Ontological-semantic text analysis and the question answering system using data from ontology

Kuznetsov V. A.*, Mochalov V. A.**, Mochalova A. V.**

** Petrozavodsk State University, Lenina str. 33, 185910 Petrozavodsk, Russia*

*** Institute of Cosmophysical Research and Radio Wave Propagation FEB RAS, Mirnaia str. 7, 684034 Paratunka, Kamchatka region, Russia.*

kuznetcv@mail.ru, sensorlife@mail.ru, stark345@gmail.com

***Abstract*—Today, with an avalanching increase in information, the task of developing the systems that allow user to quickly search desired information in large text volumes is becoming more and more urgent. An example of such system is the question answering one. In the work we describe an architecture of such system which work is based on utilizing data from an ontology. We propose an algorithm for automatic update of the ontology basing on use of an expert system and ontological rules for logical inference. We also describe an ontology with the structure based on object-oriented model and describe the functions that are used to update the ontology and extract data from it. We describe the way to update the ontology and to modify the stored data using the rules stored in the ontology. For writing ontological-semantic rules we use the Drools expert system that utilizes the PHREK algorithm for fast pattern matching.**

We analyze the issues of using Apache Spark system for distributed implementation of the algorithm.

***Keyword*—Question Answering System, ontology, expert systems, semantic analysis.**

I. INTRODUCTION

DEVELOPMENT of question answering systems is nowadays becoming more and more urgent problem. This is connected with an avalanche growth of information volume that modern people need to operate.

Basing on analysis of operation algorithms of many Question Answering Systems (QASs), including Lasso [1], QA-LaSIE [2], TEQUESTA [3] etc., one can conclude that all of them do, in the whole, comply a certain general architecture. The high-level representation of the latter is given at Fig. 1. The system receives a question on the natural language as an input. After that, the text of the question is automatically processed. The main stages of this process are preliminary text processing, tokenization, morphological, syntactic and semantic analysis, extraction of named entities and definition of the logical links between the parts of the sentences. Some of these stages may be dropped out or simplified in different QASs implementations and descriptions. Some are, on the contrary, basical for the system operation in a whole: an example is semantic analysis in the work of M. V. Mozgovoy [4]. Several more procedures that can be executed on the stage of automated question text processing are definition of the question type, definition of the expected answer type etc. Basing on results of the automated question text processing, a query is formed to be passed on to a search engine. The search engine, in its turn, selects a predefined number (N) of documents most relevant to a query from the collection. The texts of each of selected documents as well as the question text are automatically processed. Here, the machine algorithms of the question text processing may differ from the algorithms of the selected documents processing. Further, by means of internal algorithms of the QAS, the specific text fragments are selected from the documents returned by the search engine. The selected text fragments are presented by the system as an answer. The most advanced QASs can use data from factbases (FB), databases (DB) and ontologies on the stage of text fragments selection. The information from such data storages can complement the answer/answers of the system.

In the work we propose an operation algorithm of an ontological-semantic analyzer (a semantic analyzer that uses ontology) of text. We describe how to use results of its work in a developed question answering system. During operation of the ontological-semantic analyzer, the ontology is being changed and its data is used to define semantic links between parts of the sentences. As a result of these changes, the data in the ontology may be learned, deleted or updated. In such a way the ontological-semantic analysis solves one more task apart from being used in a question answering system: it performs automated ontology learning.

Further we give a brief overview of the methods of ontologies learning.

In work [5] the authors define three main methods of

Manuscript received July 9, 2015. The work was implemented with financial support from the Russian Foundation for the Humanities as part of research project No.15-04-12029 “Software development of an electronic resource with an online version of a Russian-language question answering system”.

Kuznetsov V. A. is with the Petrozavodsk State University, Russia (e-mail: kuznetcv@mail.ru) .

Mochalov V. A., is with the Institute of Cosmophysical Research and Radio Wave Propagation FEB RAS, Russia (e-mail: sensorlife@mail.ru).

Mochalova A. V. is with the Institute of Cosmophysical Research and Radio Wave Propagation FEB RAS, Russia (corresponding author to provide phone: 8-916-820-15-25; fax: 8-(41531)-33718; e-mail: stark345@gmail.com).

ontologies learning:

