selected template (by template).

When generating by equivalent, the same rules are applied to the new document as to the parent one: design, sections and attributes will match. When generating a new document, you can specify data sources (for example, other documents), from which information will be pulled into a new document to fill in the attributes. When generating by template, sections and styles from different templates are aggregated into a new document.

Under this process, paraphrasing techniques are applied to enhance the originality of the text.

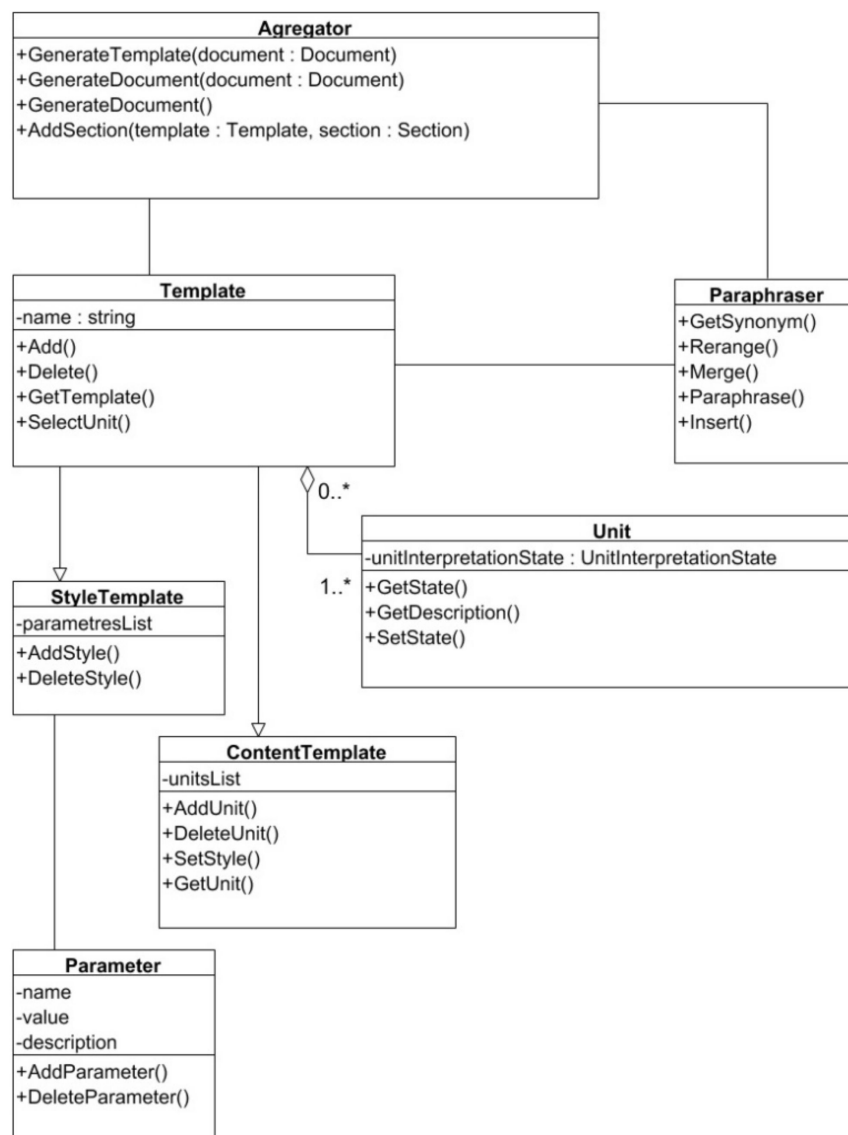


Figure 5. Text generation patterns.

5. Results

The proposed solution based on expounder hybridization was implemented in software for understanding legal documents. Currently, this solution is being probated in a number of Enterprise content management systems. This section presents an example that illustrates the benefits of the proposed approach.

Legal documentation turns out to be a good case in point to compare the concerned algorithms. It contains regulatory documents, agreements and protocols that are usually well-structured and operate with strict and logical formulations. The clauses need to be unambiguously interpreted by several parties, and therefore are based on a unified terminology. At the same time, the textual form of the documents obstruct easy interpretation by automatic algorithms and becomes a challenge for intelligent solutions.

However, the area of the proposed approach possible application is not limited by legal documents. The solution which is successful in this area can be easily expanded to other problem domains, such as, e.g., design and production documentation, medical records, social media, etc.

