Search for Answers in Ontological-Semantic Graph

Anastasia Mochalova
IKIR FEB RAS, Paratunka (Kamchatka region), Russia
PetrSU, Petrozavodsk, Russia
stark345@gmail.com

***Abstract*—In this work we propose an architecture of a question answering system constructing anticipated answers and on fuzzy comparison of ontological-semantic graphs of anticipated answers and the analyzed text. We describe a method of ontological-semantic graphs comparison using an ontological-semantic analyzer based on basic ontological-semantic rules. In the working process of the ontological-semantic analyzer, the text is gradually reduced in accordance with basic ontological-semantic rules. In the work we implemented in software the ontological-semantic analyzer and a question answering system prototype. We have conducted the experiments that allow to conclude about efficiency of using an expert system in the ontological-semantic analyzer and about productivity of the described approaches to implementation of the question answering system and the ontological-semantic analyzer.**

I. INTRODUCTION

Development of a question-answering system is becoming more and more relevant problem in the present days. It is connected with an avalanching increase in information volume that contemporary people have to operate.

A variety of approaches to organizing the architecture of question answering system (*QAS*) exist. For example, in the most known *QAS Watson*, analysis of the asked question is performed as well as its classification, decomposition into simple parts and generation of possible answers by means of search in knowledge sources. These may be unstructured knowledge such as usual web pages, weakly structured knowledge such as Wikipedia articles, and structured knowledge such as RDF storages. The process of hypotheses production is divided into two phases: primary search and generation of answers hypotheses. In the primary search process, reference to various knowledge sources is performed. On the stage of hypotheses generation, transformation of the primary search results into answer format is performed. The algorithm of this transformation is specific for the knowledge source. For example, for the results found by the index "oriented on document names", name of the found document is returned as an answer. When the relevant results are found in the RDF-triples storage, transformation into expression in natural language is performed, etc. [1].

The aim of this work is development of the architecture and software implementation of a natural-language *QAS* prototype. User inputs into the system the following data:

As output data, the system must provide the user with:

- a question in natural language;
- an analyzed text in which the system must look for an answer to user's question

As output data, the system must provide the user with:.

1) an answer in natural language;
2) information about basing on which data the answer is given.

For the moment, one of the most effective methods of implementation of a *QAS* is considered to be a method based on comparing semantic graphs of the question and of sentences of the analyzed text. Usage of such method for development and implementation of such systems is described in a series of research works (for instance, [2-6] etc.) and is admitted by their authors to be effective.

In this paper we propose a *QAS* architecture utilizes comparison of extended semantic graphs that use data from the ontology. For representation of such graphs we use the notion "ontological-semantic graph" (or, shortly, "onto-semantic").

Another feature of the proposed approach to organization of *QAS* architecture is that instead of traditional comparison of semantic graphs, comparison of ontological-semantic graphs is considered. In the working process the program compares such graphs constructed from anticipated answers with graphs constructed from the analyzed text. With that, the ontological-semantic graph of the analyzed text is constructed by all ontological-semantic graphs that were constructed from separate sentences of the given text considering logical connections between indivisible sense entities of the text.

II. ONTOLOGICAL-SEMANTIC GRAPHS

A semantic dependency is a certain universal relation that a native speaker beholds in the language. This relation is binary, that is, it holds from one semantic node to another [7]. It is convenient to regard indivisible sense entities of the language as semantic nodes. They can be represented, for example, by the named entitites. We say that two different semantic nodes α and β from the same sentence are related by a semantic dependency named R (denote $R(\alpha,\beta)$) if there is a certain universal binary relation between α and β.

For concrete semantic nodes α and β and the dependency R, the direction is selected in such a way that the formula $R(\alpha,\beta)$ would be equivalent to the statement that " β is R for α ".

By an *ontological-semantic graph* we shall call an oriented graph, the vertices of which are indivisible sense entities of the analyzed text with corresponding information from the

ISBN 978-952-68397-0-7 (paperback), ISBN 978-952-68397-1-4 (PDF)

ontology, and named edges define the name of semantic dependencies connecting these indivisible sense entities. The direction of the edges of the ontological-semantic graph defines the arguments sequence of such dependencies. In Fig. 1 there is an example of an ontological-semantic graph constructed from the sentence *"A British bacteriologist Alexander Fleming discovered penicillin in 1928"*. In curly brackets we indicate the ontological information corresponding to each vertex of the graph.