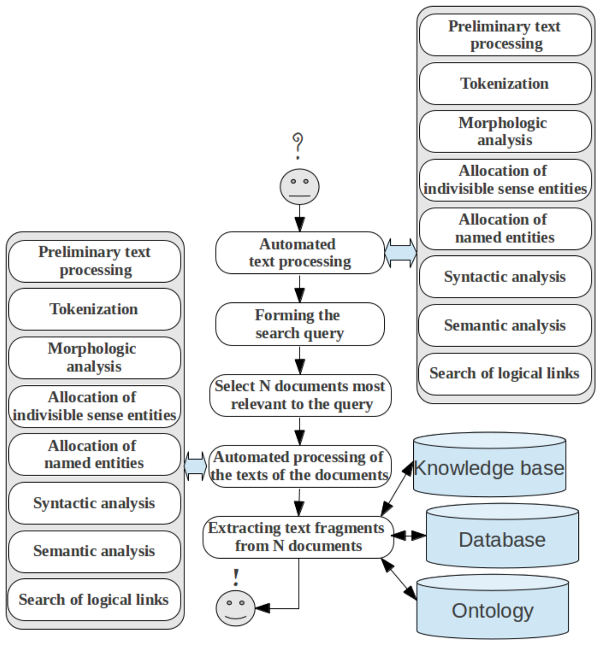
Preliminary text processing
Tokenization
Morphologic analysis
Allocation of indivisible sense entities
Allocation of named entities
Syntactic analysis
Semantic analysis
Search of logical links
?
Automated text processing
Forming the search query
Select N documents most relevant to the query
Automated processing of the texts of the documents
Extracting text fragments from N documents
!
Knowledge base
Database
Ontology

Fig. 1. Workflow architecture in a question answering system

1) «manual» input;
2) automatic or automated input using traditional lexicographical information (encyclopedic, word-defining and other dictionaries and databases);
3) automatic or automated input based on analysis of distributive characteristics of the lexis in text corpus.

Ontology learning by means of manual input is a very labor consuming procedure that requires participation of highly qualified specialists. For this reason, development of automatic (or at least partly automated) methods of ontology learning is today a very urgent task.

One of the most frequent methods of automatic ontology learning is constructed on analysis of dictionary definitions. In such methods, the ontological constructions (entities and the type of their relation) discovered by use of pattern search in dictionary definitions are added to the ontology. For instance, if one needs to collect information about all possible means of transport, the dictionary search is performed using all patterns of the kind "X is a vehicle that...", "X is a means of transport used for...", "X is a vehicle type equipped for..." etc. Such search patterns may be developed manually or automatically using self-learning programs [6]. The idea of such a way of automated ontology learning is described in works [7], [8], [9], [10].

The authors of the work [11] compare the methods of automated ontologies learning, consider their own experience and confirm that at the present day the most promising technology from the perspective of obtaining practical results is the one that uses traditional lexicographical information (encyclopedic and word-defining dictionaries).

The work [12] describes a method of automatic construction of domain-specific ontology basing on analysis of linguistical characteristics of text corpus.

A more detailed survey of the existing methods of automated ontologies learning is provided in work [13].

II. Ontology Structure and Functions for Working with the Ontology

An **ontology** (in a formulation by Gruber who was one of the first to use this notion in the area of information technology) is a formal specification of conceptualization [14]. By conceptualization, we understand a description of a set of notions (concepts) of the subject domain and links (relations) between them. As understood today, ontology in informatics is an hierarchical data structure including all relevant object classes, their connections and rules (restrictions) defined in this domain and necessary for solving the problems assigned for an information system.

In this work we propose to use an object-oriented ontology model consisting of **classes** (certain abstract images sharing in the common sense a certain set of properties) and **objects** (part of the real world having some certain properties of the class it belongs to; the same object may belong to different classes and the same class may have different objects). Classes and objects can be interconnected by various **relations** that define certain dependencies between them. Here, a sequence of arguments, for which the relation is being defined, does matter. Examples of such structured data storages are DBpedia [15], Freebase [16] and Wikidata [17], Wikipedia [18], Wiktionary [19]. For instance, in work [20] the authors propose an approach to the automated construction of a general-purpose lexical ontology based on the Wiktionary data. When adding new information (classes, objects or relations between them) to the ontology, one needs to store the timestamp of the changes being made and the source basing on which the changes are made. It should also be possible to make a recovery of the ontology modifications (for example, when an error has been detected). To make this possible, the whole history of ontology modification should be stored. The initial ontology structure can be formed using ideographic dictionaries.

To be specific, we describe high-level functions for ontology learning with the data and for data extraction from the ontology:

1) **CreateClass (idClass, nameClass)** – adds a class with unique identifier idClass and the name nameClass into the ontology;
2) **CreateObject (idObject, idClass, nameObject)** – adds an object with unique identifier idObject and the name nameObject of the class with unique identifier idClass into the ontology;
3) **CreateRelation (idRelation, relationName)** – creates a relation with unique identifier idRelation and the name relationName;
4) **CreateRelation (id1, id_2, idRelation, h)** – adds a relation with unique identifier idRelation, linking two classes or objects (or an object and a class) with unique identifiers id_1 and id_2 correspondingly, where h is a frequency (or weight) of the relation, into the ontology;
5) **Inheritance (id1, id2)** – adds information that the class with unique identifier id2 is inherited from the class with unique identifier id1 into the ontology;
6) **Inheritance_with_denial (id1, id2, {[idRelation1], [idRelation2], [idRelation3], …})** – adds information that the class with unique identifier id2 is inherited from the class with unique identifier id1 excepting relations with unique identifiers from the array {[idRelation1], [idRelation2], [idRelation3], …} into the ontology;

