UDC 004.93

TREE-BASED SEMANTIC ANALYSIS METHOD FOR NATURAL LANGUAGE PHRASE TO FORMAL QUERY CONVERSION

Litvin A. A. – Postgraduate student of department of Microprocessor Technology, V.M. Glushkov Institute of Cybernetics, Kyiv, Ukraine.

Velychko V. Yu. – PhD, Senior researcher of department of Microprocessor Technology, V.M. Glushkov Institute of Cybernetics, Kyiv, Ukraine.

Kaverynskyi V. V. – PhD, Senior researcher of department of Abrasion- and Corrosion-Resistant Powder Construction Materials, I.M. Frantsevich Institute for Problems of Material Science, Kyiv, Ukraine.

ABSTRACT

Context. This work is devoted to the problem of natural language interface construction for ontological graph databases. The focus here is on the methods for the conversion of natural language phrases into formal queries in SPARQL and CYPHER query languages.

Objective. The goals of the work are the creation of a semantic analysis method for the input natural language phrases semantic type determination and obtaining meaningful entities from them for query template variables initialization, construction of flexible query templates for the types, development of program implementation of the proposed technique.

Method. A tree-based method was developed for semantic determination of a user's phrase type and obtaining a set of terms from it to put them into certain places of the most suiting formal query template. The proposed technique solves the tasks of the phrase type determination (and this is the criterion of the formal query template selection) and obtaining meaningful terms, which are to initialize variables of the chosen template. In the current work only interrogative and incentive user's phrases are considered i.e. ones that clearly propose the system to answer or to do something. It is assumed that the considered dialog or reference system uses a graph ontological database, which directly impacts the formal query patterns – the resulting queries are destined to be in SPARQL or Cypher query languages. The semantic analysis examples considered in this work are aimed primarily at inflective languages, especially, Ukrainian and Russian, but the basic principles could be suitable to most of the other languages.

Results. The developed method of natural language phrase to a formal query in SPARQL and CYPHER conversion has been implemented in software for Ukrainian and Norwegian languages using narrow subjected ontologies and tested against formal performance criteria.

Conclusions. The proposed method allows the dialog system fast and with minimum number of steps to select the most suitable query template and extract informative entities from a natural language phrase given the huge phrase variability in inflective languages. Carried out experiments have shown high precision and reliability of the constructed system and its potential for practical usage and further development.

KEYWORDS: natural language processing, graph data base, semantic analysis, formal query, decision tree, ontology.

ABBREVIATIONS

SPARQL is a query language to the data presented by the RDF model;

RDF is a Resource Description Framework;

OWL is a Web Ontology Language;

CYPHER is a query language for graph databases accepted in Neo4j;

LODQA is a Linked Open Data Question Answering.

NOMENCLATURE

A_C is an accuracy criterion;

A is a subject model on which the text is interpreting;

a_i is a possible word t_{ij} meaning;

C_L is a left context of the grammar unit;

C_R is a right context of the grammar unit;

D is a dictionary of a natural language in the alphabet X;

F_1 is a complex criterion to estimate precision and recall;

F is a formalization for the set of the words in the alphabet X;

F_n is a number of false negatives;

F_p is a number of false positives;

L is a formalization for a natural language on the given alphabet;

P_r is a precision criterion;

P is a formalization for the aggregate of a natural language grammar rules;

p_i is one of the grammar rules of a natural language;

R is a recall criterion;

R_E is a formalization for the aggregate of a natural language grammar relationships;

R_{pi} is a formalization for a binding of grammar rule and relationship;

S is a dictionary of words defenitions for the language L;

T is a currently considered natural language text;

T_p is a number of true positives;

T_n is a number of true negatives;

t_i is a sentence of a currently considered natural language text;

t_{ij} is a grammar unit of a sentence;

X is a formalization for an alphabet of some natural language;

γ is a relationship that defines meanings and types of words in a dictionary of the language;

Π is a predicates signature;

DOI 10.15588/1607-3274-2021-2-11

π^{kr} is an atomic predicate;

τ_i is a word t_{ij} possible type;

φ is a relationship of the text interpretation on the subject model.

INTRODUCTION

Nowadays dialog and reference program systems are becoming wide-spread and convenient. They help to automate frequent typical question answering, to obtain relevant information about an actual problem or to perform a necessary sequence of action. They can reduce the burden for human consultants in the routine of answering typical, many times asked and bored questions, and also being asynchronous a program system can serve several clients at the same time. It is obvious that for many cases the most comfortable and friendly for a user is a natural language interface of such system, which means that the user enters his question in a natural language. Hence, in the most of situations such dialog system is, actually, natural language interface of a database.

The problem of natural language database interfaces building is not new. Since computers have become widespread in the most aspects of life and work and appear common for large numbers of people even far from computer science and calculation tasks the problem of user's friendly interfaces creation become actual. And one of the directions of such interfaces is a natural language interface for databases that is highly desirable for persons who not familiar with programming and formal query languages. In contrast, asking the system using natural language seems rather more natural and convenient for an ordinary person. Despite the long existence of the problem, the development of such interfaces and new approaches to their construction continues [1], and there are certain reasons for this. Due to the complexity of the task of interpreting the semantics of queries in a natural language, the development process moved with varying success, experiencing ups and downs. The processing of an incoming request by a previously prepared semantic model may not be processed correctly for other requests. New attempts to create natural language interfaces for databases are constantly being made. Nevertheless, far from all of them turn out to be quite successful and indeed make a contribution to the state of the problem. However, every new research in this area is valuable because it brings new ideas and provides a better understanding of what really works and what does not.

The object of study is a natural language interface for ontological graph databases.