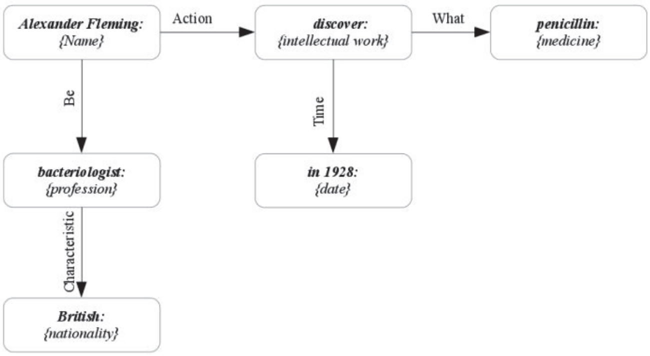

Fig. 1. Graphical representation of an ontological-semantic graph

For each interrogative sentence in a *QAS* we propose to construct a set of anticipated answers with unique semantic structure, comprising various most probable answer formulations to a user-defined question. In case the user's question contains an interrogative word (*where, who, why, when* etc.), these interrogative words are replaced with so-called "indefinite component", about which only limited information is known (for example, some morphological characteristics or some data from the ontology). In Table I we present several anticipated answers constructed for an interrogative sentence *"Who discovered penicillin?"* with examples of corresponding sentences from the analyzed text. In the examples, indefinite components are denoted by characters "*X*", and in curly brackets their morphological characteristics and corresponding data from the ontology are given. G(α) means any hyponym or hyperonym for α kept in the ontology, S(α) – any synonym for α, its alternative name or short definition.

We shall call the vertices of an ontological-semantic graph that correspond to *indefinite components* of the anticipated answer as *indefinite vertices*.

In Fig. 2 we present an example of an ontological-semantic graph of one of anticipated answers "Penicillin was discovered by *X*:{Name || G(person)}", constructed for the question "Who discovered penicillin?" with indefinite vertex *X*, about which we know that it is a family group (Name) or a hyponym/hyperonym of the word "person". Indefinite vertex *X* is encircled with dashed line in the Fig. 2. For transforming an anticipated answer into an ontological-semantic graph we use a set of specially developed rules that are more detailly described in work [8].

An alternative approach to forming the anticipated answers is described in work [9]: the authors propose an approach of constructing *SPARQL* requests after the question asked by the user in natural language. At this, the following steps are performed: segmentation of questions into phrases; mapping of phrases to semantic entities, classes, and relations; and construction of *SPARQL* triple patterns.

TABLE I. ANTICIPATED ANSWERS AND EXAMPLES OF THE CORRESPONDING ANSWERS FROM THE ANALYZED TEXT

Anticipated answer	Example of the answer
X was S(discover) S(penicillin) *X*:{Name || G(person)}	In 1928, penicillin was discovered by Alexander Fleming.
X S(discover) S(penicillin) *X*:{Name || G(person)}	A British bacteriologist discovered the first antibiotic in 1928.
X, who S(discover) S(penicillin), *X*:{Name || G(person)}	A. Fleming, who discovered penicillin, was awarded Nobel Prize.

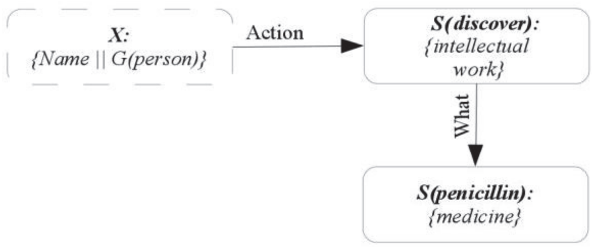

Fig. 2. Graphical representation of an ontological-semantic graph of the anticipated answer

III. QAS ARCHITECTURE UTILIZES ONTOLOGICAL-SEMANTIC GRAPHS

The proposed architecture of a *QAS*, utilizes ontological-semantic graphs, is presented in Fig. 3.