7) **PartialInheritance (id1, id2, {[idRelation1], [idRelation2], [idRelation3], …})** – adds information that the class with unique identifier id2 is inherited from the class with unique identifier id1, but only the relations with unique identifiers from the array {[idRelation1], [idRelation2], [idRelation3], …} are inherited, into the ontology;
8) **ReturnClassName (idClass)** – returns the name of the class with unique identifier idClass;
9) **ReturnObjectName (idObject)** – returns the name of the object with unique identifier idObject;
10) **What_Is (text, position)** – returns value 1 if the sense entity from the analyzed text text, having its position in this text is explicitly defined by the data stored in position variable, is an ontology class; 2 if this entity is an ontology object; 3 if there has been detected no information about this sense entity in the ontology;
11) **ReturnAllRelations (idClassOrIdObject)** — for a class or an object with unique identifier idClassOrIdObject, returns a set consisting of the following values groups:
    a) idRelation – unique identifier of the relation linking a certain class or an object to the class with unique identifier idClass;
    b) idClassRel – unique identifier of a class or an object to which the class with unique identifier idClass is linked by the relation with unique identifier idRelation;
    c) relationName – relation name;
    d) val – relation value;
    e) h — relation frequency (or weight);
12) **ReturnRelations (idClassOrObject, RelationName)** – is similar to the function ReturnAllRelations (idClassOrIdObject) in which the names of relations are restricted to RelationName only;
13) **ReturnAllObjects (idClass)** — returns a set of unique identifiers of objects from the class with unique identifier idClass;
14) **ReturnParentClasses (idClassOrObject)** — returns unique identifiers of the classes that are parent to a class or an object with unique identifier idClassOrObject;
15) **ReturnRelationValue (idClassOrObject1, idClassOrObject2, relationId)** – returns the value of the relation with unique identifier relationId, linking two objects or ontology classes with unique identifiers idClassOrObject1 and idClassOrObject2;
16) **ReturnAllAncestorClasses (idClassOrObject)** – returns unique identifiers of all classes that are ancestors to a class or an object with unique identifier idClassOrObject;
17) **ReturnAllSuccessorClasses (idClass)** – returns unique identifiers of all classes that are successors of any level for a class with unique identifier idClass;
18) **ReturnAllSuccessorObjects (idClass)** – returns unique identifiers of all objects of the classes that are successors of any level for the class with unique identifier idClass and for this class itself.
19) **insertOrUpdateRelation(RelationName, idClassOrObject1, idClassOrObject 2, h)** – if the ontology does already contain the relation named RelationName, connecting the class or the ontology object with unique identifier idClassOrObject1 to the class or the ontology object with unique identifier idClassOrObject2, then increase the weight of this relation by h. If the ontology does not contain such relation, then add it and assign the weight h.

For writing ontological-semantic rules we propose to use the following notations:
1) **Unit** – undefined token, class, object or relation from the ontology;
2) **UnitCO** – class or object from the ontology;
3) **Class** – class from the ontology;
4) **Object** – object from the ontology:
5) **Rel** – relation linking two classes, two objects or a class and an object from the ontology;
6) **UDT** – undefined token, that is a token (indivisible sense entity) the information about which is not present in the ontology yet. That means that this token is neither an ontology class nor an object.

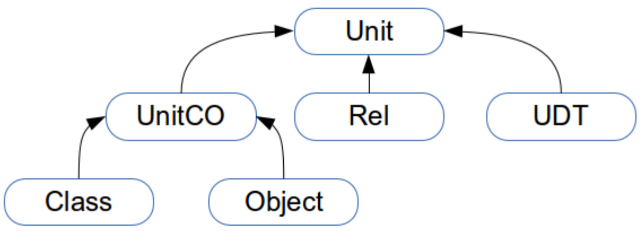

Fig. 2. Hierarchy of the notations used for describing ontological-semantic rules.

Hierarchical relationship between these notations is given at Fig. 2.

III. Architecture of the Question Answering System Integrated with an Ontology

The question answering system is supposed to operate in two modes: ontology learning mode (when system input are valid texts and the general ontology is being modified) and user answering mode. By “valid” here we understand the text containing the data the verity of which is unquestioned. In the scheme presented at Fig. 3 and depicting an architecture of the question answering system, everything concerning only the user answering mode is marked with a dash line.

Ontology learning mode

When a question answering system operates in ontology learning mode, it receives a valid text T as its input. The text enters the module of initial text processing where it is preliminarily processed: text formatting symbols that do not carry any role during text analysis are removed, orthographical and syntactic mistakes are corrected, extra symbols of whitespaces and line breaks are removed and so on. The tokenization stage follows, including breaking text into paragraphs, sentences and words. For each marked word its morphological properties are being defined with use of corresponding morphological dictionaries. The next stage is separating non-divisible sense entities which can be separate words or groups of words that are united by some common meaning. Some examples of named entities consisting of several words are some named entities (New York, Santa Claus, Mr. Smith etc.) or composite parts of speech (an idiom “good and proper”, a linking word “such as”, a numeral adjective “forty five” etc.). The final stage of the initial processing is search for logical links in the text.