The subject of study is the development of methods for natural language of inflective type conversion to formal queries for graph databases.

The purpose of the work is to create a semantic analysis method for the input natural language phrases semantic type determination and obtaining meaningful entities from them for query template variables initialization, to const the flexible query templates for the needed semantic types and to develop and test a program implementation of the proposed technique.

1 PROBLEM STATEMENT

The main purpose of this work is to develop a program system that is able to return as its output formal queries in SPARQL or Cypher query languages for graph databases using as input interrogative and imperative phrases in a natural language of inflective type. The method used for this task should be easily interpretable, be able to be easily corrected, supplemented and adapted, be fast and not resource intensive.

To solve this problem the following tasks are to be done:

1. To develop a tree-based method for semantic analysis of a natural language phrase;
2. To construct the scheme of the decision tree used in the developed system for determination of semantic type of the input natural language phrase;
3. To create flexible templates of formal queries for SPARQL and Cypher corresponding to the required semantic types;
4. To come out with the method for obtaining meaningful entities from the phrase to initialize templates variables;
5. To develop a program realization of the business logic separate from the tree and templates files;
6. To integrate the natural language to formal query conversion modulus as a service into a multi-agent dialog system;
7. To carry-out experiments for testing the developed system performance.

To test the quality of the developed system results the following criteria are to be used: accuracy, precision, recall and F_1-score, which are common for such kinds of systems [2]. These metrics are calculated as follows:

$$A_C = \frac{T_p + T_n}{T_p + F_p + F_n + T_n}, \quad (1)$$

$$P_r = \frac{T_p}{T_p + F_p}, \quad (2)$$

$$R = \frac{T_p}{T_p + F_n}, \quad (3)$$

$$F_1 = \frac{2 \cdot P \cdot R}{P + R}. \quad (4)$$

Formal mathematic problem characteristic is given here according to [3]:

Let X is an alphabet of some natural language and $F(X)$ is the set of the words in X. $L \subseteq F(X)$ is a natural language on the given alphabet and its sentences are constructed according to the grammar rules $P = \{p_i : i = 1,\ldots,m\}$. The grammar rules define the relationships $R_E = \{R_{p_i} : p_i \in P\}$ each from them is corresponding to a grammar rule. Let $T \in L(X)$ is a natural language L text

© Litvin A. A., Velychko V. Yu., Kaverynskyi V. V., 2021
DOI 10.15588/1607-3274-2021-2-11

and $t_i \in T$ are sentences of the text *T*, $i = 1, 2,\ldots, n$. Every sentence of the text *T* has its structure $t_i=t_{i1}t_{i2}\ldots t_{im}$, where t_{ij} are grammar units that form a sentence. $C_L(t_{ij}) = t_{i1}\ldots t_{ij-1}$ is the left context of the word t_{ij} and $C_R(t_{ij}) = t_{ij+1}\ldots t_{im}$ is the right context of it in the sentence t_i. *S* is a dictionary of the language *L(X)* where the definitions of words t_{ij} are. $\gamma \subseteq T \times S$ is a relationship that defines meanings and types of words in *S*. $A = (D, \Pi)$ is the subject model on which the text *T* is interpreting. $\varphi \subseteq T \times A$ is the relationship of the text *T* interpretation on the model $A = (D, \Pi)$. Predicates signature $\Pi = \{\pi^{k1},\ldots, \pi^{kr}\}$ includes atomic predicates to build more complicated formulas. Every atomic predicate has its type. Relationship γ is evaluated as:

$$\gamma(t_{ij}) = \{(a_1, \tau_1), (a_2, \tau_2), \ldots, (a_s, \tau_s)\} \qquad (5)$$

The relationship φ if the model $A = (D, \Pi)$ is defined can be determinate as following:

$$\begin{aligned}&\varphi(t_i) = \\ &= \{\varphi(\gamma(t_{i1})\gamma(C_R(t_{i1}))), \varphi(\gamma(C_R(t_{i2}))\gamma(t_{i2})\gamma(C_R(t_{i2}))), \qquad (6)\\ &\ldots, \varphi(\gamma(C_L(t_{in}))\gamma(t_{in}))\},\end{aligned}$$

$$\varphi(\gamma(t_{ij})) = \gamma(\varphi(t_{ij})),$$

$$\varphi(\gamma(C_L(t_{ij}))) = C_L(\gamma(\varphi(t_{ij}))),$$

$$\varphi(\gamma(\pi_r^k(p_1,\ldots,p_k))) = \gamma(\varphi(\pi_r^k))(\varphi(\gamma(p_1),\ldots,\gamma(p_k))).$$

It should be noticed, that here $\gamma(\varphi(\pi_r^k))$ is a predicate coordinated with arguments $\gamma(p_1),\ldots,\gamma(p_k)$.

Thus, the main formal goal of a semantic analysis technique development is to construct a method for γ and φ implementation.

2 REVIEW OF THE LITERATURE

In [4] proposed a transformation system that includes three main components: the first component converts a natural language query into a query tree, the second component interactively checks the conversion, turning to the user, and the third component converts the query tree into SQL. The main disadvantage of this approach is a user's checking the constructed query tree, which makes such system not convenient for a user not familiar with formal queries and databases.

Methods based on statistical machine learning and neural networks have begun to appear in recent years. A methodology is proposed in [5] implements reinforcement training. This approach suggests using a deep neural network to translate natural language phrases into corresponding SQL queries. This method takes advantage of the structure of SQL queries to significantly reduce the output range of generated queries. Despite the huge data set in the training sample, the accuracy of the model was not high: the accuracy of execution was 59.4 %, the accuracy of the logical form was 48.3 %. Thus, for the real implementation of such system, an extensive collection of training sets will be required, which is difficult to provide.