As the input into the system, the user provides an interrogative sentence Q in natural language and the analyzed text T, which is anticipated to contain an answer to the question. Q and T are passed on the input of the module of initial text processing, where they take the initial processing: text formatting symbols that do not bear any semantic role are deleted, orthographical and syntactical errors are corrected, the text is tokenized (that includes breaking the text into paragraphs, sentences and words; for each selected word its morphological characteristics are defined with use of corresponding morphological dictionaries). The next stage is segregation of indivisible sense entities, that may be separate words of word groups united by some common meaning.

Having taken the initial processing T and Q, together with all data received on this stage (depicted as $A(T)$ and A(Q) on the diagram), are passed: $A(T)$ — to the module of ontological-semantic graphs construction, $A(Q)$— to the module of anticipated answers construction. After constructing anticipated answers $a_1, a_2, a_3, \ldots, a_N$, in accordance with the rules using functions for work with ontology (such as *G()*, *S()* etc.), all s, as well as $A(T)$ are passed to the module of ontological-semantic graphs construction.

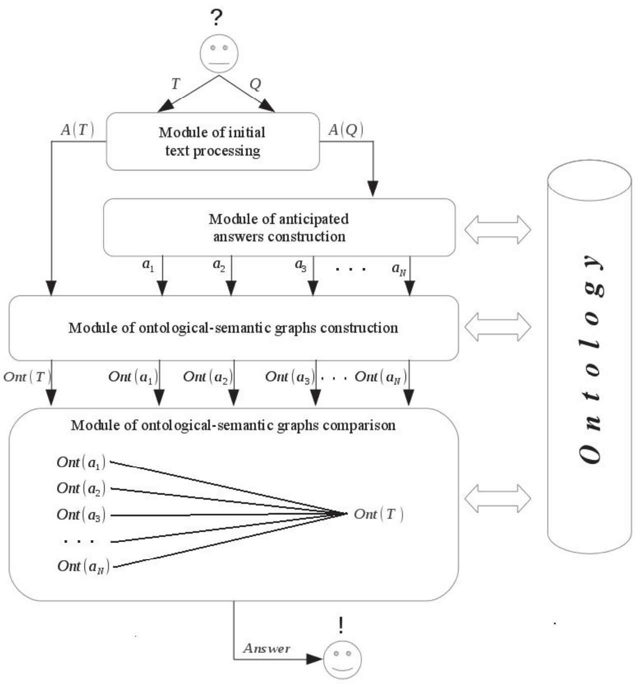

Fig. 3. Architecture of a QAS, utilizes ontological-semantic graphs

The work result of the module of ontological-semantic graphs construction will be N ontological-semantic graphs of anticipated answers: $Ont(a_i),\ \mathrm{i}=1,\dots,\mathrm{N}$ and ontological-semantic graph of the analyzed text: $Ont(T)$. All these graphs are passed to the module of ontological-semantic graphs comparison, where, by the special algorithm detailly described in the following section, graphs $Ont(a_i)$ are pairwise compared with graph $Ont(T)$, and for each such pair the similarity coefficient is calculated.

If for some graph $Ont(T^*)$ being a subgraph of $Ont(T)$: $Ont(T^*)\subseteq Ont(T)$ the similarity coefficient with one of ontological-semantic graphs of anticipated answer $Ont(a_i)$ is higher than the value defined in the program, then $Ont(T^*)$ is considered to be *similar* to $Ont(a_i)$. As a short answer, the user receives the vertex of the graph $Ont(T^*)$ similar to $Ont(a_i)$ that corresponds to the indefinite vertex $Ont(a_i)$ (in case the indefinite vertex exists which implies existence of an interrogative word in the question Q). If an indefinite vertex is not present in $Ont(a_i)$, the user receives answer «yes» as the short answer. As an extended answer, the user is provided with one or several sentences basing on which such graph $Ont(T^*)$ had been constructed. User can view the text that precedes and succeeds the sentences proposed by the system as an answer.

IV. FUZZY COMPARISON OF ONTOLOGICAL-SEMANTIC GRAPHS

The architecture of a *QAS* described in the previous section is utilizes ontological-semantic graphs. The result of such graphs comparison is the *similarity coefficient* of ontological-semantic graphs (let us call it η), taking its values in the interval $[0,1]$. Let us consider comparison of ontological-semantic graphs from the point of view of isomorphism of two graphs and their subgraphs.

