DEVELOPMENT OF NATURAL LANGUAGE DIALOGUE SOFTWARE SYSTEMS

Anna Litvin, Vitalii Velychko, Vladislav Kaverinsky

Abstract: *This paper discusses problems of natural language dialogue systems development. Here are considered different types of such program systems and their peculiarities based on our development experience. The systems discussed here are aimed at using graph ontological databases. Five basic types of natural language systems are highlighted: reference system, reference system with clarifying and technical questions, questionnaire system, active initiative system, control system. The first four of them are considered in detail. The reference system type, which is the basic type, is shown and discussed. Problems considered here are: natural language phrase to a formal query transformation, semantic analysis and answer formation. The ways are shown for these processes program implementation. For all the types the problem of the current dialogue subject keeping is considered, especially for an active initiative system where it is the most crucial. The different approaches are shown and compared for the principles of graph database (ontology) construction, which could be used in the dialogue systems.*

Keywords: natural language dialogue system, ontology, graph database, dialogue subject keeping.

ITHEA Keywords: *I.2.7 Natural Language Processing, E.2 Data Storage Representations*

DOI: https://doi.org/10.54521/ijita28-03-p03

Introduction

The user's interface in a dialogue form is convenient and user's friendly for many cases. So that it becomes rather widespread if a user is to obtain some information from the system (for instance, its database) or the system is to get

some information or instructions from a user. It is known that ordinary user is not quite familiar neither with programming nor formal query languages. Thus there are two basic options: developing the interface with previously prepared graphical control elements like buttons, menus, and so on, which are tied to the determined program procedures; creating a natural language interface, which does not exclude traditional control elements, but the main instructions a user tells the system in a natural language (English, Russian, Norwegian, etc.). The first option nowadays is still the most popular, because it is rather easy and understandable in its development and the functioning of such interface is very predictable. But if the system control becomes complicated its complicity is also growing. If it is a database interface it becomes very limited in possible options or needs to have long and complicated menus. If the system is designated for obtaining information from a person, its user becomes rather restricted in the answering options. Thus, for both mentioned dialogue systems a natural language interface will be more convenient and desirable.

Building of natural language interfaces assumes solving of some complicated problems, among which are the following: syntactic and semantic analysis of the natural language phrases for the intents included the correct interpretation; conversion of semantic information and obtained entities into formal queries and/or program instructions; presenting of the performed actions (like formal queries) results; keeping in the frame of current dialogue subject.

In this paper, we are trying to present and analyze our experience in building natural language dialogue software systems of different types and purposes. A lot of attention is given to the principles of ontology structure and its influence on the system development, semantic analysis, formal queries automatic construction, and presentation results of their execution. Also, a significant focus is made on the problem of conversation subject (context) keeping, its peculiarities, and importance for the different types of such systems.

Formulation of the problem

The main goal of the paper is to present and analyze our experience in the development of dialogue systems with a different destination. To consider the approaches to ontology constructing, semantic analysis of user's phrases,

formal queries automatic construction, and the interpretation of their results. To develop basic principles of dialogue context keeping through the conversation in different types of natural language dialogue systems.

A review of existing principles of natural language dialogue system development

Let us consider some known approaches to natural language interfaces and dialogue systems development, which include such problems as natural language text analysis and synthesis.

Syntactic and semantic text analysis is an important direction in automatic natural language processing and natural language understanding area. Different approaches have been developed for this purpose. They include ones based on machine learning and based on the system of rules (instructions).

The basic natural text analysis purposes are: named entity recognition, classification of text documents, finding answers in texts, determination of the intents told in the text. For such purposes as named entity recognition, documents classification there are several developed program implementations, for example, Algorithmia [Algorithmia, 2021] and Aylein [Aylein, 2021].

Another purpose of semantic text analysis is so-called "text mining" which is aimed at the discovery of previously unknown knowledge that can be found in text collections [Stsvrianou et al, 2007]. Text mining includes the following basic points: filling the database with text documents; finding existing information in the texts and storing it in the appropriate format; integration and querying of text data after it has been stored in databases; deduplication of datasets using standard data mining techniques, such as clustering. The natural language text mining process assumes the following issues: tokenization – separation of the test to phrases and words; determination and localization of the stop words; words stemming and lemmatization; clearing noisy data; determination and clarifying of words meaning; tagging – data part of speech characteristics; collocation – determination of compound terms and name groups; syntactic and grammatical analysis, finding of words dependency; determination of the importance and selection of the terms of the most representative ones; terms categorization.

There are several models of text presentation. Since the context of a term is a useful part of semantic information, in [Rajman et al, 1999] the contest was represented as a vector that contains the co-occurrence frequencies between a term and a predefined set of indexing features. Context templates have been used, for instance, in [Nenadic and Ananiadou, 2006] for the analysis of biomedical documents. These patterns are represented in a form of regular expressions and also contain part of speech tags and ontology information. N-grams are also could be useful for context determination [Stsvrianou et al, 2007]. They were used for text ordering, representation, and categorization [Caropreso at al, 2001]. They replace some unigrams with bigrams and use document frequency and information gain to reduce the number of n-grams extracted from the text. In [Cimiano, 2005] the context is modelled as a vector of syntactic dependencies found in texts. The concept hierarchy is extracted by applying a method based on the formal concept analysis. A linguistic parser extracts the syntactic dependencies, which are weighted and from which a lattice of formal concepts is constructed. In [Kehagias et al, 2001] sense-based representation is used. But the results of that research have not shown improvement in the accuracy of text semantic classification compared with word-based representation [Stsvrianou et al, 2007]. In [Carenini and Zwart, 2005] they attempt to match texts that describe product overviews with user-defined hierarchy. The advantage of such taxonomy is adding background user knowledge to the model and reducing redundancy. But its disadvantage is that for every domain a user-defined hierarchy is to be created. A matrix space model has been proposed in [Antonellis and Gallopoulos, 2006]. It is based on the idea that a document consists of hierarchically extracts – sections, paragraphs, sentences, and terms. So such matrices as term-by-section, term-by-paragraph, and term-by-sentence could be created. The advantage of this matrix representation is that it "remembers" the intermediate steps of the final matrix construction.