As an example of converting natural language queries into formal queries in SPARQL can be considered the LODQA system presented in [6]. It parses a natural language phrase and creating a graphical representation of the request, which is called a pseudo graphic template. The pseudo graphic template is a graph pattern for finding the target graph of RDF subgraphs that match it. Natural language term can be normalized for more than one RDF term due to ambiguity. Therefore, from one pseudo graphic template, more than one linked template can be obtained by normalization. In order to take into account the structural inconsistency between the attached pseudo graphic template and the actual structure in the target data set, it tries to generate SPARQL queries for all possible structural variations. The considered LODQA system is focused on the English language only. Detailed features of its functioning are not given in [6], limiting only to a general description and analysis of work examples.

Another approach to natural language to SPARQL conversion is considered in [7] PAROT framework. PAROT adopts an approach that generates the most likely triple from a user query. The triple is then validated by the lexicon. It relies on a dependency parser to process user's queries to user triples. The user triples are then converted to ontology triples by the lexicon. The triples generated by the lexicon are used to construct SPARQL query that fetches the answers from the underlying ontology. Testing the PAROT framework by the authors of [7] showed that for simple questions it demonstrates about 81 – 82 % precision, about 43 – 56 % for complex questions, and for a specific thematic dataset (geography) precision was up to 88 %. But even the authors pay attention to some weaknesses of PAROT: it has low precision and recall when processing aggregation based questions.

In spite most of natural language to formal query conversion systems deal with English, works is also proceeding for other languages. For example, in [8] proposed a method to generate SPARQL queries from Korean natural language queries.

Taking into account the success in machine translation systems development using neural networks achieved in recent years it is no wonder that approach is trying to be applied to translation from a natural language to a formal query language such as SPARQL. For now, there are only a few examples of such approach, for example [9]. Some tricks to avoid or at least minimize criticality of typical machine translation mistakes are taken into account there. For instance, to train the model they use not SPARQL queries as they are but previously converted them into special sequences where language symbols and constructions are encoded as constant symbol sequences. Some constant query structure elements were omitted or abbreviated. Thus the translations result in this method is a specific sequence that is actually an instruction, by which it is possible to build a sufficient SPARQL query. Despite the authors of [9] claimed a rather good accuracy of their

© Litvin A. A., Velychko V. Yu., Kaverynskyi V. V., 2021
DOI 10.15588/1607-3274-2021-2-11

model its behavior on others datasets related to different ontologies is still a question.

An example of natural language to SPARQL conversion open source realization is FREyA [10]. It is available on GIT-hub [11]. FREyA is an interactive Natural Language Interface for querying ontologies. It uses syntactic parsing in combination with the ontology-based lookup in order to interpret the question and involves the user if necessary. The user's choices are used for training the system in order to improve its performance over time. It deals with English language. In [11] some examples of natural language questions are given can be converted to SPARQL using FREyA. It seems that the query formation by FREyA very depends on the ontology data, and its configuration needs to be tuned to the certain ontology.

Conversion systems for natural language questions to Cypher (a query language for graph database Neo4j) are rather less developed, but works in this area are also proceeding. Analyzing posts in topical internet forums it seems to be that development of such tool is highly desirable. Here are a few examples of such works: [2, 12]. The system proposed in [12] is rather primitive. The queries need to be of a pre-defined structure; actually it needs a file with ready natural language sentences where some words are replaced by placeholders and matching Cypher templates. The described approach has its advantages and disadvantages. The main advantage is simplicity, which guarantees that the result will be just obtained or not obtained without appearing of strange situations when wrong or not completely correct result springs up. But obviously, there are a lot of drawbacks. First of all, for a real big system, a large number of phrases templates is needed which involves all possible users' questions considering their variety.

The flexibility of the query template approach can be increased if rigid phrases templates will be replaced with semantic analysis of an input phrase which also recognizes the entities (words) that are to be substituted into the query template [13]. In this work is considered the problem of selection of a correct query template and singling out corresponding entities basing on the user's phrase.

3 MATERIALS AND METHODS

The only input information coming from the user is a phrase in Ukrainian language. It should be a single sentence. If there ate several sentences in the user's replic, each of them will be separately considered by the system. For this purpose the input text is tokenized to a list of sentences. Each sentence is then tokenized to a list of words. These operations are conforming so called preliminary graphematic analysis. Then these words list are cleaned from senseless and not informative words (interjections, emotion expressions, senseless introductive words). For this task lists of such words are pre-prepared. Each of the words from the input phrase is lemmatized and comparing with words from this lists. If it matches this word is to omit from the sequent processing. Obtained in such way input material than used for semantic analysis.

© Litvin A. A., Velychko V. Yu., Kaverynskyi V. V., 2021
DOI 10.15588/1607-3274-2021-2-11

The first goal of the semantic analysis is to determine the semantic type of the input phrase, which is φ from the formal problem statement, i.e. what kind of information the user wants to receive. The considered here system deals only with interrogative and imperative sentences. Narrative sentences where nosing is implicitly asking are out of consideration in this work.

The criteria of the semantic type determine are facts of the presence of certain words and words sequences (including preposition) in certain forms. The main complicity is that there are a great number of possible semantic intents φ and many of them could be represented in plenty of different ways (*A*). Hence, runtime enumerating all possible options is a long and unproductive way. In the basis of the proposed here solution for is a tree approach when analysis is going step by step and on each step only a few factors are considered that allows one to exclude many of others variants so they do not need sequent consideration. The determination algorithm is proceeding until all the necessary and sufficient conditions for the certain semantic type appear observed. The method also allows at the same process to find input entities to be substituted into the query template. Thus, this determinative tree is actually γ from the formal problem statement.