Modifying for ontological-semantic graphs the notion of isomorphism given in [10] for classical graphs, we give the following definition: ontological-semantic graphs G_1 and G_2 are called isomorphic ($G_1 \simeq G_2$), if there exists such one-to-one correspondence between their vertices and edges that corresponding edges connect corresponding vertices; in addition, ontological information about graph G_1 vertices does not contradict ontological information about corresponding vertices of G_2.

For convenience we introduce the function $Des(G_1,G_2)$ that defines similarity coefficient of the graphs G_1 and G_2: $Des(G_1,G_2)=\eta$.

When comparing ontological-semantic graph $Ont(a_i)$ of the anticipated answer a_i with ontological-semantic graph $Ont(T^*):Ont(T^*)\subseteq Ont(T)$, there may arise the following cases (from the point of view of isomorphism of these graphs):

$$Des\big(Ont(a_i),Ont(T^*)\big)=\begin{cases} 0 & \text{if } \begin{cases} Ont(T^*)\not\simeq Ont(a_i) \\ \quad AND \\ \nexists Ont(T^{**}),Ont(a_i^*):Ont(T^{**})\simeq Ont(a_i^*) \end{cases} \\ 1, & \text{if } \quad Ont(a_i)\simeq Ont(T^*) \\ \xi\in(0,1), & \text{if } \begin{cases} Ont(T^*)\not\simeq Ont(a_i) \\ \quad AND \\ \exists Ont(T^{**}),Ont(a_i^*):Ont(T^{**})\simeq Ont(a_i^*) \end{cases} \end{cases}$$

where $Ont(T^{**})\subseteq Ont(T),\ \ Ont(a_i^*)\subseteq Ont(a_i)$.

In Fig. 4 we give an example of an ontological-semantic graph constructed from one of the sentences of the analyzed text “A. Fleming discovered the first antibiotic.” This graph is isomorphic to the graph presented in Fig. 2.

During our work, we implemented in *Java* programming language the algorithm of ontological-semantic graphs comparison.

Similarity coefficient η of two graphs $Ont(a_i)$ and $Ont(T^*)$ ($\eta=Des(Ont(q_i),Ont(T^*))$) is calculated in such way that η takes value allowing to consider the graphs to be similar (in case η exceeded the experimentally defined value) only upon obligatory fulfillment of the following two conditions:

1) $Ont(T^*)$: $Ont(T^*) \subseteq Ont(T)$ contains ontological-semantic relation «Action», connecting the same arguments that have equal initial form (or, for indefinite vertices — does not contradict) with the arguments of the relation «Action» of the graph $Ont(a_i)$;

2) in case of presence of an interrogative word in the question, there is a semantic dependency in $Ont(T^*)$, defined by the interrogative sentence as main (for a question with interrogative word "where" — relation "Location", for a question with interrogative word "when" — "Time" etc.) and the arguments of these dependencies are either equal or (for an indefinite vertex) not contradicting.

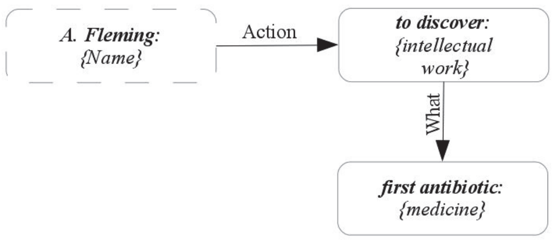

Fig. 4. Example of an ontological-semantic graph, isomorphic to the graph presented in Fig. 2

In case of finding in $Ont(T^*)$ and $Ont(a_i)$ other equal or non-contradicting relations, the value of the coefficient η is increased; moreover, the higher is the sum of all edges of connected graph from $Ont(a_i)$ isomorphic to a connected graph from $Ont(T^*)$ is, the higher the sum at which the increase comes is.

V. ONTOLOGICAL-SEMANTIC ANALYZER

In the course of work we developed an ontological-semantic analyzer based on basic ontological-semantic rules.