Another part of semantic analysis is categorization. It is needed to organize text documents into categories and to determine to which category the considered text belongs. This task could be supervised or unsupervised, dependent on whether the categories are previously known or not [Stsvrianou et al, 2007]. The

task and algorithms of categorization could vary depending on the purpose. The goal could be identification documents of the same topic; identification of the semantic orientation of the text; selection of papers by one author and even the distinction between interesting and uninteresting texts based on the person's preferences.

Many algorithms for categorization exist. In the case of thematic categorization, the focus is usually on noun terms that may characterize a topic. Also, machine learning approaches could be used [Yang and Liu, 1999], [Dumais et al, 1998], [Sebastiani, 2002]. A sentiment classification task deals with the texts classification according to the subjective opinion of the author [Jindal and Bing, 2006]. In this case, the focus is on finding the semantic orientation of words. In [Hatzivassiloglou and Mckeown, 1997] the main focus was on adjectives ant they study phrases where adjectives are connected with conjunctions such as "and" or "but". They use a log-linear regression model to clarify whether two adjectives have the same orientation. Then they divide the adjectives into two sets considering the set of the higher frequency to be a "positive" one. To discover the semantic orientation of words, they use an LSA-based measure to find out the statistical relation of a specific word towards a set of positive or negative words. Sentiment classification seems more difficult than the topic-based one and it cannot be based on just observing the presence of single words [Stsvrianou et al, 2007], and reliable methods for the subjective and objective author's opinions determination are still to be developed and employed.

The main purpose of the semantic analysis in the framework of the considered here problem is the determination of a semantic type of the user's input phrase, which is closely related to the intents that are assumed in it. So here we needed some kind of supervised semantic classification, i.e. assurgent the phrase to one (or might be several) of previously defined types. The final goal of this task is the selection of the most appropriate pattern for building a formal query. So let us consider some existing approaches for the conversion of a natural language phrase into a formal query to a graph database.

The LODQA system was presented in [Shaik et al, 2016]. It parses a natural language phrase and creates 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 terms 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. 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.

Another framework for natural language to SPARQL conversion is called PAROT [Ochieng, 2020]. It 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 a SPARQL query that fetches the answers from the underlying ontology. Testing the PAROT framework by the authors of [Ochieng, 2020] showed that for simple questions it demonstrates about 81-82% precision, about 43-56% for complex questions, and 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 recalls when processing aggregation-based questions.

For now, there are only a few examples of natural language to SPARQL conversion approach based on neural networks. One of them is described in [Yin et al, 2021]. Some tricks to avoid or at least minimize the 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 an instruction, by which it is possible to build a sufficient SPARQL query.

FREyA is an Interactive Natural Language Interface for querying ontologies [Damljanovic et al, 2011]. It uses syntactic parsing in combination with the

ontology-based lookup to interpret the question and involves the user if necessary. The user's choices are used for training the system to improve its performance over time. It deals with the English language. Query formation by FREyA very depends on the ontology data, and its configuration needs to be tuned to the certain ontology.

Conversion for natural language questions to Cypher is still not completely realized and widespread, but such a tool seems to be highly desirable. Here are a few examples of such works: [Sun, 2018], [CypherConverter]. The system proposed in [CypherConverter] is quite primitive. It needs a file with ready natural language sentences where some words are replaced by placeholders and matching Cypher templates. The main advantage of this approach is simplicity, which guarantees that the result will be just obtained or not obtained without appearing in strange situations when wrong or not completely correct result springs up. But it has many disadvantages, first of all, for a real big system, a large number of phrase 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 [Litvin et al, 2020].

Synthesis methods of natural language texts are an important component in constructing the human/computer interface, which is based on a linguistic processor – a component that implements a formal linguistic model and can work with natural language [Sudakov and Malyokin, 2012]. The main functions of a linguistic processor are understanding modelling (analysis) and text production modelling (synthesis). Applications that generate natural language texts work with language information as a string of characters. They manipulate sentences and phrases as building blocks of a future text. Such technologies are simple, reliable, and therefore are widely used.

The template text synthesis system uses ready-made text fragments and combines them to occupy the given positions in the stereotyped text. The simplest realization is just to insert text fragments into templates. More complex

template systems can perform linguistic processing, such as using some grammatical parameters of the text or combining template sentences into a binding text using certain rules [Sudakov and Malyokin, 2012].

The work [Bolshakova, 2014] considers the language of lexical-syntactic templates LSPL, which is a declarative language for the specification of the lexical and grammatical properties of structures allocated in Russian texts to automate many tasks in processing scientific and technical texts. The preparation and configuration of the templates are carried out by a linguist or a specialist in the analyzed texts subject area, without programmer participation. The language of lexical-syntactic templates LSPL is a declarative language with built-in tools for specifying the lexical and syntactic properties of structures in Russian-language texts. The elements of the template can be the words and phrases, as well as grammatical conditions. For word elements, in the general case, a part of speech and a token is indicated; morphological characteristics (case, gender, number, etc.) are specified. The matching conditions determine the equality of either specific morphological characteristics or all common morphological characters of the words being coordinated. In LSPL templates, you can specify repetitions of elements, optional elements, as well as alternative language constructs. Already defined patterns can be used to specify patterns of more complex natural language expressions, using their parameters to specify or coordinate elements of the described construction.