Let us consider the proposed determination method in more detail.

Even in inflective language, at least in Ukrainian and Russian words are not going completely random but some patterns of their sequence exist, which significantly simplifies the analysis. For instance, most types of questions, excluding so called general questions, begin with question words ("що" – "what", "коли" – "when", "як" – "how", etc.), which are the most crucial factor in the process of the semantic type determination. Imperative sentences begin with a verb in imperative form, or with appeal followed by such verb. Preposition in Ukrainian and Russian languages can go only before nouns, pronouns and name groups (the linked group of nouns and adjectives that in aggregate describe one entity). Thus, the tree algorithm analyses not just presence of a certain word but it goes through the words of the phrase as a sequence.

The algorithm could be described like a frame mowing through the sequence of words. The size of this frame is from one word to several and is equal to a number of words in consideration during the current condition analysis. The tree is a-ary, so there is no limitation of outgoing links from its node to the nodes of the lower level. The tree, its conditions, should be constructed in such manner that only one possible way must exist to the lower level from a node; i. e. algorithm should go through the tree without dividing. If not only one from the conditions options of the node are suitable, one that is the most suitable must be selected (but such situations are reasons to improve the conditions, and may be add another level of conditions). More suitable means the following:

– if the number of checked words (size of frame) in condition A is bigger than it in condition B and both conditions are matching the current situation, the preference is to be given to condition A (with bigger frame size);

– if condition A matches only by word characteristic (verb in the certain form, noun in some case, etc) and condition B matches by the text (at least in lemma form), the preference is to be given to condition B (text matching);

– if condition A matches only by part of speech (just verb, or noun, or adjective etc.) and condition B matches also by word form characteristic (case, tense, verb form etc.), the preference is to be given to condition B (word form matching).

On each step, only one of the conditions should be chosen, which determines the conditions checked in the following step. If there is no matching condition the only result is that the answer can't be obtained, which can appear if the conditions system in the tree is not full. Such situation is perfectly acceptable if the tree is specialized for a narrow subject area and number of existing query templates is rather restricted. Such approach makes the system easier and faster by early screening out the questions for which there are no answers in the database. Each following step could restrict the number of possible query templates or just check the next necessary condition. At the end, only one of the possible templates should remain and all the necessary and sufficient conditions for it should be observed. If in some case more than one query template is remain (that is not desirable) several formal queries are to be formed, but that is evidence of the initial phrase nebulosity or imperfection of the determination system. For some cases the frame size is flexible, it is necessary when there expected a possibility of a words group with some characteristics but of not determined length (name group, homogeneous parts of the sentence, parts of full name, date, etc.). The frame shift for the following step depends on the actual frame size of the selected condition, so each time only new words appear in consideration, crossover shifts of the frame are not provided in this method. To determine the semantic type of the user's phrase (γ) and the corresponding query template is important, but not enough. A set of entities substituted to the query template is also needed. Selection of those entities is performed after the template determination, because they depend on it. For this purpose in the end node of the solution tree are also indicated positions of the way in the tree (by level) from where corresponding variables of the template should be initialized. If in some position a number of entities are present (described above cases of flexible size frames), a list will be obtained to initialize the template variable. If some of template variables are initialized as list, the condition row of the template during variables values substitution will be repeated for the each value, if this variable in the template is marked as "allow list", otherwise its value shall be made by joining the list members. The first option ("allow list") is better for homogeneous parts of the sentence and the second one – for name groups, which are stored in the ontology (graph database) in such joined way. In some cases of narrow subjected ontologies some of the entities are not to be substituted from the user's phrase but are predefined in the template, which makes it more reliable in reasons of ontology answer obtaining possibility. Also, there could be some cases when obtained from the user's phrase entity is replaced by its synonym which presence in the ontology is guaranteed but of the user's word is not. If on the expected graph way position there is no entities to initialize the corresponding template variable (undesirable, degenerate, but also possible case), this condition string shall be omitted in the forming query, so the query becomes wider.

Here we limited to just a brief description of the proposed query template method without linguistic and semantic details. They very depend on the language, the subject area and the peculiarities of the ontology structure accepted in the system. It only should be noticed, that the full and complete system of semantic types of questions is not always necessary for most of the databases. It could be limited to only those types of queries types for which the answers in this ontology are expected and provided.

The common scheme of the provided natural language to formal query algorithm is illustrated by a UML activity diagram on Fig. 1.

Putting query templates and the template selection tree scheme into separate files allows a developer to adopt the system to certain ontology without touching the program code logics. The program code is in a python file to which a linguistic analyzer is bind to deal with a specific natural language.