A. Basic ontological-semantic rules

We shall call the rule, according to which the expert system (*ES*) finds in the analyzed text (where indivisible sense entities have been segregated and each of them has been referred to a certain ontology class or object) semantic dependencies between classes and the objects they consist of, the basic ontological-semantic rule. The basic ontological-semantic rule, that is a rule of *ES*, consists of the left and the right side. The left side describes conditions upon which the actions described at the right side are performed. For example, in the left side of the rule a biconnected facts list is always described, as well as Boolean functions taking facts from this biconnected list as their arguments. A fact of *ES* consists of: ontology class (or object), link to the previous fact (from the left) and to the next fact (from the right), morphological characteristics and coordinates in the text (sentence number and position of the class (or the object) in the sentence).

The right side of the rule contains the list of actions, each of which can: modify any fact of *ES* (by means of modifying a relation in the corresponding common ontology Class or Object); add to the queue for removal a fact of *ES* that has a certain removal priority; other actions.

In the present work we have developed a program in Java language to transform a BOSR of the form *(MORPH→A:-:nom,sinMORPH→N:-:nom,sin RELATION→ATTRIBUTE-1-0- ADD_TO_THE_QUEUE_FOR_REMOVAL(0,7))* into rules for the ES Drools. The given BOSR means that if in the analyzed text there has been found an adjective (A) in nominative case (nom), singular number (sin), followed by a noun (N) in nominative case, singular number, then form a semantic relation ATTRIBUTE (belongs to the group of semantic relations with priority 7), connecting these two facts. Below we provide a template of the *ES Drolls* generated by the program implemented by the author using the described *BOSR*. The text after the symbols "//" or between "/*" and "*/" is a comment.

rule "338" *// name of the rule (number 338)*

salience 100 */* priority of the rule (is not related to the queue with priority. The point is that the higher is the rule's priority, the more the rule is likely to be selected upon condition of the trueness of left-hand sides of several rules) */*

when *// defines the beginning of the condition WHEN*

$w0 : Fact(partOfSpeech == "A", hsAttrs contains "nom", hsAttrs contains "sin") */* $w0 – address of the fact. Fact -> fact with attributes */*

$w1 : Fact(prev == $w0, partOfSpeech == "N", hsAttrs contains "nom", hsAttrs contains "sin") */* $w1 - address of the fact. Fact -> fact with attributes. prev -> previous fact */*

then *// defines the beginning of the condition THEN*

SemanticRelation sem = new SemanticRelation("ATTRIBUTE"); */* create an object sem with the type ATTRIBUTE */*

sem.setLeftAutoPosInText($w1);

// set the left-hand argument to be $w1

sem.setRightAutoPosInText($w0);

// set the right-hand argument to be $w0

boolean changed = myQueue.addOrUpdateCheckToDelete($w0, 7); */* add to the queue for removal the fact $w0 with priority 7. If the new priority for removal is less or equal to the old one (which is stored in the queue for removal myQueue), then changed = false. Otherwise changed = true;*/*

if(changed)

update(myQueue); */* update the queue for removal myQueue in the ES Drools */*

String indexSem = sem.getIndexString();

if(hsAllIndexedSemanticRelations.contains(indexSem) == false)

```
/* if the semantic relation has not been found before, then add the new fact → semantic relation */
{
        hsAllIndexedSemanticRelations.add(indexSem);
        insert(sem);
}
end // end of the rule
```

After all basical ontological-semantic patterns (with true left sides) have been found on the current facts of the expert system, one fact with the highest priority is removed from the queue for removal and from the working memory of the expert system. When removing a fact from the working memory of the expert system, one should update the left and the right facts for the fact being removed.

B. Ontological-semantic analyzer, integrated with the ontology

At the input the ontological-semantic analyzer (*OSA*) receives a verifiable text T, that goes into the module of initial text processing where it undertakes a preliminary processing: text formatting symbols that do not bear any semantic role are deleted, orthographical and syntactical errors are corrected, extra spaces and line breaks are deleted etc. Then the text is tokenized, that includes breaking the text into paragraphs, sentences and words. For each selected word its morphological characteristics are defined with use of corresponding morphological dictionaries. The next stage is segregation of indivisible sense entities, that may be separate words or word groups united by some common meaning. Examples of named entities consisting of several words may be certain named entities or composite parts of speech (adverb "good and proper", linking word "because", numeral adjective "forty five" etc.). The final stage of initial processing is search for logical connections in the text.