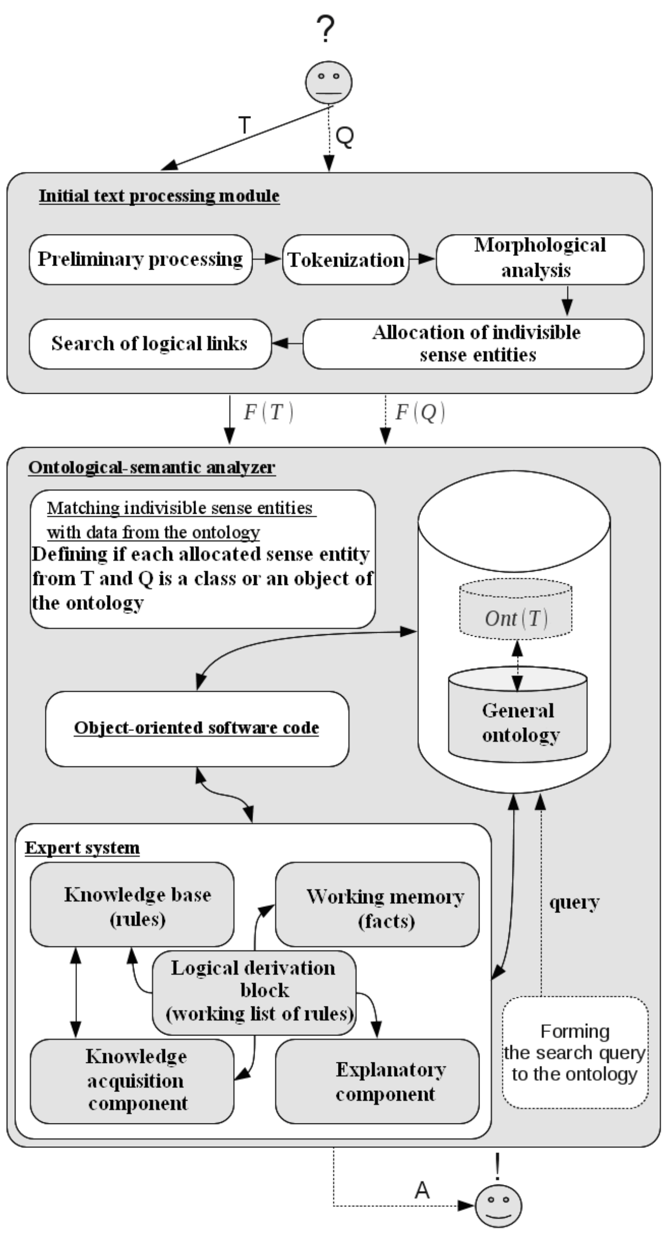
?
T
Q
Initial text processing module
Preliminary processing
Tokenization
Morphological analysis
Search of logical links
Allocation of indivisible sense entities
F(T)
F(Q)
Ontological-semantic analyzer
Matching indivisible sense entities with data from the ontology
Defining if each allocated sense entity from T and Q is a class or an object of the ontology
Ont(T)
General ontology
Object-oriented software code
Expert system
Knowledge base (rules)
Working memory (facts)
Logical derivation block (working list of rules)
Knowledge acquisition component
Explanatory component
query
Forming the search query to the ontology
!
A

Fig. 3. Architecture of the question answering system.

The text T that has been initially processed, together with all data received on this stage, is denoted as F(T) on the scheme and passed onto input of ontological-semantic analyzer.

The work of ontological-semantic analyzer begins with association of non-divisible sense entities separated in the valid text T with classes and objects from the ontology. At this stage, a task of lexical ambiguity resolution is being solved for the purpose of defining which one of the set of existing classes and/or objects with the same names does the considered sense entity belong to.

After the valid text has been initially processed, the ontology modification takes place. For this, the ontology itself, the expert system (consisting of the Knowledge base containing rules, the Working memory containing facts, the Block of logical derivation containing the operational list of the rules and the Component of knowledge acquisition) and the object-oriented code are used. As a result of certain rules stored in the expert system (ES), the ontology may experience addition or removal of certain classes or objects, as well as addition, removal or modification of the relations between them.

User answering mode

During operation of the question answering system, the system receives as its input the question Q asked by a user and the text T selected by the user for searching the answer in. The user provides the question sentence Q in natural language. The text T to search the answer in and the question Q are passed onto input of the module of initial text processing where they get through the same stages of automated text processing that were described for the ontology learning mode. The working results of this module (F(Q) for the question sentence and F(T) for the text in which an answer is searched), on the analogy with the ontology learning mode, are passed onto input of the module of the ontological-semantic analyzer.