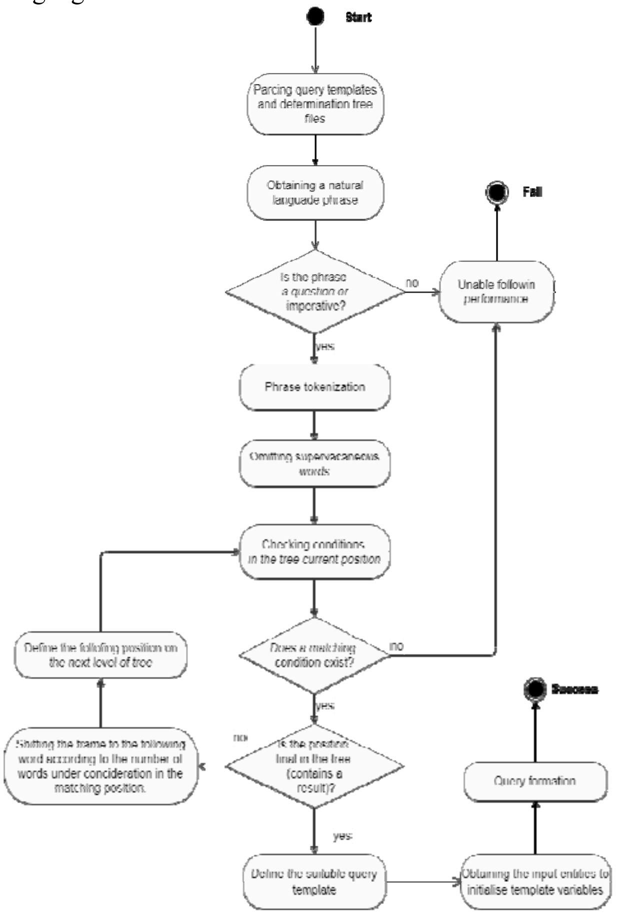
Start
Parcing query templates and determination tree files
Obtaining a natural languade phrase
Fail
Is the phrase a question or imperative?
no
Unable followin performance
yes
Phrase tokenization
Omitting supervacaneous words
Checking conditions in the tree current position
Define the folloing position on the next level of tree
Does a matching condition exist?
no
yes
Success
Shifting the frame to the following word according to the number of words under concideration in the matching position
no
Is the position final in the tree (contains a result)?
Query formation
yes
Define the suitable query template
Obtaining the input entities to initialise template variables

Figure 1 – UML activity diagram of the process of natural language phrase to a formal query conversion

© Litvin A. A., Velychko V. Yu., Kaverynskyi V. V., 2021
DOI 10.15588/1607-3274-2021-2-11

The input information is a natural language user's phrase and the output is a SPARQL or Cypher formal query. The modulus is included to a program agent (web service) shell, so it could easily become a part of a natural language dialog or reference system.

4 EXPERIMENTS

To test the developed system we used the OWL ontology of letters written to the famous Ukrainian writer of XX century Oles Honchar. This ontology was semiautomatically constructed on the basis of the digest book of the letters [14]. Let us briefly describe the ontology structure. Now the working system is available by the following URL: https://oles-gonchar-bot.herokuapp.com This dialog web application involves not only the ontology for Oles Honchar letters but also several other ontologies devoted to the literary work of this writer. All ontologies are in OWL (RDF/XML) and SPARQL is used to query them. Neo4j graph database and Cypher queries are used in another developed by us dialog system which is a virtual financial consultant with a natural language interface (in this case – Norwegian). This application is available by the next URL: http://178.128.245.158:8888/chatbot/

For the system performance testing serial of experiments were carried out. As input were used natural language (Ukrainian) interrogative and imperative phrases. To estimate obtaining negative and positive results 30 of the phrases were directly related to the considered in the ontology subject (letters were written to Oles Honchar) and the answer on them supposed to be in the ontology, the others 30 of them were made specially to obtain a negative result. These phrases for a negative result obtaining were not completely meaningless or grammatically incorrect (because for such ones there is no a certain positive result). They actually could be divided into three groups: in the first group phrases are devoted to Oles Honchar letters but there is no a supposed answer to them (20 phrases), in the second one the phrases were about letters but not to Oles Honchar (10 phrases), and in the third one phrases were about Oles Honchar but not about letters (10 phrases). In total, a set of 60 test phrases was used in the testing all phrases were grammatically correct Ukrainian sentences. Not obtaining an answer to a phrase about a completely different subject is trivial, thus they were not used for the testing.

An answer of the system was considered as a true positive if the answer was informative and was given as it was supposed to be. True negative result is the absence of the requested information (but not wrong or not relevant information!) but only in the case when we expected it. As false negative we considered all results where a certain answer was expected but was obtained either wrong answer, not a complete answer, or no answer. False positive result may contain any information (complete or not, maybe even not completely correct) in the case when the absence of an answer was expected.

The obtained scour was used for the calculation of accuracy, precision, recall and F_1 criteria.

Also the time intervals of the phrase analysis and query formation process were measured. These times include just these process but not also the times of query execution, messages between agent and to the user sending, page rendering etc.

5 RESULTS

The scour for each type of the obtained during the testing results is shown in table 1. The values of formal estimation criteria are given in table 2.

Table 1 – Experimental results score

Type of result	Number of results
T_p	24
T_n	29
F_n	6
F_p	1

Table 2 – Evaluation criteria of the system performance results values

Criterion	Value
Accuracy	0.883
Precision	0.960
Recall	0.800
F_1	0.873

It should be noticed that one answer considered here as a false positive is actually not a completely correct answer, but the true negative result for it ought to be “no answer”. From the false negative results actually in four cases there were no answer and in two ones the answer was but incorrect.

The average time interval of the analysis and query formation process was 6 ± 3 ms. Any statistically valuable decadence of it from the phrase length and complicity was not observed.

6 DISCUSSION

Generally, according to the values of the criterion, the developed system seems to be rather qualitative and is on the level of claimed in [2, 7, 10]. Thus the proposed method may be workable and practically acceptable. But some of its peculiarities revealed in the experimental testing are to be discussed in more detail.

The system shows a high precision criterion value because it is not inclined to false positive results. The cause of this is the sensitivity of the tree-based method and then ontology querying system to not subjected information. It rather gives no results in this case than some results. Additional information, which we did not expect to receive, but suddenly received, simply cannot be taken with this approach to the construction of the ontology and queries to it. But sometimes seldom if the input information is rather similar to one stored in the ontology some answers (may be not very relevant) could be obtained.