The considered source documentation was represented by a number of documents, which include various types of contracts. As a rule, at the end of the agreement, the details

of the involved companies are indicated. Requisites may include data such as the name of the company (full and short), contract party, legal entity sign, address, phone number and others. As part of the research, a module has been developed aimed at processing these details.

Initially (see Example 1), a Base Extractor is created for each attribute, and a value is extracted from the corresponding graph based on the rules. This solution allows to extract the specified attributes from the document, but it has a significant drawback: the short name of the company is extracted from the document, but in addition to the short name, the full name of the company is also written into it.

Example 1. *Legal document details extraction.*

```
<contract_side1>
<side1_TRRC span="12976, 13220">770801001</side1_TRRC>
<side1_TIN span="12976, 13205">7705401340</side1_TIN>
<side1_type_name span="12976, 13006">Customer</side1_type_name>
<side1_is_legal>0</side1_is_legal>
<side1_short_name span="12976, 13194">Some company name (short name)</side1_short_name>
<side1_address span="12976, 13298">Address</side1_address>
<side1_phone span="12976, 13375">Phone number</side1_phone>
</contract_side1>
```

That is, the applied extraction rules do not allow for correctly extracting and then separating different types of names in documents of this type. So, there is a need to refine the solution to correctly extract the names of the company from the block of details for this type of document.

The expounder creates a new solution based on the Combining Extractor (see Example 2). This new solution includes the ability to extract details according to the existing rules, but with the additional elaboration of the name of the company.

Example 2. *Legal document details extraction (modified).*

```
<contract_side1>
<side1_TRRC span="12976, 13220">770801001</side1_TRRC>
<side1_TIN span="12976, 13205">7705401340</side1_TIN>
<side1_type_name span="12976, 13006">Customer</side1_type_name>
<side1_is_legal>0</side1_is_legal>
<side1_full_name span="12976, 13194">Full company name</side1_full_name>
<side1_short_name span="12976, 13194">Short company name</side1_short_name>
<side1_address span="12976, 13298">Address</side1_address>
<side1_phone span="12976, 13375">Phone number</side1_phone>
</contract_side1>
```

The quality of expounders' combination can be further improved by introducing the pro-active competition between them on the basis of, e.g., the auctioning algorithm. To provide it, the control expounder should retain the characteristics of all the other expounders. The list of characteristics used for auctioning includes the following parameters, calculated for the extraction or each type of attributes:

- Precision, ε .
- Solution performance, v (number of processed sentences per second).
- Score, representing the given measure of comparison $\sigma = \varepsilon^2 \cdot v$.

The calculated characteristics of expounders are used to determine their priority in the mechanism of the auctioning. The characteristics of the expounders are adjusted during the self-learning process of the control expounder.

The auctioning algorithm includes the following stages:

1. The control expounder sends all expounders one document for training.
2. Precision of the expounders is calculated.
3. The obtained values are “frozen”, and the entire dataset is sent to the expounders for processing.
4. The accuracy of the expounders is calculated.
5. A new document from the dataset is added to the training set.
6. Repeat steps 2–4.

Expounder hybridization was implemented as a part of enterprise content management solution being applied for text understanding and text generation based on machine learning technologies designed for automated analysis, examination and generation of project documentation when creating new projects, as well as distribution best practices between design companies and departments.

6. Evaluation

Analysis of the proposed approach was carried out using a dataset of legal documents including joint-stock company decision record sheets and protocols. The experiment task was to extract the pre-defined list of attributes from these documents, which includes the following:

Proper names (full name),

- Names of companies,
- Addresses,
- Dates.

The test dataset contains 200 documents. The text was preliminarily extracted from the scanned images using the Tesseract OCR program (version 5.0.0-alpha.20200328) [26] and then split into sentences using the Segmenter module from the Natasha library [27]. The experiments were carried out at a stand with the following characteristics: Intel core i9-10900, 32 GB DDR4, 512 GB SSD, NVIDIA GeForce RTX 2070 SUPER.

The following versions of libraries were used to extract the attributes:

- Natasha 1.4.0 [27],
- Spacy 3.0.5 [28],
- SDK Pullenti 3.23 [29],
- Stanza 1.2 [30],
- DeepPavlov 0.14.0 [31].