Thus, we see that many approaches, methods, and tools have been developed to build a natural language dialogue system. Such systems are being developed and there is great progress observed in them. Most of them can be divided into three groups: based on text search, neural networks, and database interfaces. It should be noted that dialogue systems based on neural networks and using deep learning have recently become popular [Tarasov, 2015]. Such bots can make a good first impression. But a detailed analysis highlights some of their shortcomings. First of all, they have a slow learning rate. It needs a lot of time and large amounts of data to train such a system. An interesting feature of neural network-based dialogue systems is that they have difficulty simply saying "I don't know", however, this disadvantage in some cases could be inherent also to the other types. The third drawback is difficult results obtaining reasons

interpretation: how it being guided and why this and not that answer was given. And, therefore, it is difficult to verify the adequacy of such an answer [Levin, 2016].

An example of a well-known modern dialogue system could be the Mitsuku chatbot developed by Steve Worswick [Kuki]. For its development AIML (Artificial Intelligence Markup Language) [AIML] language was used. It is considered one of the best natural language dialogue systems and is a 4-time Lobner Prize winner. However, Mitsuku testing has revealed an important disadvantage: it does not well keep in the context of the dialogue, it simply forgets what has just been discussed, and does not distinguish between really interrogative and imperative phrases and answers to the questions given in a matter similar to such phrases types.

A good alternative to dialogue systems based on neural networks could be ones using ontology in a form of the graph database, which is considered in this paper. As we can see from the review above, there is a rather big experience in the word on building natural language interfaces and needed for them grammatical, lexical, and semantic analysis methods. However, the existing approaches still have the potential to be further developed and new ones are also ought to be tried to create and tested. In this work, we make an effort to consider and analyze our experience in natural language dialogue system creation and to figure out the perspectives of further development of the proposed approaches.

Dialogue system types

A dialogue software system is a form of a program interface designed to interact with a user imitating a conversation. The type of such interface, considered in this work, implements the communication between a user and program in natural language. The user is able, depending on the purpose and implementation of a particular system, perform one or several of the following actions: to ask a question, to ask (to command) the program to perform some action, answer a question issued by the system, express a narrative phrase that is not an explicit question or imperative, but assumed a response from the

program (comment, advice or action according to the described situation). The system can perform the following actions: to give an answer to a question, to ask a question, to perform some actions (if they are possible and provided by this system), to give some comment or narration. In a certain software system, the above types of actions can be implemented only partially and selectively, depending on its purpose. Natural language interaction between the program and the user at a given technological level can be realized both in written and oral forms. Within this work is only considered the written interaction type, because oral interaction can be reduced to a written one using software solutions that implement speech recognition and synthesis. Significant success has been achieved so far in the development of such transformation processes, and their further development continues [Shaik et al, 2016], [Ochieng, 2020], [Yin et al, 2021]. However, these methods are separate areas of knowledge and are not considered here. The variants of dialogue systems that work with the written language could, if desired, be adapted to work with an oral speech by adding to them modules of recognition (speech into text) and synthesis (generation of a sound from a text).

Depending on the functionality implemented in a particular dialogue system, both from the user and program sides, various options are possible. Each of them can have its application's area and features of the tasks to be solved during its development. Here are some of the typical options:

1) A user can ask questions, as well as enter imperative phrases (requests) with the meaning of information request, but neither more. The system only answers the user's questions, but does not ask questions, does not provide any information without the user's request, does not launch any processes that are not related to the search and giving of information. This version can be called a "question/answer" reference system, a natural language interface to a database or their combination.

2) A user can ask questions end enter imperative phrases that have a piece of information requesting purpose. The system not only gives answers to the

user's questions but also asks questions but is limited. The essence of the limitation consists in the following: questions asked by the program have rather a technical goal, such as the user's question semantics clarification, requesting of the concepts subject area. Questions of a purely technical nature are also possible, such as "Do you want to complete the dialogue?", "Are you satisfied with the received answer?" etc. This is, in fact, also a help or reference system, but supplemented by the possibility of refinement to increase the relevance of the answers.

3) A system that asks questions to the user. Responses to a user interface are either not implied at all or are purely formal and followed by a question. It may also be possible to receive the resulting response of the system after a questions series. The system's response will be based on the collection of responses given by the user. The user, in turn, can only answer the questions posed by the system, which, within the framework of the considered paradigm, will occur in natural language. Such a type of dialogue system implementation can be acceptable for automatic questionnaires, testing, or diagnostics systems.

4) The user can both ask questions to the system and answer questions coming from it. The types of actions that the system can perform are similar. Also, the additional types of actions in this variant are input and output of declarative phrases without interrogative or imperative meaning, which, in turn, can be commented, ignored, or they can be followed by a question, request, or a narrative phrase. This is a "virtual companion" system type, a typical conversational natural language chatbot.

5) The user gives instructions to the system in the form of imperative statements aimed at starting or stopping some action. Also, optionally, the user can ask questions, but only about the status of the running process execution or about the action completion. Based on the instructions of the user, depending on its purpose, the system performs some action or starts a process. This can be either a purely software process on a computer, or actions performed on surrounding objects ("turn on/off a device", "change the sound volume","

Start/stop a technological process ", etc.). An action is performed if it is technically possible and provided in this system. Also, such a system can optionally give natural language responses related to the fact of the start/end of the action, the process status, the results of the execution, the ability to perform the requested action, etc. This is, in fact, a control system interface. This type is not considered within the framework of the current work, since it is, in fact, a variety of types 1 and 2, supplemented by starting a process and controlling it. The launch of processes on a computer by a program and their control belongs to system programming, and the control of objects of the material world belongs to automation and robotics, which is beyond the scope of our research.