In the ontological-semantic analyzer, the association of non-divisible sense entities that have been separated in Q and T with classes and objects of the ontology takes place.

Next, with use of the Ontology and the Expert system, the separate ontology Ont(T) is created by means of the object-oriented code after the text that has been provided by the user and experienced the initial processing. All classes and objects of the ontology Ont(T) refer to the classes and objects from the main ontology but do not modify the latter. No modification of the common ontology is performed when the system operates in user answering mode.

The next stage of work of the ontological-semantic analyzer in the considered mode is formation of a search query to the ontology basing on existing information about Q and T. The search query is formed using functions describing work with the ontology (see p.1.) and the Unit-terminology, the hierarchical dependency of which is presented in Fig. 2.

When developing the query to the ontology, one should consider not only the user-defined question Q itself, but also the text T in which an answer is to be found. It is related to the fact that the user-provided text can help in solving the tasks of lexical ambiguity that can arise when forming the query. For example, if the question contains the word “bank” which may mean either “coast” or “credit institution”, then the user-provided text T may help to solve the lexical ambiguity that arose in the question. It is more likely that the word “bank” is used in the meaning “coast” if the text T contains the words which are such classes or objects in the ontology that are close to the class or the object “bank” in the meaning “coast” of a river, a sea etc.

In case the query to Ont(T) has been successfully executed and returned data from the ontology, these data are provided to the user as an answer. In case an answer has not been found in Ont(T) or it has not satisfied the user, the search is continued in the general ontology. Using the explanatory component of the expert system, the user can learn how the system has obtained the solution.

IV. Ontology Learning Mode

This section will describe the working algorithm of the ontological-semantic analyzer operating as a part of the question answering system which is functioning in the ontology learning mode.

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 [21]. 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 ".

Queue with priority

In a classical definition a queue with priority is defined as an abstract data type allowing to store pairs (key, value) and supporting the following operations [22]:

1) **init** — initialize a new empty queue;
2) **insertToPriorityQueue** — insert a new element into the queue;
3) **remove** — remove and return the highest-priority element of the queue.

In this work we will use the "queue with priority" for removing facts from the working memory of the expert system. The fact of the expert system consists of the following: UnitCO, a link to the previous fact (left) and to the next fact (right), morphological characteristics and coordinates in text (the number of the sequence and the position of UnitCO in the sentence). In this context we use two values to define the element priority in the queue:

1) priority of the group that the considered semantic relation belongs to;
2) position of ***UnitCO*** in the analyzed sentence.

In this work we will consider the queue **Q** with priority, the elements of which will consist of the following triples:

1) ***a*** – the name of UnitCO;
2) ***sp*** – the priority of a semantic group which the semantic relation belongs to, one of the arguments being UnitCO;
3) ***pos*** – the position of UnitCO in the analyzed sentence.

We will say that the element ***(a, sp, pos)*** of the described queue ***Q*** has the highest priority if ***sp*** value of this element is minimal and ***pos*** value is maximal. Consequently, the elements of the queue with priority are sorted in ascending order of priorities of semantic dependencies groups. If the queue contains several elements with the same priority values of semantic groups, then such elements are sorted in descending order by the last ***UnitCO*** related to the considered element in the analyzed sentence.

Below we describe the rules to add the element ***(a', sp', pos')*** into the queue Q with priority for the case when Q contains the element ***(a, sp, pos)*** such that ***(a = a')*** and ***(pos = pos')***:

1) ***(sp'> sp)***, hence ***Q = Q\(a, sp, pos)U(a', sp', pos');***
2) ***(sp'<= sp)***, hence ***Q*** is not modified.

Basical ontological-semantic patterns

Let us call an ontological-semantic pattern the rule by which the expert system finds semantic dependencies between classes and objects in the analyzed text (where indivisible sense entities are marked and each of them is refered to a certain class or an object of the ontology). A basical ontological-semantic pattern which is a rule of the expert system consists of the left and the right sides. The left side of the pattern describes the conditions upon which the actions described in the right side are executed. So, for example, the left side of the pattern always describes a biconnected facts list, and can also describe boolean functions having the facts from that biconnected list as their arguments.

The right side of the pattern contains the list of actions each of which can: modify any fact of the ES (by modifying the relation in a corresponding UnitCO); queue for removal a fact of the ES having a certain priority of removal; other actions.