Using the library of extracting routines, one control expounder and five specialized expounders were created according to the approach introduced above. Each expounder starts the execution of experiment tasks according to incoming requests. The process starts from the control expounder. If it cannot solve the task itself, then it delegates it to other expounders or creates a new one. The decision of task delegation is made on the basis of the expounder’s evaluation, and several parameters can be used to select the best one.

For the considered experiment, the results of expounders’ evaluation are presented in Figure 6. The expounders are evaluated and compared by two parameters: performance and accuracy.

The results of the experiment showed that the fastest is the solution based on the Spacy library (80.94 sentences per second with an accuracy of 0.46), and the most accurate is the solution based on the Natasha library (12.17 sentences per second with an accuracy of 0.53). Such a low accuracy of each solution is associated with the presence of typos in the text, resulting from the work of the Tesseract OCR.

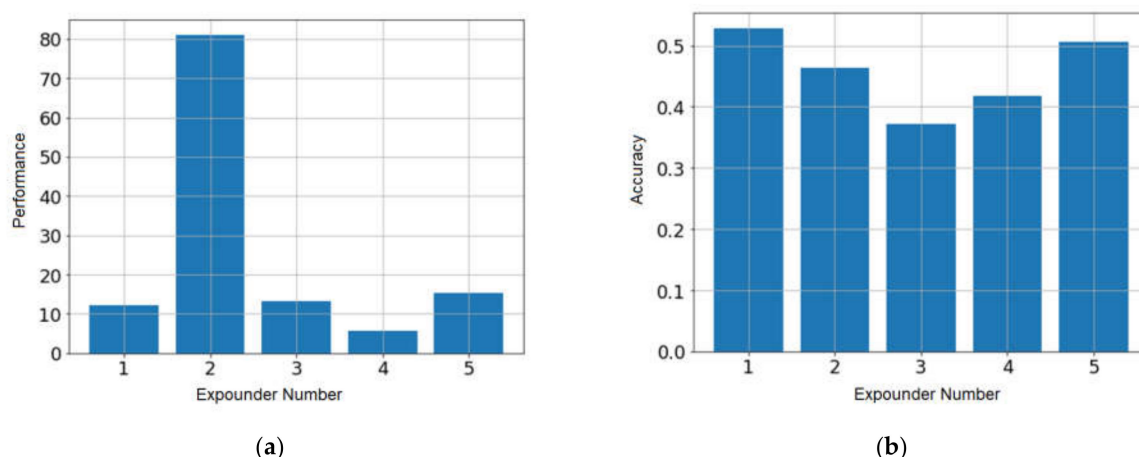


Figure 6. Comparative evaluation of the expounders: (a) performance, (b) accuracy.

The results of solving the task by the best expounder were improved using the method of aggregating multiple expounders. It was proposed to choose the best individual solution for each extracted attribute. The results of the experiment showed that document processing performance decreases and the accuracy increases with an expansion of aggregated solutions (see Figure 7). Therefore, when aggregating solutions obtained as a result of launching five expounders, the document processing performance is 2.43 sentences per second, and the accuracy is 0.85.

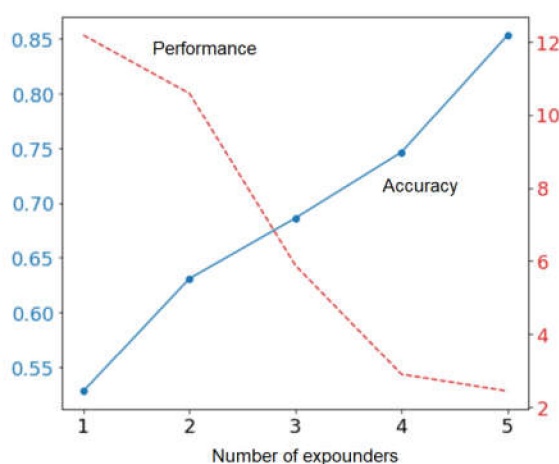


Figure 7. Accuracy versus performance dependency from the number of expounders used.

Implementing the auctioning approach is illustrated by the following cases of expounders' hybridization, which were tested under the framework of the described experiment.

Case 1: An expounder based on the Natasha library is installed as a priority expounder. If any type of attribute processing fails, the control expounder calls the expounder that has the highest priority for this type of attribute.

The result of using combination 1 is shown in Figure 8. The efficiency of the solution when choosing the best expounder in terms of accuracy is 0.67, and the processing performance is 6.34 sentences per second. These characteristics are equal for both using the precision and score to prioritize the expounders.