Unlike the precision, recall criterion is not so high. This is mostly because the system tends to not find answers even if they are in the database. The reason for this is some incompleteness of the decision tree rules (some possible constructions or/and distinguishing words were not taken into account). Another cause may be that some words which should to be ignored actually were not putted to the corresponding lists. This type of problem could be solved quite easily – by adding the necessary lacking rules and words lists. But the problem of wrong answers obtaining is some more complicated: it needs increasing of the semantic analysis profundity and precision for distinguishing more subtle aspects, also taking into account homonymy and correct interpretation of pronouns.

In the whole, the developed method has the potential to show a rather good accuracy and high performance. It is scalable and could be tuned both to small narrow subjected ontologies or wide and complicated ones. Moreover, it is rather simply correctible and the obtaining results could be easily explained and if needed it is clear what and where the corrections and additions have to be made. It is the same suitable for SPARQL and Cypher queries, and may by others query languages used in graph or relations databases.

For the moment it is elaborated only to some narrow subjected ontologies that need a rather limited variety of query templates and templates are constructed not for all the possible semantic types of questions. In the future, we are to make a more thoughtful and complicated decision tree to determine most of the semantic types of questions and imperative sentences.

As a main disadvantage of the proposed method is the need for manual creation of the query templates and the decision tree to determine the most suitable template, which is also very depend on the ontology structure. That may take a lot of effort. Thus, it seems to be suitable for the systems using narrow subjected ontologies that need to have high accuracy and stability of the answers and where databases are tend mostly growing in data size but not significantly change their structure.

CONCLUSIONS

A method is proposed for the conversion of natural language questions and imperative phrases to formal queries in SPARQL and Cypher query languages used in graph databases. The proposed technique assumes the presence of several query templates corresponding to each for a special semantic type of question. Meaningful entities extracted from the user's natural language phrase are substituted into the corresponding query template. For the most suitable query template selection a tree based semantic analysis method is proposed. The method assumes that the frame is shifting through the words list of the phrase considering on each step one or several words. These words are analyzed to match one of the conditions on the current tree position. The most matching condition determines the following position on the next level of the tree. The process proceeds until there will remain the only option of query template and all the sufficient conditions for the corresponding semantic type are proofed to be observed. Then depending on the selected query template input entities for it are taken from the given positions of the previous consideration.

The scientific novelty of the proposed approach is following: a technique of automatically conversion of a natural language phrase to formal queries in SPARQL and CYPHER query languages was further developed, the peculiarity of the presented method is using flexible query templates which selection and variables substitution is controlled by the tree-based semantic analysis, development and application of such methods is currently little elaborated; important feature of the proposed approach is its capability for inflective languages, especially, Ukrainian and Russian, for which such tools are poorly developed for the moment; were investigated the efficiency and possibilities for practical application of the proposed tree-based semantic analysis approach, standard criterions values were estimated, data on the assessment of this method were not previously known.

The practical value of the obtained results is that the proposed method seems to be useful in dialog and reference program systems which use graph databases either OWL/RDF or Neo4j based. The performance results of a program that implemented the proposed technique are clearly interpretable and explainable, which makes this approach highly customizable, fixable, and extensible. The tree-based method is rather scalable and could be suitable either for small narrow-subjected databases or big and complicated ones.

Quantitative indicators of the research results showed during the system testing are expressed by the following evaluation criteria values: A_C = 0.883, P = 0.960, R = 0.800, F_1 = 0.873. The obtained testing results seem rather reassuring and promising. Thus, the developed method is useful in constructing dialog and reference systems with a natural language interface. Also it shows the ability to easily corrections and editing if needed.

Prospects for further research are seemed as follows: investigation of different an more complicated graph data bases which represent more possible types of semantic relationships allocation explicitly and implicitly existing in them and constructing of new types of formal query templates for this purpose; further developing the proposed here semantic analysis technique for more (both general and subject oriented) semantic types determination, including mixed ones, and methods of meaningful concepts obtaining for them; development of methods that are able to as better as possible fit the terms obtained from an input natural language phrase to the corresponding values of the nodes in the graph database, which increases the possibility of an answer obtaining even for cases of long multiword terms in the nodes and expanding and narrowing an input term context regarding them.

ACKNOWLEDGEMENTS

Presented here investigation and results are the part of the research performed in the framework of the department project which is carried out in The V.M. Glushkov Institute of Cybernetics "Development of the methods for creating and using tools for working with knowledge bases of information systems for research" (state registration number 0119U002226).