Below we provide an example of a basical ontological-semantic pattern that describes how the ontology is modified if in the analyzed text there has been discovered a fact X containing UnitCO being a class of the ontology (defined as a Property in the ontology and containing among the morphological characteristics the information that it is the singlular number), and what follows next is some fact Y containing UnitCO being a class or an object of the ontology, and it is known that in the text it is preceded by the fact X, and among the morphological characteristics of Y there is information that it is the singlular number of a noun.

```
when
{
$X : Fact (unitCO.type == "Class",
unitCO.hsOntAttrs contains "Property",
hsMorphAttrs contains "singular")
$Y : Fact (unitCO.type == "Object" ||
unitCO.type == "Class" , prev == $X,
hsMorphAttrs contains "singular", hsMorphAttrs
contains "noun")
}
then
{
Logic.insertOrUpdateRelation("Property",
$Y.unitCO.id, $X.unitCO.id, 1);
Logic.insertOrUpdateRelation("Property for",
$X.unitCO.id, $Y.unitCO.id, 1);
Logic.insertToPriorityQueue ($X, sp);
}
```

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 (as shown in Fig. 4).

The Table 1 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". Algorithm of ontology learning using the basical ontological-semantic patterns

Used notations:

1) P – the analyzed sentence;
2) p_i – i-th indivisible sense entity of the analyzed sentence P;
3) S – the set of all rules of the ES including the basical ontological-semantic patterns;
4) S_i – i-th rule of S;
5) Q – the queue with priorities.

TABLE I
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] AT: [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] AT: [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] AT: [Yesterday], [sport] [honors] [left] [for] [a camp].		
	(new semantic relationship is not found) AND isEmpty (Q) = false	} =>	Continue
		remove(Q, (sport, 2, [2])	Q = {(a camp, 3, [7]), (honors, 15, [4])}
4	[0] [4] [5] [6] [7] AT: [Yesterday], [honors] [left] [for] [a camp].		
	(new semantic relationship is not found) AND isEmpty (Q) = false	} =>	Continue
		remove(Q, (a camp, 3, [7]))	Q = {(honors, 15, [4])}
5	[0] [4] [5] AT: [Yesterday], [honors] [left] .		
	(new semantic relationship is not found) AND isEmpty (Q) = false	} =>	Continue
		remove(Q, (honors, 15, [4]))	Q={}
6	[0] [5] AT: [Yesterday], [left] .		
	Time (left, yesterday)	insert(Q, (yesterday, 7, [0]))	Q={(yesterday, 7, [0])}
		remove(Q, (yesterday, 7, [0]))	Q={}
7	[5] AT: [left] .		
	(new semantic relationship is not found) AND isEmpty (Q) = true	} =>	End

Aiming to check the usage possibilities of Apache Spark platform, we have implemented the programs for the latter. The first program computed in a distributed manner the inverted text index using the Lucene library on a 10-nodes cluster. After that, a second program executed in a distributed manner a large number of search queries using Lucene language to the distributed inverted index. The working results of the programs allowed to establish the following: a) a significant profit in working time of distributed operations of indexing and search in comparison with performing the same operations on a single computer; b) reasonability to implement the proposed algorithm on the Apache Spark platform.

Algorithm:

Step 1. Put in the input of the ontological-semantic analyzer the preprocessed sentence P (see the architecture of the question answering system) in natural language and consisting of indivisible sense entities p_i: P = (p_1, p_2, p_3, …, p_N).

Step 2. Match the indivisible sense entities p_1, p_2, p_3, …, p_N with the ontology data: associate with each p_i some $UnitCO_i$ — a class or an object from the ontology. Form the facts of the ES P_{Facts}, presented using a double-linked list where the first and the last facts ($Fact_0$ and $Fact_{N+1}$) are empty, and the rest are formed according to the fact definition: that is, $Fact_i$ contains $UnitCO_i$, the morphological information about $UnitCO_i$, the links to the left and the right facts, the fact position in the text.

Step 3. Add to the knowledge base of the expert system (S) all ontological-semantic patterns and other rules.

Step 4. Initialize the variable i which will store the sequential number of the considered rule as zero: i := 0.

Step 5. Using fast patterns matching algorithm, form the array G consisting of pairs (rule, fact list) in which we will put the rules from S and the corresponding facts from P_{Facts} with true left part.

Step 6. Sort the elements of the array G in the order of fulfilling the rules by the block of logical derivation.

Step 7. Check whether the value of the loop variable i has exceeded the limits of array G (i < |G|), which would mean that all elements of G has been looked through. If i < |G|, go to Step 8. Else go to Step 10.

Step 8. According to the right side of the rule G_i the following actions may be performed: update the facts of the ES; add facts to the queue for removal together with their priorities; etc.

Step 9. Increase the value of the loop variable i by 1: i := i + 1. Go to Step 7.

Step 10. Check whether the queue for removal Q is empty: if the queue is empty (isEmpty(Q) is true), then finish the algorithm execution.

Step 11. Set the variable b to be equal to the fact being removed from Q and having the highest priority, that is: element b := remove(Q).

Step 12. Update the links of the fact a (a=b.left) and the fact c (c=b.right) in the following way: a.right=c; c.left=a. In the working memory of the expert system: update the facts a and c; remove the fact b. Go to Step 4.