The text T that has undertaken initially processing, together with all data received at this stage (marked as *A(T)* on the diagram), are passed onto input of the module of collation with basic ontological-semantic rules (*BOSR*).

The work of the module of collation with *BOSR* starts from collation of indivisible sense entities segregated from the verifiable text T with classes and objects of the ontology. At this stage the problem of resolving lexical polysemy is solved in order to define, which of the set of existing classes and/or objects with equal names does the considered sense entity belong to. If some sense entity from *A(T)* corresponds to no object or class from the ontology, it will be considered without association with the ontology in the next semantic analysis.

Further, with use of the *BOSR* that the *ES* is based on, search of ontological-semantic dependencies and/or modification of the particular ontology *Ont(T)* are performed. *ES*, being a component of the *OSA*, consists of the following main parts:

1) Knowledge base — the assembly of all *BOSR*;

2) Working memory — facts of the *ES*, that is, indivisible sense entities from *A(T)*, for which the following characteristics are defined:

- corresponding classes or objects from the "general ontology";
- morphological characteristics (data from the module of initial text processing);
- coordinates in the text (sentence number of *A(T)* and position in this sentence).

Each fact in the *ES* keeps information about the previous (and the next) facts from *A(T)*, if they exist (that is, if the fact does not correspond the first (or the last) indivisible sense entity) in *A(T)*.

3) Block of logical output — a program that forms the working rules list, selects from it a *BOSR* with the highest priority and performs it. Performed rule is deleted from the working rules list.

4) Component of knowledge acquisition — software component that automates updating of the *KB* of the expert system with *BOSR*.

5) Explanatory component — software component that displays the course of solution by user's request (which *BOSR* from *KB* worked on which facts).

The result of work of the ontological-semantic analyzer is the ontological-semantic graph *Ont(T)*, that is, a semantic graph constructed from the analyzed text T, having indivisible sense entities segregated from T as its vertices, and direction of the connecting edges as the sequence order of semantic dependencies arguments found in the text. Each edge of the semantic graph *Ont(T)* is named in accordance with the name of the semantic dependency that produced it.

In Table II we provide examples of *BOSRs* written in simplified form, examples of corresponding texts and semantic relations discovered in these texts.

During our work, we implemented in software the ontological-semantic analyzer in *Java* programming language, basing on basic ontological-semantic templates with deletion.

The Table III shows an example of using the ontological-semantic analyzer and how the priority queue (*Q*) is gradually changing. The analyzed text (*AT*) is "Yesterday, the yachting sport school honors left for a camp".

VI. CONCLUSION

Software implementation of the *QAS* based on construction of the set of anticipated answers and comparison of ontological-semantic graphs proves the efficiency of such approach to organizing the *QAS* architecture.

The ontological-semantic analyzer that was developed and implemented in the course of this work and based on basic ontological-semantic rules with deletion showed the productivity of the proposed method of its implementation. This ontological-semantic analyzer has been used as a component module in the *QAS* for constructing ontological-semantic graphs.

TABLE II. SEARCH OF SEMANTIC RELATIONS IN THE TEXT USING BOSR

BOSR	Text	Semantic relations
ONT->{Hyponym(person) || NAME} ONT->{Synonym(write) || Synonym(create)} ONT->{Synonym(literary work) || NAME} RELATION->Author (2, 0)	*Lermontov wrote «Borodino»*	Author(«Borodino», Lermontov)
	Workers composed a petition	Author(petition, workers)
ONT->{DATE} ONT->{Synonym(start)} ONT->{Synonym(event)} RELATION-> Event_start(2, 0)	*Lunch starts at 12:00*	Event_start(lunch, at 12:00)
	The swimming competitions started on Thursday	Event_start(competitions, since Thursday)
	On 21 October 1947, the Indo-Pakistan war started	Event_start(war, 21 October 1947)