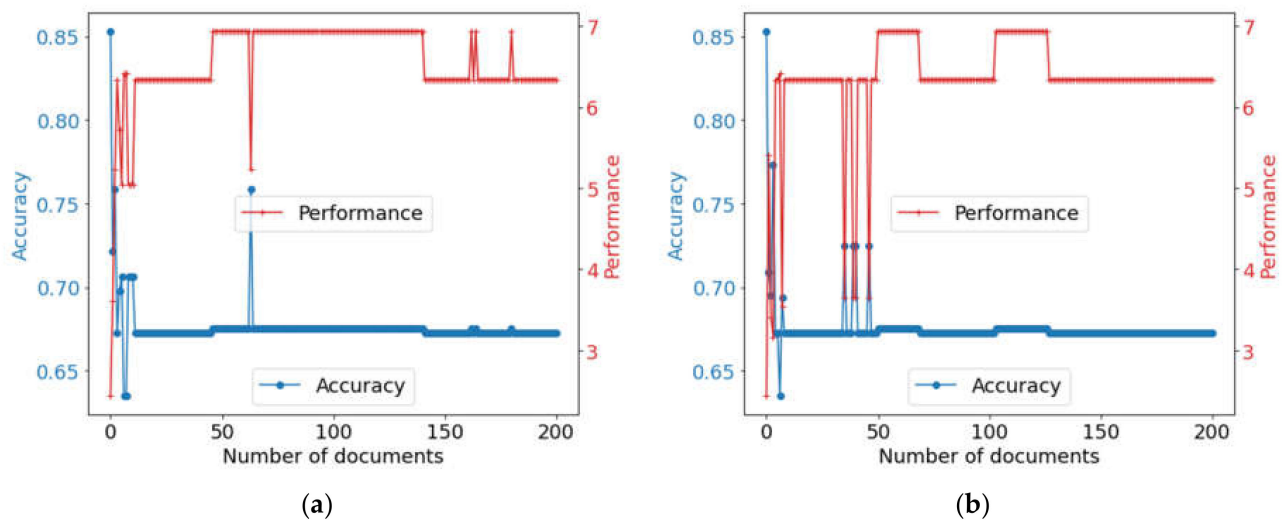


Figure 8. Case 1 auctioning results: (a) prioritization by accuracy, (b) prioritization by performance.

Case 2: In addition to case 1, the auctioning algorithm is extended in the following way. Other expounders are additionally called with a frequency equal to their accuracy for each attribute type. The results are presented in Figure 9. For expounders' prioritization by precision, the resulting accuracy was 0.71, and the performance was 5.09 sentences per second. When choosing an expounder by score, the accuracy of the solution is 0.67, and the processing performance is 6.3 sentences per second.