Here will be considered the first 4 of the listed options and the problems related to their implementation features, as well as technical solutions proposed for such software systems implementation.

The question/answer reference system is the most technically simple and could be considered as the basic version of a dialogue system. It is essentially a natural language database interface.

Graph databases

Within this work will be considered graph databases of the ontological type. Such databases have several advantages for interactive systems building. So, a graph database, in the general case, is easily expandable, new nodes and links can be added to it, including their new types, and it could be performed in a fairly arbitrary order. Unlike a relational database, a graph database does not require a pre-designed schema of related tables and their field types. It is devoid of some typical problems which could arise when deleting and adding new fields, especially foreign keys, such as ambiguity of default values, loss of a foreign key reference object, etc. A graph database is convenient for representing objects collections that have a natural graph structure. In particular, it could be suitable for ontologies implementation, which are representations of objects and phenomena of a certain subject area and connections and interactions between them. One of the ways to implement a graph database could be an RDF triples storage, which is built on the principle

of object – predicate – subject atomic structures. Any related structure can be represented in this way [Gomez-Perez et al, 2002]. To implement queries to such databases SPARQL [Antoniou, 2016] language is traditionally used. SPARQL queries are mainly considered in this work owing to their popularity and widespread use for such databases. There are several formats for storing information in the form of RDF triples. We, in particular, used RDF/XML. The de facto RDF/XML variant is OWL. The main difference is that RDF assumes a fairly arbitrary system of predicates and node types, but the OWL standard imposes restrictions on them. In particular, in OWL, data (graph vertices) can be a class, a property or an individual, possible relationships (and therefore predicates) are as follows: a subclass of the class, an instance of the class, a sub-property of the property, a domain of the property, a range of the property (domain and range refer the property to classes), equivalence, difference. OWL also allows storage of some additional information, such as Label, Comment, Language, and some others. In common the types of additional information bound to a class, property, or instance (Annotations) could be quite arbitrary. It should be noted, that the OWL format is converted into a set of RDF triples, where different types of “Annotations” form the corresponding predicates. Thus, OWL and RDF/XML are largely interchangeable. Another option for implementing a graph database, which was used in our projects is Neo4j DBMS which uses the Cypher query language. It is a high-performance, scalable open-source graph database management system [Goel, 2015]. At the moment (2021) it is the most popular and widespread graph database management system, significantly ahead of such competitors as ArganoDB, Virtuoso, AllegroGraph, OrientDB, etc. [DBEngines, 2021]. It allows one to apply additional optimization for data with a complex structure, the graph processing does not require placing it entirely in the RAM, thus processing very large graphs is possible [Goel, 2015].

Reference dialogue system. Basis principles of formal queries construction on natural language phrases.

The general scheme of the dialogue system of the "question/answer" type is as follows. User messages are analyzed for meaningful entities extraction.

Semantic analysis of the user's phrase can also be carried out. Its result is a certain semantic type of the input phrase, which can then be used for the following purposes: determination of a scheme for significant entities extraction, selection of a formal request constructing paradigm, selection of an ontology the formed formal query will be directed to, determination of a form the answer to be presented to the user. Then, using the obtained data, a formal query to the database or a set of them is to be built. The received formal request or the set of requests is sent for execution to a service that interacts with the DBMS. This service can also perform some additional queries to the database. This can be done for the following purposes: queries with entities that have undergone reduction if the query results are not received (in this way more general concepts or closely related concepts are requested, which increases the probability of an answer receiving, although perhaps less relevant); request for additional technical information that can be used, for example, for ranking the responses (such information, for example, can be the repetition of the requested concepts in the response text if the response is a certain piece of text); obtaining additional information related to the requested information, which may also be of interest to the user (in this case, for example, a section "see also on the topic" or "perhaps you will be interested" may be added to the answer, containing in a collapsed form related information items obtained in this way).

Further actions will be related to the nature of the responses received from this database. The most primitive option is simply to return the resulting set of results to the user as a response. But this is not always acceptable. So the result of a query can be merely a link or a set of links to response texts, which are obtained from another (document-oriented) database. Also, the answer can contain media content – images, video, and sound inserts, as well as external links substituted from other sources. In this case, additional processing of the response text should be carried out to substitute the corresponding materials indicated in it or links to them in the appropriate positions. Also is not excluded, the case of multiple responses received as a result of the request. In this case,

further actions depend on the expected semantics. So, in many situations, the user's question suggests a listing of certain entities as an answer, such as "What journals exist on the problem N.?" or "From what writers did N. receive letters in 1968?" In this situation, an enumeration is formed from the list of obtained results, which is substituted in the response. A similar situation is when it is supposed response formation from several parts. For example, a request returns, as a result, a certain set of entities (title and body of the text; event description, date, and place, etc.). A completely different situation is when the result is a ready response text (definition of a concept, detailed explanation, instructions, etc.). In this case relevance ranking of the received responses is needed. The ranking method by calculating the corresponding metric is described in our work [Litvin et al, 2020]. Also, additional answers, received by additional queries execution, if any, are also subjected to ranking. The ranking process could occur even at the stage of the queries forming and executing. So, a query made using more reduced entities will have a lower rating. It is also possible that several types of queries are assumed. They are ranked according to a decrease in probable relevance. The next one is to be executed if there are no results for the previous one, or just to obtain additional related information. For example, the question looks like "What is N.?" Several types of formal requests are provided for it: a request for the definition of the concept N.; a request for clarification on the concept N.; a request for the class to which the concept N belongs; a request for entities related to the concept N. in any way. It is obvious that the potential relevance in these semantic types options associated with this question, respectively, decreases.