REFERENCES

1. Galitsky B. Developing Enterprise Chatbots. Learning Linguistic Structures. Berlin, Springer, 2019, 566 p.
2. Sun C. A Natural Language Interface for Querying Graph Databases: master's thesis … master in computer science and engineering. USA, Massachusetts Institute of Technology, 2018, 69 p.
3. Palagin O. V., Krivij S. L., Bibikov D. S., Velichko V. Ju. ta in. Formal-logical approach to building analysis systems of knowledge in different domains, *Problems in progtamming*, 2010, No. 2–3, pp. 382–389.
4. Li F., Jagadish H. V. Understanding natural language queries over databases, *SIGMOD Record,* 2016, Vol. 45, pp. 6–13. DOI: 10.1145/2949741.2949744
5. Zhong V., Xiong G., Socher R. Seq2sql: generating structured queries from natural language using reinforcement learning, *2017 [Electronic resource].* Access mode: https://arxiv.org/pdf/1709.00103.pdf. arXiv: 1709.00103
6. Shaik S., Kanakam P., Hussain S. M., Suryanarayana D. Transforming natural language query to SPARQL for semantic information retrieval, *International Journal of Engineering Trends and Technology*, 2016, No. 7, pp. 347–350. DOI: 10.14445/22315381/IJETT-V41P263
7. Ochieng P. PAROT: Translating natural language to SPARQL, *Expert Systems with Applications,* 2020, No. 5, pp. 1–16. DOI: 10.1016/j.eswa.2021.114712
8. Jung H., Kim W. Automated conversion from natural language query to SPARQL query, *Journal of Intelligent Information Systems*, 2020, Vol. 55, pp. 501–520. DOI: 10.1007/s10844-019-00589-2
9. Yin X., Gromann D., Rudolph S. Neural machine translation from natural language to SPARQL, *Future Generation Computer Systems*, 2021, Vol. 117, pp. 510–519. DOI: 10.1016/j.future.2020.12.013
10. Damljanovic D., Agatonovic M., Cunningham H. FREyA: an interactive way of querying linked data using natural language, *The Semantic Web: ESWC 2011 Workshops,* 2011, pp. 125–138. DOI: 10.1007/978-3-642-25953-1_11
11. GIT-hub: FREyA documentation [Electronic resource]. Access mode: https://github.com/nmvijay/freya
12. GIT-hub Convert English sentences to Cypher queries documentation [Electronic resource]. Access mode: https://github.com/gsssrao/english2cypher
13. Litvin A. A., Velychko V. Yu., Kaverynskyi V. V. Method of information obtaining from ontology on the basis of a natural language phrase analysis, *Problems in progtamming*, 2020, No 2–3, pp. 322–330. DOI: 10.15407/pp2020.02-03.322
14. Kiral' S. S. recenzenti, ta in. pid zagal'n. red. M. Stepanenka Listi do Olesja Gonchara. Kyiv, Sakcent Pljus, Vol. 1, 1946–1982, 2016, 736 p.

Received 09.03.2021.
Accepted 29.04.2021.

УДК 004.93

БАЗОВАНИЙ НА ДЕРЕВІ МЕТОД СЕМАНТИЧНОГО АНАЛІЗУ ФРАЗИ ПРИРОДНОЮ МОВОЮ ДЛЯ ПЕРЕТВОРЕННЯ ЇЇ У ФОРМАЛЬНИЙ ЗАПИТ

Литвин А. А. – аспірантка, аспірантка відділу мікропроцесорної техніки, Інститут кібернетики ім. В. М. Глушкова, Київ, Україна.

Величко В. Ю. – канд. техн. наук, доцент, старший науковий співробітник відділу мікропроцесорної техніки, Інститут кібернетики ім. В. М. Глушкова, Київ, Україна.

Каверінскій В. В. – канд. техн. наук, старший науковий співробітник відділу зносо- і корозійностійких порошкових конструкційних матеріалів, Інститут проблем матеріалознавства ім. І. М. Францевича, Київ, Україна.

АНОТАЦІЯ

Актуальність. Ця робота присвячена проблемі побудови природномовного інтерфейсу для отримання інформації з графових баз даних. Основна увага приділяється методам перетворення фраз природною мовою у формальні запити на мовах запитів SPARQL та CYPHER.

Мета. Цілями роботи є створення методу семантичного аналізу типу вхідних природномовних фраз та виділення з них значущих сутностей для ініціалізації змінних шаблону запиту, побудова гнучких шаблонів запитів для відповідних семантичних типів фраз, розробка програмної реалізації запропонованого способу.

Метод. Розроблено метод, що базується на дереві прийняття рішень, для визначення семантичного типу фрази користувача і отримання з неї набору понять, для підстановки їх у певні місця найбільш підходящого шаблону формального запиту. Пропонована методика вирішує завдання визначення типу фрази (що безпосередньо пов'язано з критерієм вибору шаблону формального запиту) і отримання значущих понять, для ініціалізації змінних обраного шаблону. У поточній роботі розглядаються тільки питальні й наказові фрази користувача, тобто ті, які в явному вигляді пропонують системі дати відповідь. Передбачається, що розглянута діалогова або довідкова система використовує графову онтологічну базу даних, що безпосередньо впливає на формальні шаблони запитів – результуючі запити використовують SPARQL або Cypher. Приклади семантичного аналізу, розглянуті в цій роботі, відносяться переважно до мов флективного типу, а саме, української та російської, але основні принципи можуть бути придатними і для більшості інших мов.

Результати. Розроблений метод перетворення фрази на природній мови у формальний запит на SPARQL або CYPHER було програмно реалізовано для української та норвезької мов із використанням вузьких предметних онтологій та протестовано на відповідність формальним критеріям ефективності.

Висновки. Запропонований метод дозволяє діалоговій системі швидко та з мінімальною кількістю кроків вибрати найбільш підходящий шаблон запиту та витягти інформативні сутності із вхідної природномовної фрази, враховуючи величезну варіативність фраз у флективних мовах. Проведені експерименти показали високу точність та надійність побудованої системи та її потенціал для практичного використання та подальшого розвитку.

КЛЮЧОВІ СЛОВА: обробка природної мови, графова база даних, семантичний аналіз, формальний запит, дерево прийняття рішень, онтологія.

DOI 10.15588/1607-3274-2021-2-11

УДК 004.93

ОСНОВАННЫЙ НА ДЕРЕВЕ МЕТОД СЕМАНТИЧЕСКОГО АНАЛИЗА ФРАЗЫ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ ДЛЯ ПРЕОБРАЗОВАНИЯ ЕЕ В ФОРМАЛЬНЫЙ ЗАПРОС

Литвин А. А. – аспирантка, аспирантка отдела микропроцессорной техники, Институт кибернетики им. В.М. Глушкова, Киев, Украина.