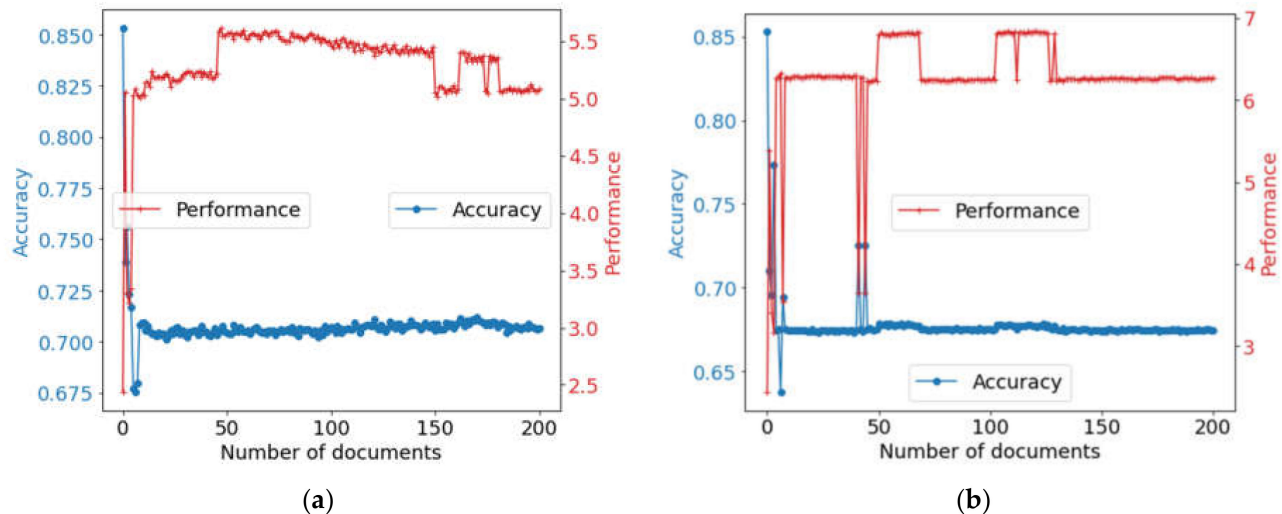


Figure 9. Case 2 auctioning results: (a) prioritization by accuracy, (b) prioritization by performance.

Case 3: To extract each type of attribute, the expounder is called, which provides the highest extraction accuracy. The result is shown in Figure 10. For expounders' prioritization by precision, the resulting accuracy was 0.69, and the performance was 6.78 sentences per second. When determining the priority by score, the absolute accuracy is 0.77, and the processing speed is 6.25 sentences per second.

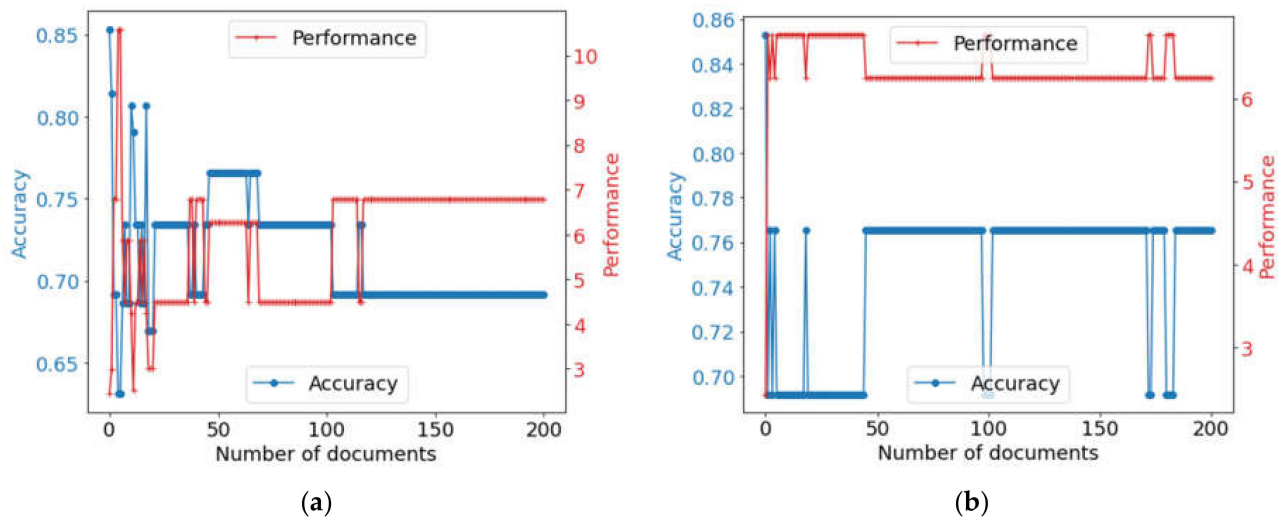


Figure 10. Case 3 auctioning results: (a) prioritization by accuracy, (b) prioritization by performance.

Case 4: In addition to case 3, the algorithm is extended in the following way. The remaining expounders can be called with a probability equal to the accuracy for a given attribute type. The results of using the combination are shown in Figure 11. For expounders' prioritization depending on precision, the performance was 3.45 sentences per second, and the accuracy was 0.81. When using the score for prioritization, the accuracy was 0.71, and the performance was 5.85 sentences per second.