After receiving answers to formal requests and ranking them (if necessary), the next step is to provide this information to the user. One of the possible operations has already been named, that is the substitution of media content if any. If the complete ready response text is received in this way, then it is simply transferred to the user interface, additional information is also transferred there, but for displaying in a collapsed form that could be expanded if desired. However, there are situations when, the obtained result is not a complete text,

but only a set of entities. Providing such an answer directly to users is possible, but it will look sloppy and not friendly enough. Therefore, a more acceptable option in this situation is to generate a phrase for the answer. For this, the following data are used: the semantic type of the user's question, or rather the type of formal request based on it; words from the user's original phrase; predefined phrase elements (selected based on semantic type); the results of a formal query themselves. More details about the method of natural language answers generating from program responses using flexible templates are described in our work [Velychko, 2020].

Let us consider in more detail such components of the above process as semantic analysis of the original phrase and the formation of a formal request since these stages play a key role in it and determine the relevance of the response.

The database of the system could be a single ontology (one graph) or a set of several ones. The second type is suitable when there are some different subjects or topics the system is devoted to. Few smaller graphs in this situation are more acceptable because operation on them could be performed faster and the probability of information adulteration from outlying subjects is significantly less. But if the database has several graphs there appears a problem of the most suitable one selection to perform the certain query. In our dialogue systems, it is solved in a rather simple way. Each of the ontologies has a set of keywords bound to it which represent the concepts and terms included in it. These lists are quite long – from tens to hundreds of words, depending on the graph size. These keywords lists are to be compared with the list of entities selected from the input question for the formal query formation. Each of these terms is to be checked whether it is present in the keywords list. That ontology is to be chosen for which its keywords list contains most of the input concepts. If this number becomes equal for several ontologies it should be chosen one which keywords list contains fewer other concepts, i.e. is shorter. It decreases the probability of search noise appearance. There also could be a previous stage where a certain group of ontologies is selected. For this purpose are used words directly from the input phrase. The presence or not the presence of some

words there leaves only a few of the ontologies for the following selection by the described above method. This technique was practically tested in our reference system devoted to the oeuvre of Ukrainian writer Oles Honchar and showed acceptable results. In that case, the keywords lists were automatically created when unmanned ontological graph creation. Preliminary ontology group selection was manually created and in the case divided the input questions into two groups: about O. Honchar's works and letters to O. Honchar. This previous stage was needed here also because there were two different types of ontologies structure, which were needed different query structures.

Except for the selection of the most relevant graph, a very important part of the semantic analysis is the determination of question semantic type. However, it is needed not in all the system paradigms. We have built and tested several dialogue systems that are used only one universal query template for every case. Let us describe this method, which has both its advantages and drawbacks. The first thing is the ontological graph structure that was described in our work [Litvin et al, 2020]. Briefly, exposition is as follows: the graph contains concepts atomized to words classified to several groups according to pars of speech; on lower hierarchy levels (subclasses) there are name groups that extend the parent concept, there could be any number of hierarchy levels on each next concepts contain one more extra word; there are links to ready answers in the ontology, which are descendants of the corresponding entity "Ready answer"; there is a group of properties that inherits "Main property", which binds the sets (intersections) of the present in the ontology concepts to the corresponding link to a ready answer, where the domain is the set of concepts and range is a link to an answer text. Also in the graph, some additional technical information is stored, for instance, data on the repetition of some concepts in the certain answers texts, which is used as one of the factors for answers ranking. From the input user's phases, the program extracts all the meaningful concepts except auxiliary words. All of the entities in the lemmatized form are used for substitution into the query template. Then the extracted set of words undergoes reduction. During it, the words are removed one by one so each following query is aimed at a wider concept than the query with the initial set of words. At this stage occurs the problem of the reduction order, i.e.

removing which terms make correspondently more or less meaning loss and possible misinterpretation. Also, here appears another problem: if the initial phrase and so extracted set of words are rather long there will be a big number of the reduction options. Processing of all the possible options could become quite time and recourse-consuming, which is not desirable for dialogue systems operating in a real-time regime. Our projects we basing on the practical experience and possible semantic value of different word types have proposed the following reduction scheme. Several levels of reduction are performed each next of them is assumed as possibly less relevant: 1 – removing of verbs only; 2 – removing of numbers only; 3 – removing of the question, negotiation, and binding words only; 4 – removing of verbs and question words; 4 – removing verbs, numbers, question, negotiation, and binding words; 5 – removing of adjectives; 6 – removing of adjectives, question, negotiation, and binding words; 7 – removing of nouns and adjectives; 8 – only verbs and numbers remain; 9 – only verbs remain. Then it performs a series of random reductions merely by just phrase shortening doesn't matter by what part of speech. This stage could be as long as possible, the reduction of user's phrases usually is not a very consuming process but the performing of the queries is. So in practice not all queries formed using these options of reductions often need to be performed. If just one answer is needed the process may be stopped after the first not empty query response obtaining. If some additional information is to be obtained there will be enough to perform that number of queries which gives you the desired number of responses not equivalent to each other. This uncomplicated approach was implemented and tested by us for several dialogue reference system and have been shoving rather acceptable results: it extracted answers hooked on at least a few related concepts and also pulled more or less related information (both after a single query and their series). Moreover, the obtained in such way responses (excluding long responses sets returned by a single query performance) appear prearranged. After subsequent metric ranking [Litvin et al, 2020] the most relevant of the responses occur on the top.