The proposed working algorithm of the ontological-semantic analyzer can be implemented in distributed manner. For instance, the analyzed texts can be stored in a distributed file system (such as HDFS). The following tasks can be distributed among the nodes of a computational cluster: the ontology construction and learning for each analyzed text; indexing of ontological-semantic graphs; search for the answers to the questions; etc. A promising platform for distributed implementation of the proposed algorithm on a cluster may be the system Apache Spark. This platform has already been integrated in the Hadoop ecosystem (HDFS, Hadoop YARN) and is a part of such popular integration projects as Cloudera, HortonWorks, MapR etc.

Aiming to check the usage possibilities of Apache Spark platform, we have implemented the programs for the latter. The first program computed in a distributed manner the inverted text index using the Lucene library on a 10-nodes cluster. After that, a second program executed in a distributed manner a large number of search queries using Lucene language to the distributed inverted index. The working results of the programs allowed to establish the following: a) a significant profit in working time of distributed operations of indexing and search in comparison with performing the same operations on a single computer; b) reasonability to implement the proposed algorithm on the Apache Spark platform.

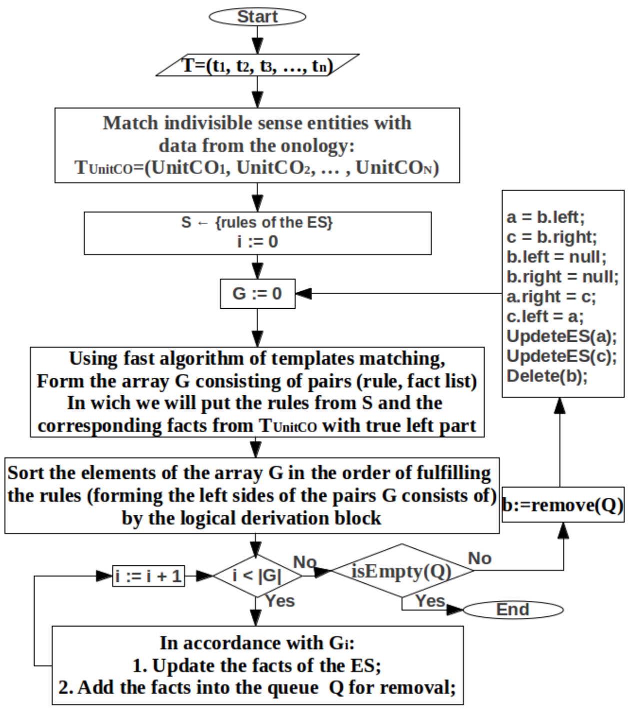
Start
T=(t1, t2, t3, …, tn)
Match indivisible sense entities with data from the onology: TUnitCO=(UnitCO1, UnitCO2, … , UnitCON)
S ← {rules of the ES}
i := 0
G := 0
a = b.left;
c = b.right;
b.left = null;
b.right = null;
a.right = c;
c.left = a;
UpdeteES(a);
UpdeteES(c);
Delete(b);
Using fast algorithm of templates matching, Form the array G consisting of pairs (rule, fact list) In wich we will put the rules from S and the corresponding facts from TUnitCO with true left part
Sort the elements of the array G in the order of fulfilling the rules (forming the left sides of the pairs G consists of) by the logical derivation block
b:=remove(Q)
i := i + 1
i < |G|
No
Yes
isEmpty(Q)
No
Yes
End
In accordance with Gi:
1. Update the facts of the ES;
2. Add the facts into the queue Q for removal;

Fig. 4. Logical diagram of the algorithm of automated ontology learning.

V. Conclusion

The work is devoted to development of architecture and prototype of a question answering system that uses data from the ontology. The structure and the functions to work with the ontology are described in this paper. A working algorithm of an ontological-semantic analyzer using basical ontological-semantic patterns with removal is programmatically realized in Java language. The program has been registered in the Rospatent [23]. The Drools system, that uses a fast pattern matching algorithm with PHREAK patterns, has been used as the expert system. Using the expert system Drools has ensured high working speed of the ontological-semantic analyzer. For instance, the described algorithm of the ontological-semantic analysis using the Drolls expert system and 2160 basical ontological-semantic patterns has determined 8213 semantic relations in 6390 ms in the text of E. T. A. Hoffmann's fairy tale “The Golden Pot”. Without the Drolls expert system, the implementation of the algorithm of the ontological-semantic analyzer works in average 6-8 times slower. The experiments were performed on an Intel Core i3 M CPU 2.27 GHz under OS Ubuntu 12.04.

References

[1] Moldovan, D. I., Harabagiu, S. M., Pasca, M., Mihalcea, R., Goodrum, R., Girju, R., and Rus, V.: LASSO – A Tool for Surfing the Answer Net. *TREC-8*, pp. 175-183, 1999.

[2] Scott, S., Gaizauskas, R.: QA-LaSIE: A Natural Language Question Answering System. *Proceedings of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence*, 2001.