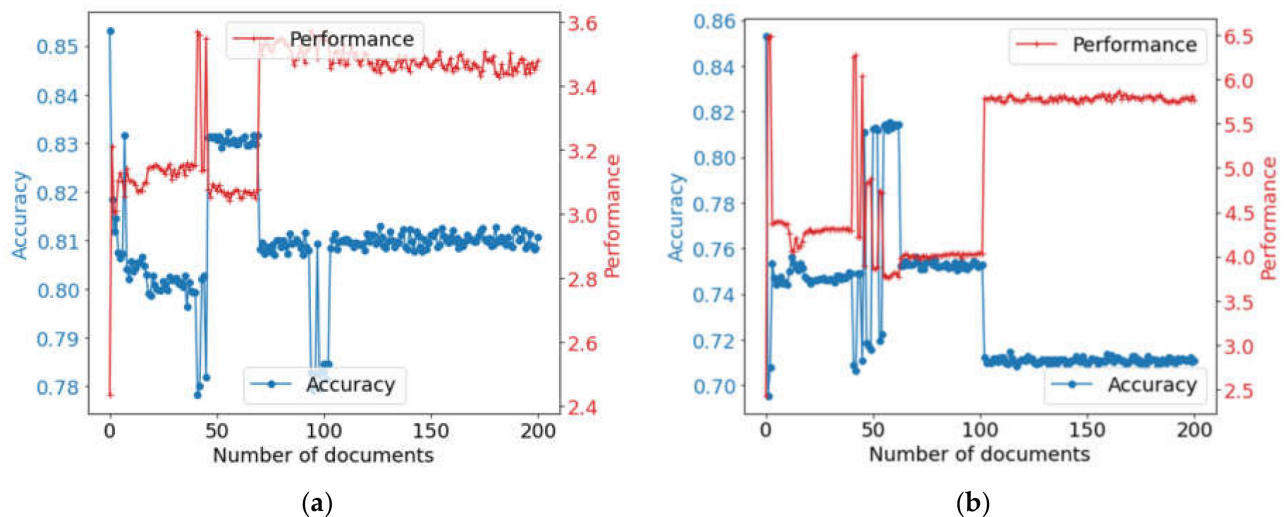


Figure 11. Case 4 auctioning results: (a) prioritization by accuracy, (b) prioritization by performance.

Experimental results are generalized in Figure 12. The histograms show a comparison of all listed options of the combination of the solutions. The cases were compared by performance and accuracy. The comparison results show that the option with aggregation of solutions is the most accurate (0.85) but at the same time, the slowest (2.43 sentences per second). Case 3 presents the optimal result in the case of choosing the best expounder by score: the solution accuracy is 0.77 with the performance of 6.25 sentences per second. Additionally, in this case, about 50 documents are enough to train the control expounder.

In turn, cases 1 and 2 are slightly inferior in accuracy and performance, but require a small number of documents, with at least 25 being enough for training the control

expounder. For case 4, the solution is not stable, converges poorly and requires a large number of more than 100 documents for training.

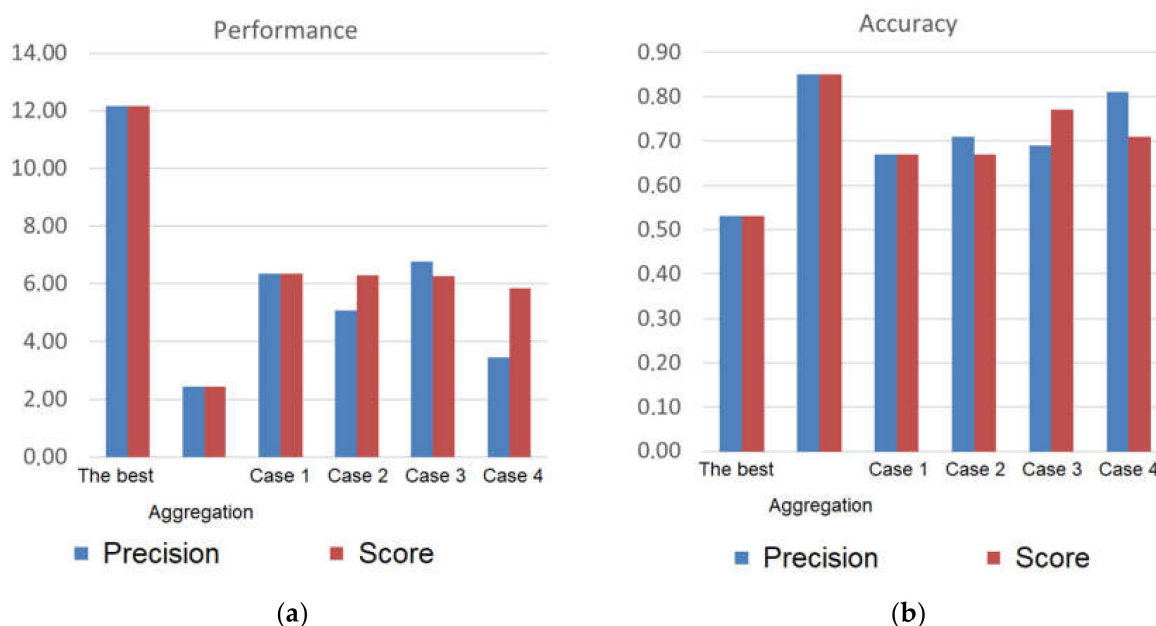


Figure 12. Experimental results: (a) performance, (b) accuracy.