The advantages of such an approach are following: simplicity – deep semantic analysis and many query templates are not needed; universality – it needs a rather simple ontology structure that building from natural language texts could

be quite easily automated, it does not very dependent on the language structure and can pull answers (more or less relevant) in the most of cases related to the ontology subject. However, it also has some strong disadvantages, among them are the following: it is aimed at a certain simple ontology structure, so it is unable to deal with complicated semantic graphs of advanced structure created by separated persons and teems based on their subject vision and not familiar with the described above approach; it is devoid of semantic focus, and this leads to some drawbacks – sometimes rather long query set is needed to obtain at least some information, which is consuming, there is no guarantee of semantic shades relevance, especially using reduced concepts sets. So that, to operate with more complicated semantic graphs a more advanced approach was developed. Here we give a brief description of it.

It is assumed that terms in graph nodes are not atomized to single words and short name groups but could be rather deployed concepts containing up to tens of words. There are possible many types of relationships between the graph nodes which correspond to semantic types. The semantic relationships types could be general or specific to the subject, which may simplify the ontology and queries structure. Such graphs could be stored in RDF/XML or Neo4j formats. Restriction to using only standard OWL predicates and types leads to unnecessarily complicating both ontologies and queries. Let us consider an example: we are to express a simple statement “the antifriction bushing is made of tin bronze" in a formal way – as a part of an ontological graph. If we want to use the standard OWL methodology only, one of the manners will be as follows. There are two classes "anti-friction bushing" and "tin bronze", they might be descendants of higher hierarchy classes, for instance, "bushing" and "bronze", correspondently, which are also may be descendants of higher ones "machinery" and "material", but for the current purpose in doesn't matter. The main goal considered here is semantic binding, which is not hierarchical. So in OWL, we need a property, it may be something like "material" or "is made of" (to avoid mismatching with the corresponding class name). This property itself is abstract one or binds only higher hierarchy classes like "machinery" and "material" in this case. For more certain entities binding, it should have descendants; each of them binds a certain product to its material or materials (if

it is made of several ones). Here this property may be something like "anti-friction bushing material". The domain of this property will be "anti-friction bushing" and its range "tin bronze" that are linked to the correspondent classes. Using RDF/XML the construction can be simpler. The hierarchy of the classes remains the same, but it is not the subject. We merely need one arbitrary predicate "material" or "is made of". So we have ready RDF triple: object – "anti-friction bushing", predicate – "material", subject – "tin bronze". The same construction appears using Neo4j. In this case, we have two nodes "anti-friction bushing" and "tin bronze" and a directed relationship between them of type "material". The direction of the relationship comes from the node "anti-friction bushing" to "tin bronze". And now let us compare formal SPARQL (or Cypher) queries for the considered cases. Perhaps the goal of the query is to return the machinery parts made of tin bronze. For the described above OWL structure the SPARQL query will be:

```
SELECT DISTINCT ?result WHERE {
        ?propName rsdfs:subPropertyOf :IsMadeOf.
        ?propName rdfs:range :TinBronze.
        ?propName rdfs:domain ?className.
        ?className rdfs:label ?result.
        ?className rsdfs:subClassOf ?intermediateClass.
        ?intermediateClass rsdfs:subClassOf :Machinery.
}
```

In the example of the query above we need a variable for the property which range we know but are to ask about its domain. And formally this property must be a descendant of the "IsMadeOf" property, we do not need other relationships here. Also the class we request here, as a result, must be a descendant of something that is a descendant of the "Machinery" class. For RDF/XML with arbitrary predicates a SPARQL query with the same goal will be:

```
SELECT DISTINCT ?result WHERE {
        ?className rdfs:IsMadeOf :TinBronze.
        ?className rdfs:label ?result.
        ?className rsdfs:BelongTo :Machinery.
}
```

Here we have arbitrary predicates "IsMadeOf" with the meaning of the material semantic reference and "BelongTo" which describes belonging to something regardless of hierarchy. This seemingly not very significant simplification may have a crucial role for more complicated queries, where it might allow one to avoid intricate links chains. Here is another example, where we just want to request the years when letters have been written by some person called Sergiy Oleksiyovych Zavgorodniy. You can see how complicated the query becomes using OWL ontology:

```
SELECT DISTINCT ?result WHERE {
        ?p rdfs:range : Sergiy.
        ?p rdfs:range : Oleksiyovych.
        ?p rdfs:range : Zavgorodniy.
        ?p rsdfs:subPropertyOf :FullName.
        ?p rdfs:domain ?y.
        ?y rdfs:subClassOf :Person.
        ?x rdfs:domain ?y.
        ?x rdfs:range ?t.
        ?x rsdfs:subPropertyOf :Authorship.
        ?t rdfs:subClassOf :TextLink.
        ?d rdfs:domain ?t.
        ?d rdfs:range ?r.
        ?d rsdfs:subPropertyOf :SendingDate.
        ?r rdfs:subClassOf :Date.
        ?yd rdfs:domain ?year.
        ?yd rdfs:range ?r.
        ?year rdfs:subClassOf :Year.
        ?year rdfs:label ?result.
    } ORDER BY ?year
```

Here a chain of several properties is used to build the linkway from the parts of the person's full name to the year, which is part of a date, which is bound to a letter text, which is bound to a person by "authorship" property, and each person is bind to the parts of his/her name. This is a real example of a SPARQL

query that could be automatically constructed by the template in one of our dialogue reference systems.