TABLE III. AN EXAMPLE OF USING THE ONTOLOGICAL-SEMANTIC ANALYZER

Step	Found semantic relationships	Operations on Q	Elements of Q
1	[0] [1] [2] [3] [4] [5] [6] [7] [Yesterday], [the yachting] [sport] [school] [honors] [left] [for] [a camp]		
	Belong(honors, school)	insert(Q,(school, 2, [3]))	Q = {(school, 2, [3])}
	Belong(school, sport)	insert(Q,(sport, 2, [2]))	Q = {(school, 2, [3]), (sport, 2, [2])}
	Property(sport, the yachting)	insert(Q,(the yachting, 1, [1]))	Q = {(school, 2, [3]), (sport, 2, [2]), (the yachting, 1,[1])}
	Location(left, for a camp)	insert(Q,(for a camp, 3, [7]))	Q = {(school, 2, [3]), (sport, 2, [2]), (the yachting, 1, [1]), (for a camp, 3, [7])}
	Action(left, honors)	insert(Q,(honors, 15, [4]))	Q = {(school, 2, [3]), (sport, 2, [2]), (the yachting, 1, [1]), (for a camp, 3, [7]), (honors, 15, [4])}
		remove(Q, (the yachting,1,1))	Q = {(school, 2, [3]), (sport, 2, [2]), (for a camp, 3, [7]), (honors, 15, [4])}
2	[0] [2] [3] [4] [5] [6] [7] [Yesterday], [sport] [school] [honors] [left] [for] [a camp]		
	(new semantic relationship is not found) AND (isEmpty(Q) = false) => Continue		
		remove(Q, (school,2,[3]))	Q = {(sport, 2, [2]), (for a camp, 3, [7]), (honors, 15, [4])}
3	[0] [2] [4] [5] [6] [7] [Yesterday], [sport] [honors] [left] [for] [a camp]		
	(new semantic relationship is not found) AND (isEmpty(Q) = false) => Continue		
		remove(Q, (sport,2,[2]))	Q = {(for a camp, 3, [7]), (honors, 15, [4])}
4	(new semantic relationship is not found) AND (isEmpty(Q) = false) => Continue		
		remove(Q, (for a camp,3,[7]))	Q = {(honors, 15, [4])}
5	(new semantic relationship is not found) AND (isEmpty(Q) = false) => Continue		
		remove{(honors,15,[4])}	Q = {}
6	[0] [5] [Yesterday], [left]		
	Time(left, yesterday)	insert(Q,(yesterday,7,[0]))	Q = { Q,(yesterday,7,[0])}
		remove(Q,(yesterday,7,[0]))	Q = {}
7	[5] [left]		
	(new semantic relationship is not found) AND (isEmpty(Q) = true) => End		

During our work, we implemented in Java the *QAS* for Russian language with the architecture described in section 3. Ontological-semantic graphs that are used in system's work are constructed with help of the software implementation of the semantic analyzer based on basic ontological-semantic rules (see section 5). The program of the described ontological-semantic analyzer is registered in *Rospatent*.

Also, in the process of software implementation of this *QAS* we developed various component modules of the *QAS*, such as the modules of morphology, tokenization, segregation of named entities etc., but description of their operation algorithms is outside the scope of this paper. Also, during our work we developed an object-oriented ontology model, the data from which are used when constructing ontological-semantic graphs. Initial filling of the ontology with data, including information from hierarchical dictionaries, was performed. This allowed to define classes inheritance and belonging of objects to certain classes. We plan to extend the used ontology using such structured information storages as *DBpedia* [11], *Freebase* [12], *Wikidata* [13], *Wikipedia* [14], *Wiktionary* [15] and *Yago* [16].

Results of the *QAS* work were compared with *NLUlite* system [17]. NLulite is based on the use of *CYK* stochastic parser, *CCGbank* [18], *Discourse Representation Theory* [19], Ontology given by *Wordnet* [20] and others. In Table IV we present examples of texts and questions to them, which were correctly answered with use of software implementation of the *QAS* described in this work, while *NLUlite* did not give any answers.

TABLE IV. ANALYZED TEXT AND QUESTIONS.

№	Text	Question
1	The red and rubber ball lay in the field.	What is the ball color?
2	Peter gave the ball to Jack.	Who has the ball?
3	Elephants can live up to 70 years in the wild.	How long can live elephants?
4	Peter has three apples. Peter ate two apples.	How many apples remained?
5	On the table lies an apple.	What is on a table?
6	Pushkin wrote the novel.	Who is the author of the novel?