7. Discussion

The most obvious way to improve the quality of intelligent text understanding is to choose a more adequate neural network model and retrain it with a more complete and more representative dataset. However, these changes lead to a complete reconstruction of the logic and can reduce the quality of the algorithm for previously successful cases. In addition, the constant expansion of training samples leads to a loss of recognition quality.

Therefore, some developers of practical applications came to an idea of combination of multiple neural networks. Herewith, supporting an extended number of intelligent algorithms requires additional efforts and computational costs. The solution can be optimized by implementing a heuristic to choose only one type of neural network according to the specifics of the problem being solved. For example, the document type and content can be considered to choose the most perspective technology of text understanding and text generation.

Nevertheless, combination of multiple neural networks in practical applications requires supporting software architecture. The proposed unified approach based on expounder patterns can help solving this problem. Compared to the solution of choosing only the algorithm which has the highest performance but lowest accuracy, as well as the solution of aggregation characterized by high accuracy and low performance, implementing the auctioning approach helps in finding the efficient solution considering the combination of both objectives.

In general, the proposed approach has fairly broad prospects for practical use in the development of distributed intelligent software. The described field of text understanding and text generation is not unique, but illustrates the achieved benefits in the best way.

8. Conclusions

Implementation of the proposed patterns allows increasing the usability and efficiency of intelligent technologies' application for text understanding and text generation. The model of expounder hybridization provides new opportunities to develop adaptive soft-

ware using multi-agent technologies and knowledge bases, such as a possibility to combine a variable set of neural networks instead of developing only the one.

The next steps are concerned with the evaluation of the proposed approach application in enterprise content management systems.