[3] Monz, C., de Rijke, M.: Tequesta: The University of Amsterdam’s Textual Question Answering System. *TREC*, 2001.

[4] Mozgovoy, M. V.: Simple question-answering system, based on Russian semantic analyzer. *Vestnik of the SPb University*, series 10, issue 1, 116-122, 2006.

[5] Rubashkin, V. Sh., Kapustin, V. A.: Using term definitions in encyclopedic dictionaries for automated ontologies learning. *XI All-Russian united conference «The Internet and the modern society»*, SPb, 2008, http://old.conf.infosoc.ru/2008/pdf_CL/Rubashkin&Kapustin.pdf

[6] Mitrofanova, O. A., Konstantinova, N. S.: Ontologies as systems of data storage. *All-Russian competitive selection of the review analytical papers on the priority direction “Information telecommunication systems”*, 54 p., 2008.

[7] Aramaki, E., Imai, T., Kashiwagi, M., Kajino, M., Miyo, K. and Ohe, K.: Toward medical ontology using Natural Language Processing. http://www.m.u-tokyo.ac.jp/medinfo/ont/paper/2005-aramaki-1.pdf

[8] Hovy, E., Knight, K., Junk, M.: Large Resources. Ontologies (SENSUS) and Lexicons. www.isi.edu/natural-language/projects/ONTOLOGIES.html

[9] Rubashkin, V. Sh.: *Ontological semantics. Knowledge. Ontologies. Ontologically-oriented methods of informational text analysis*. Moscow, Fizmatlit, 2013.

[10] Rabchevsky, E. A.: Automatic ontology construction based on lexical-syntactic patterns for information retrieval. *Proceedings of the 11th All-Russian scientific conference RCDL'2009*, Petrozavodsk, 69-77, 2009.

[11] Rubashkin, V. Sh., Bocharov, V. V., Pivovarova, L. M., Chuprin, B. Yu.: The approach to ontology learning from machine-readable dictionaries. *Dialog — 2010*. http://www.dialog-21.ru/digests/dialog2010/materials/html/63.htm

[12] Orobinska, E. A.: Automatic Method Of Domain Ontology Construction based on Characteristics of Corpora POS-Analysis. *Proceedings of the XV All-Russian united conference «The Internet and the modern society» (IMS-2012)*, SPb, 209-212, 2012.

[13] Buitelaar, P., Cimiano, P.: Ontology learning and population bridging the gap between text and knowledge. *Series: Frontiers in artificial intelligence and applications*, V. 167, Amsterdam; Washington, DC : IOS Press, 2008.

[14] Gruber, T. R. A Translation Approach to Portable Ontology specification. Knowledge Acquition, 5(2), 1993.

[15] DBpedia, http://wiki.dbpedia.org

[16] Freebase, http://www.freebase.com

[17] Wikidata, https://www.wikidata.org

[18] Wikipedia, the free encyclopedia, https://ru.wikipedia.org

[19] Wiktionary, the free dictionary, https://ru.wiktionary.org

[20] Krizhanovsky, A. A., Smirnov, A. V.: An approach to the automated construction of a general-purpose lexical ontology basing on the wictionary data [in Russian]. *Izvestiya RAS, Conntrol theory and systems*, No. 2, 53-63, 2013.

[21] Sokirko, A. V.: Semantical dictionaries in automated text processing: as exemplified by the DIALING system. PhD. tech. sci. diss. [in Russian]. Moscow, 2001.

[22] Downey, A. B. *Think Python*. O'Reilly Media, 300 p., 2012.

[23] Mochalova, A. V. *Certificate on the state registration of the computer program “A program of text semantic analysis using basical semantic patterns with removal”*, No. 2015613430, Russia. 28.01.2015

Vladimir Kuznetsov is a scientist, mathematician, Professor, Doctor of technical sciences, Honorary worker of higher professional education of the Russian Federation, Honored worker of science of the Republic of Karelia. He is the author of over 200 scientific papers in the field of mathematical modeling and solution of applied optimization problems, as well as academic literature on the programming. He is one of the coaches of the teams who were winners of the world championship on programming (in 2007, 2008, 2010). His research interests include optimization techniques, Olympiad programming.

Vladimir Mochalov was born in Lyubertsy, Russia in 1985. He received the Ph.D. degree in electronic engineering from Moscow Technical University of Communications and Informatics. His research interests include networks structure synthesis, artificial intelligence, bio-inspired algorithms, query answering systems and Big Data.

Anastasia Mochalova was born in Petrozavodsk, Russia, in 1987. She received the bachelor's degree at Petrozavodsk State University, the master's degree in St. Petersburg State University of Aerospace Instrumentation. She is an external PhD student in technical sciences at Petrozavodsk State University. Her research interests include automated processing of natural language texts, development of question-answering systems, automation of ontologies creation, development of the semantic analyzer.