If in the ontology was built on somewhat other paradigm using direct arbitrary predicates and long deployed class names the query with the same purpose would have the following look:

```
SELECT DISTINCT ?result WHERE {
        ?text rdfs:subClassOf :TextLink.
        ?text rdfs:Authorship Sergiy_Oleksiyovych_Zavgorodniy.
        ?text rdfs:SendingDate ?date.
        ?date rdfs:YearOfDate ?year.
        ?year rdfs:label ?result.
} ORDER BY ?year
```

Here we can see a significant simplification and simpler queries are easy to construct and faster to perform.

If Neo4j is used as the DBMS the queries in Cypher for the considered above two examples will be as follows:

```
MATCH
        (n:MachineryPart)-[:MadeOf]-(m:Material)
WHERE
        m.name = "Tin bronze"
RETURN n.label as result;


MATCH
        (n:Person)-[:Authorship]-(t:TextLink)-[:SendingDate]-(d:Date)
WHERE
        n.name = "Sergiy_Oleksiyovych_Zavgorodniy"
RETURN d.year as result;
```

So we can see that Cypher queries with the same purpose could be very simple and understandable also owing to the possibilities Neo4j provides in the graph building. It should be noticed that more complicated queries are also possible in Cypher either as graph algorithms, which made its usage preferable for most

cases assuming large graphs with a complicated structure that also need queries with complex and multiplied semantics.

As has been mentioned, usage of direct arbitrary predicates with meanings of semantic relationships and deployed entities for nodes makes the ontology more semantically structured and simplifies queries simultaneously making them more semantically aimed. But in terms of dialogue systems development arises the problem of the relation between concepts extracted from the user's phrase and ones to be substituted to a formal query. And differently from the single query template version here appear several (maybe tens and up to hundreds) semantic types and templates for them. So another important problem here is the appropriate template selection, which needs a semantic analysis. But at first, let us consider the issue of translation concepts extracted from the user's phrase to the terms that are stored in the ontology.

In the case of ontology where the terms are rather atomized it for most cases will be enough to reduce the terms list originally extracted from the user's phrase. And also, looking ahead, the extraction of terms, in that case, was quite simple – it was enough to get out of less meaningful words, here terms are to be extracted from certain places only. One of the versions of reducing highly probably becomes appropriate at least one of the terms bind with an answer. Nevertheless, this approach has its disadvantages named above. If in the ontology are long terms of many words, which are also placed in a certain order there is a rather low probability for them to be precisely the same in a phrase the user wrote. And reduction does not help here. First – it does not assume the order of words (and enumeration of reshuffle options is a consuming process), second – it that method is helpless for the terms that have some additional words (the context expanding is generally an ambiguous problem). So in this case it is needed to extend or reduce the context of the obtained concepts to the most suitable term stored in the ontology. Sometimes some words obtained from the user's phrase are to be omitted but others are to be added to fit the concept to one stored in the ontology. To solve this problem we propose the following approach. There is previously prepared a file where all the terms from the ontology are stored in the way they are presented there. To each of them is binding a list of words from the concept. The words are stored in lemmatized

form, and to each of them its part of speech is given (this revokes its determination during the program performance, which makes the working process quicker). The words set corresponding to the extracted term are compared with these words. Both additional and missing words are taken into account. To estimate the similarity degree of the input words set and one from the file is used the metric similar to that one from [Litvin et al, 2020]. The metric gives different scores to matching, additional, and missed words depending on their part of speech. To make the process some faster concepts which have the number of words less than in the most fitting for the current moment are not analyzed. So the most fitting term is to be substituted into the formed query, and its presence in the ontology is guaranteed.

Another important task here is terms extraction. In our approach, this process is strongly combined with the determination of the phrase semantic type. To determine the semantic type a tree-based method is proposed. It is based on the following facts: even in inflective languages words in a meaningful sentence have a defined order; there are words, among which are question words, prepositions, verbs, and others, that are mostly responsible for the semantic type. In the method, a frame is moved through the phrase. The size of the frame is flexible and depends on the number of words under consideration. The coincidence of the words in the frame determines the current condition, which excludes some of the possible semantic types or proofs of the considered ones. More detailed this method is described in [Litvin, 2021]. At the end of the tree, one type remains, but there could be also several ones hierarchically ordered, as mentioned above.

The tree for the semantic type determination is designed in that way which also provides determination of the position from where the concepts substituted into the query are taken. These concepts are obtained after the semantic type is determined and are represented by a set of words, which is transformed into the ontology concept in a way described above.

For a reference dialogue system, where a user just asks a question, the problem of the context keeping is not particular. Moreover, it is condemned to flexibly switching its topic according to the incoming user's queries. However, sometimes if the system is to be users friendly, some features should be taken