Величко В. Ю. – канд. техн. наук, доцент, старший научный сотрудник отдела микропроцессорной техники, Институт кибернетики им. В.М. Глушкова, Киев, Украина.

Каверинский В. В. – канд. техн. наук, старший научный сотрудник отдела износо- и коррозионно-стойких порошковых конструкционных материалов, Институт проблем материаловедения им. И.М. Францевича, Киев, Украина.

АННОТАЦИЯ

Актуальность. Данная работа посвящена проблеме построения интерфейса на естественном языке для онтологических графовых баз данных. Основное внимание уделяется методам преобразования фраз на естественном языке в формальные запросы на языках запросов SPARQL и CYPHER.

Цель. Целями работы является создание метода семантического анализа типов входных естественноязыковых фраз и выделения из них значимых сущностей для инициализации переменных шаблона запроса, построение гибких шаблонов запросов для соответствующих семантических типов фраз, разработка программной реализации предложенного способа.

Метод. Разработан метод, основанный на дереве принятия решений, для определения семантического типа фразы пользователя и получения из нее набора понятий, для подстановки их в наиболее подходящие шаблоны формального запроса. Предлагаемая методика решает задачу определения типа фразы (непосредственно связано с критерием выбора шаблона формального запроса) и получения значимых понятий, для инициализации переменных выбранного шаблона. В текущей работе рассматриваются только вопросительные и повелительные фразы пользователя, то есть те, которые в явном виде предлагают системе дать ответ. Предполагается, что рассматриваемая диалоговая или справочная система использует графовую онтологическую базу данных, что непосредственно влияет на шаблоны формальных запросов – в получаемых в результате запросах используются SPARQL или Cypher. Примеры семантического анализа, рассмотренные в этой работе, относятся преимущественно к языкам флективного типа, а именно, украинскому и русскому, но основные принципы могут быть пригодными и для большинства других языков.

Результаты. Разработанный метод преобразования фразы на естественном языке в формальный запрос на SPARQL или CYPHER был программно реализован для украинского и норвежского языков с использованием узких предметных онтологий и протестирован на соответствие формальным критериям эффективности.

Выводы. Предложенный метод позволяет диалоговой системе быстро и с минимальным количеством шагов выбрать наиболее подходящий шаблон запроса и извлечь информативные сущности их исходной естественноязыковой фразы, учитывая огромную вариативность фраз в флективных языках. Проведенные эксперименты показали высокую точность и надежность разработанной системы и ее потенциал для практического использования и развития.

КЛЮЧЕВЫЕ СЛОВА: обработка естественного языка, графовая база данных, семантический анализ, формальный запрос, дерево принятия решений, онтология.

ЛІТЕРАТУРА / ЛИТЕРАТУРА

1. Galitsky B. Developing Enterprise Chatbots. Learning Linguistic Structures / B. Galitsky – Berlin : Springer, 2019. – 566 p.
2. Sun C. A Natural Language Interface for Querying Graph Databases: master’s thesis … master in computer science and engineering / C. Sun. – USA: Massachusetts Institute of Technology, 2018. – 69 p.
3. Formal-logical approach to building analysis systems of knowledge in different domains / [O. V. Palagin, S. L. Krivij, D. S. Bibikov, V. Ju. Velichko ta in.] // Problems in progtamming. – 2010. – № 2–3. – P. 382–389.
4. Li F. Understanding natural language queries over databases / F. Li, H.V. Jagadish // SIGMOD Record. – 2016. – Vol. 45. – P. 6–13.
5. Zhong V. Seq2sql: generating structured queries from natural language using reinforcement learning / V. Zhong, G. Xiong, R. Socher – 2017 [Electronic resource]. Access mode: https://arxiv.org/pdf/1709.00103.pdf
6. Transforming natural language query to SPARQL for semantic information retrieval / [S. Shaik, P. Kanakam, S. M. Hussain, D. Suryanarayana] // International Journal of Engineering Trends and Technology. – 2016. – № 7. – P. 347–350.
7. Ochieng P. PAROT: Translating natural language to SPARQL / P. Ochieng // Expert Systems with Applications. – 2020. – № 5. – P. 1–16.
8. Jung H. Automated conversion from natural language query to SPARQL query / H. Jung, W. Kim // Journal of Intelligent Information Systems. – 2020. – Vol. 55. – P. 501–520.
9. Yin X. Neural machine translation from natural language to SPARQL / X. Yin, D. Gromann, S. Rudolph // Future Generation Computer Systems. – 2021. – Vol. 117. – P. 510–519.
10. Damljanovic D. FREyA: an interactive way of querying linked data using natural language / D. Damljanovic, M. Agatonovic, H. Cunningham // The Semantic Web: ESWC 2011 Workshops. – 2011. – P. 125–138.
11. GIT-hub: FREyA documentation [Electronic resource]. Access mode: https://github.com/nmvijay/freya
12. GIT-hub Convert English sentences to Cypher queries documentation [Electronic resource]. Access mode: https://github.com/gsssrao/english2cypher
13. Litvin A. A. Method of information obtaining from ontology on the basis of a natural language phrase analysis / A. A. Litvin, V. Yu. Velychko, V. V. Kaverynskyi // Problems in programming. – 2020. – № 2–3. – P. 322–330.
14. Листи до Олеся Гончара / [рецензенти: С. С. Кіраль та ін.]. – Київ : Сакцент Плюс. Т. 1: 1946–1982 ; під загальн. ред. М. Степаненка. – 2016. – 736 с.

DOI 10.15588/1607-3274-2021-2-11