The results show the importance of using ontologies in the *QAS* and prove the working efficiency of the *QAS* the architecture of which is utilizes comparing ontological-semantic graphs, and also the working productivity of the implemented ontological-semantic analyzer.

Also, we can conclude about possibility to construct a hybride *QAS* based on combination of the algorithms proposed in this work with the algorithms of *NLUlite* system.

In the work we have shown by experiments that the implementation of the proposed working method of the ontological-semantic analyzer with use of the expert system *Drools* [21], using the algorithm of quick comparison with templates *PHREAK* [22], results in time profit in average 9-11 times in comparison with a software implementation that does not use expert systems. Testing of the *QAS* and the ontological-semantic analyzer were performed with Intel Core i7-4702MQ CPU 2.20GHz processor, SSD disk and in the operating system Ubuntu 14.04.

ACKNOWLEDGMENT

The work was implemented with financial support from the Russian Foundation for the Humanities as part of research project №15-04-12029 — Software development of an electronic resource with an online version of a Russian-language question answering system.

REFERENCES

[1] D.A. Ferrucci, E.W. Brown, J. Chu-Carroll et al, “Building Watson: An Overview of the DeepQA Project”, *AI Magazine*, vol. 31, No 3, 2010, pp. 59–79.

[2] D. Han, Y. Kato, K. Takehara, T. Yamamoto, K. Sugimura, M. Harada, “QA system metis based on web searching and semantic graph matching”, *IFIP International Federation for Information Processing*, vol. 228, Boston: Springer, pp. 123-133.

[3] L. Dali, D. Rusu, B. Fortuna, D. Mladenić, M. Grobelnik, “Question answering based on semantic graphs”. Web: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.204.5721.

[4] Srini Narayanan, Sanda Harabagiu, “Question Answering Based on semantic structures”, *Proceeding COLING '04 Proceedings of the 20th international conference on Computational Linguistics Article No. 693.*

[5] D. Shen, M. Lapata, “Using Semantic Roles to Improve Question Answering”, *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Prague, June 2007, pp. 12–21.

[6] A.A. Solovyev, “Who is to blame and Where the dog is burried? Method of answeres validations based on fuzzy matching of semantic graphs in Question answering system” , *Russian Information Retrieval Evaluation Seminar. ROMIP Proceedings*, 2010, Kazan.

[7] A.V. Sokirko, “Semantic dictionaries in automated text processing: as exemplified by the DIALING system”, *PhD. tech. sci. diss*, Moscow, 2001.

[8] A.V. Mochalova, “Some issues of work of a Russian-language question answering system using data from the ontology”, *Proceedings of the Sixth international conference “System analysis and information technologies”*, Svetlogorsk, 2015.

[9] M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, G. Weikum, “Natural language questions for the web of data”, *EMNLP-CoNLL '12 Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 379-390.

[10] S.V. Yablonsky, Introduction to discrete mathematics, Moscow: Higher school, 2003, 384 p.

[11] DBpedia official website, Web: http://wiki.dbpedia.org.

[12] Freebase official website, Web: http://www.freebase.com.

[13] Wikidata official website, Web: https://www.wikidata.org.

[14] Wikipedia official website, Web: https://ru.wikipedia.org.

[15] Wiktionary official website, Web: https://ru.wiktionary.org.

[16] Yago official website, Web: www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/.

[17] NLUlite , Web: https://nlulite.com/.

[18] J. Hockenmaier, M. Steedman. CCG bank: User's Manual. Web: http://repository.upenn.edu/cgi/viewcontent.cgi?article=1054&context=cis_reports.

[19] H. Kamp, U. Reyle, *From discourse to logic: introduction to modeltheoretic semantics of natural language, formal logic and discourse representation theory. Part 1*, Springer Science & Business Media, 1993, 713 p.

[20] WordNet official website, Web: https://wordnet.princeton.edu/.

[21] Drools official website. Web: http://www.drools.org/.

[22] Drools Documentation official website. Web: http://www.drools.org/learn/documentation.html.