Author Contributions: Conceptualization, A.I.; Data curation, A.S.; Investigation, A.K. and A.S.; Methodology, O.G.; Software, A.K.; Supervision, A.I. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Norvig, P. *Inference in Text Understanding*; AAAI Press: Seattle, WA, USA, 1987; pp. 561–565.
2. Zhang, X.; LeCun, Y. Text understanding from scratch. *arXiv* **2015**, arXiv:1502.01710.
3. Coppersmith, G.; Leary, R.; Crutchley, P.; Fine, A. Natural Language Processing of Social Media as Screening for Suicide Risk. *Biomed. Inform. Insights* **2018**, *10*, 117822261879286. [[CrossRef](#)] [[PubMed](#)]
4. Ahmad, S.; Varma, R. Information extraction from text messages using data mining techniques. *Malaya J. Mat.* **2018**, *5*, 26–29. [[CrossRef](#)]
5. Fenogenova, A.; Kazorin, V.; Karpov, I.; Krylova, T. Automatic morphological analysis on the material of Russian social media texts. *EPiC Ser. Lang. Linguist.* **2019**, *4*, 11–17. [[CrossRef](#)]
6. Rishita, M.V.S.; Raju, M.A.; Harris, T.A. Machine translation using natural language processing. In *MATEC Web of Conferences*; EDP Sciences: Les Ulis, France, 2019; Volume 277, p. 02004.
7. Graesser, A.; Tipping, P. Understanding Texts. In *A Companion to Cognitive Science*; Blackwell Publishing: Malden, MA, USA, 2017; pp. 324–330.
8. Britton, B.K.; Graesser, A.C. *Models of Understanding Text*; Psychology Press: London, UK, 2014; 376p.
9. Boguslavsky, I. Semantic descriptions for a text understanding system. In *Computational Linguistics and Intellectual Technologies. Papers from the Annual International Conference “Dialogue”*; RSUH: Moscow, Russia, 2017; pp. 14–28.
10. Buey, M.G.; Román, C.; Garrido, Á.L.; Bobed, C.; Mena, E. Automatic Legal Document Analysis: Improving the Results of Information Extraction Processes Using an Ontology. In *Intelligent Methods and Big Data in Industrial Applications*; Springer: Cham, Switzerland, 2019; pp. 333–351.
11. Li, Z. A Classification Retrieval Approach for English Legal Texts. In *Proceedings of the 2019 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS)*, Changsha, China, 2019, 12–13 January; pp. 220–223.
12. Selivanova, I.V.; Ryabko, B.Y.; Guskov, A.E. Classification by compression: Application of information-theory methods for the identification of themes of scientific texts. *Autom. Doc. Math. Linguist.* **2017**, *51*, 120–126. [[CrossRef](#)]
13. Wu, L.; Tian, F.; Zhao, L.; Lai, J.; Liu, T.-Y. Word attention for sequence to sequence text understanding. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, New Orleans, LA, USA, 2–7 February 2018; pp. 5578–5585.
14. Hu, F.; Xu, X.; Wang, J.; Yang, Z.; Li, L. Memory-Enhanced Latent Semantic Model: Short Text Understanding for Sentiment Analysis. In *Database Systems for Advanced Applications*; Springer: Cham, Switzerland, 2017; pp. 393–407.
15. Zupan, J. Graph theoretical view on text understanding. *Informatica* **2018**, *42*, 85–93.
16. Otto, C.; Springstein, M.; Anand, A.; Ewerth, R. Understanding, Categorizing and Predicting Semantic Image-Text Relations. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, Ottawa, ON, Canada, 10–13 June 2019; pp. 168–176.
17. Chatterjee, A.; Gupta, U.; Chinnakotla, M.K.; Srikanth, R.; Galley, M.; Agrawal, P. Understanding Emotions in Text Using Deep Learning and Big Data. *Comput. Hum. Behav.* **2019**, *93*, 309–317. [[CrossRef](#)]
18. Wiseman, S.; Shieber, S.; Rush, A. Challenges in Data-to-Document Generation. *arXiv* **2017**, arXiv:1707.08052.
19. Puduppulli, R.; Dong, L.; Lapata, M. Generating data into text with content selection and planning. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, Honolulu, HI, USA, 27 January–1 February 2019; pp. 6908–6915.
20. Pawade, D.; Sakhapara, A.; Jain, M.; Jain, N.; Gada, K. Story Scrambler—Automatic Text Generation Using Word Level RNN-LSTM. *Int. J. Inf. Technol. Comput. Sci.* **2018**, *10*, 44–53. [[CrossRef](#)]
21. Benedetti, F.; Beneventano, D.; Bergamaschi, S.; Simonini, G. Computing inter-document similarity with Context Semantic Analysis. *Inf. Syst.* **2019**, *80*, 136–147. [[CrossRef](#)]

22. Minshan, R.; Hongxia, C. Research on mass text similarity detection based on simhash algorithm. *Metrol. Meas. Tech.* **2018**, *4*, 25.
23. Surnin, O.L.; Sitnikov, P.V.; Ivaschenko, A.V.; Ilyasova, N.Y.; Popov, S.B. Big Data incorporation based on Open Services Provider for distributed enterprises. *CEUR Workshop Proc.* **2017**, *1903*, 42–47. [[CrossRef](#)]
24. Ivaschenko, A.V.; Stolbova, A.A.; Krupin, D.N.; Krivosheev, A.V.; Sitnikov, P.V.; Kravets, O.J. Semantic analysis implementation in engineering enterprise content management systems. *IOP Conf. Ser. Mater. Sci. Eng.* **2020**, *862*, 042016. [[CrossRef](#)]
25. Ivaschenko, A.; Stolbova, A.; Golovnin, O. Data Market Implementation to Match Retail Customer Buying Versus Social Media Activity. *Adv. Intell. Syst. Comput.* **2020**, *1228*, 363–372. [[CrossRef](#)]
26. Tesseract. Available online: <https://github.com/tesseract-ocr/tesseract> (accessed on 24 May 2021).
27. Natasha. Available online: <https://github.com/natasha/natasha> (accessed on 24 May 2021).
28. Spacy. Available online: <https://spacy.io> (accessed on 24 May 2021).
29. Pullenti. Available online: <https://www.pullenti.ru> (accessed on 24 May 2021).
30. Stanza. Available online: <https://github.com/stanfordnlp/stanza> (accessed on 24 May 2021).
31. DeepPavlov. Available online: <https://deeppavlov.ai> (accessed on 24 May 2021).