into account. One of them is pronouns substitution. In some cases, it could be easily done by analyzing such characteristics of the pronoun as its gender and number. But this does not always work, because there may be ambiguities in such comparing. The ambiguities could be different depending on the language. For example in the languages where inanimate entities don't have a gender distinction, like in English, using such pronouns as "he" or "she" defiantly points to an animate entity (often a person) mentioned before, but "it" is defiantly not for persons. However, it makes harder inanimate entities distinction – they all are "it". And in English, there is no gender distinction for demonstrative pronouns. In many other European languages, there is gender distinction for nouns for inanimate entities. So it could be easier to guess what the pronoun points even for inanimate entities. But this does not completely solve the problem, because there could be previously mentioned several entities with the same grammatical gender and number. One of the ways here is the usage of heuristics trying to guess the pointed entity by the probability of having characteristics or making actions (for instance, "cars do not wear hats", "cats normally could not be green", "trees can not be students" etc). Such an approach will work for most cases, except very specific and absurdum ones, which are rather confusable even for humans. But the mentioned approach needs a lot of affording for its complete realization. There should be another additional database, probably of ontological type, where at least possible entities combinations are listed. Listing of the impossible ones is not needed for two main reasons: if the combination is not given as possible it is treated like impossible or at least unlikely; there could be a great number of impossible and odd compositions, which will make the ontology very resource consuming and slow working. An easier way that does not assume having of an additional ontology is questioning the user about what he means if the confusion in the pronoun substitution appears. So we go to the second type of dialogue system that assumes asking some questions to the user.

This type of dialogue system asks the user questions only if there is a technical need. One of them is mentioned above: pronouns substitution. There could be also other cases: the system "believes" that the question is exhausted, so it asks a question whether the user has another question of the conversation is

over; the user initiates the dialogue end, so the system interests, if the user receives the right answers or not (these data could be gathered with the corresponding dialogues histories and further using for the program and the database improving); informing of the user abut that the question is not on the topic, which is another part of the context keeping.

The problem of context keeping is not very crucial for the systems with a narrow subject ontology. There all the questions far from the subject will result in an empty ontology response, which leads to the warning of the user in the dialogue that the question is out of the subject the system condemned to. If the system is multi-subjected and probably has several ontologies, for a reference system it rather will be a better way to flexible switch from one ontology to another than trying to keep the user near a single topic, which could be even disgusting for the person. Imagine a situation, when the user just wants to ask about other things he is also interested in, but the system refuses to answer and instead proposes not to change the subject of conversation.

Questionnaire dialogue system

The third type of dialogue system is rather particular. It is condemned to ask questions rather than answer them. Such systems are also very useful in some situations, among which are: interactive fulfilment of a questionnaire, which in this way could be flexible; testing systems that, for example, check a student the learned material comprehension; interactive diagnostic systems that depending on the user's answers to certain questions determines whether something is going wrong and what exactly does.

In the simplest type of such system, there is only a fixed set of questions to be asked to the user. The only distinction between it and the ordinary computer-based test with automatic checking is the answering of the user in a natural language. Here not the questions but the answers of the user undergo a semantic analysis and meaningful entities extraction. So the user's phrases are treated as narration ones rather than interrogative. The following scheme is mostly the same. Depending on the semantic type and extracted concepts a formal query or set of them is constructed. Its results are stored, may be further used.

There could be a more complicated system type where the following question of the system depends on the previous ones. It is a nonlinear questionnaire dialogue system. There could be several realization options. One of them is a testing system with questions about adaptation. It determines on what questions the user gives correct answers and for what ones the answers are wrong. So, depending on the purpose, the following questions will be rather from the topics the user is familiar with (more correct answers previously given) or contrary to the subjects the user is unversed in. Another version is an adaptation in objective hardeners of the questions: if the user is giving correct answers to the easier questions the following will be harder, and vice versa. Another nonlinear dialogue sequence in such type systems could be the following: there is a decision tree at the base of the dialogue control. The conditions for the bifurcation points of this there are determined by the answers of the user. So the following question shall be different depending on the result the program obtained using the material of the last user's phrase.

Finally, the system may give an answer, which is the test or diagnostic result. If the system is just a test this final result may be, for instance, the score of correct and incorrect answers with the information if the test is passed and on what mark. If the dialogue system is a source for a decision tree bifurcation conditions this resulting answer is determined to be the final vertex (node) of the tree, which links to the diagnosis. If the system's destination is merely a questionnaire that is designed for database fulfilment based on users answers the resulting answer may not exist at all or be just formal, like "Thank you for the information. We have taken pleasure talking with you" or "OK. The data are successfully saved".

The problem of dialogue subject keeping is the least for this type because the system enforces the subject by itself through asking questions that are not random, but are determined by the program depending on certain conditions.

An active initiative dialogue system

The fourth type of dialogue system is the most complicated. It assumes the active participation of the program in the dialogue process, i. e. it should ask questions and answer incoming ones, tell some facts to the user that were not

asked previously, tell some comments related to not interrogative but narrative phrases of the user, and sometimes to use them for its questions asking. Based on such a system could be a combination of types 2 and 3, but just a combination is not enough. The system should flexibly, automatically switch from one type to another, and do in the appropriate moment. There could be two general schemas of the dialogue unfolding depending on which of the sides initiates the conversation topic by first meaningful phrase (not a formal greeting). Let us consider the case when the user enters such a replica. It could be either: question, impetration, or narration. Consider separately each of these three options.

If the user's initial phrase is a question, it assumes a system should try to answer it. This could be performed like in a reference type of dialogue system (types 1 and 2). Some answers could be obtained anyway: informative answer, warning about unknowing the appropriate answer, finally obtained answer after a clarifying question (type 2). If the system will restrict itself for cases of user's interrogative phrase to only an answer returning it could degenerate to the simple reference system. So the initiative of the program should not be suppressed by a user. The first thing the system can do in this case (as well as in the others next considered) is to determine the general topic of the first phrase. It could be done by the selected ontology or in a similar way. If a subsequent user's phrase will be determined to belong to the other topic it will lead to asking the question about topic change and its reason. It is already a partial interception of the initiative by the system. It makes the user answer the system question including unswerving about the reasons for the topic change. But the most efficient method of interception of the initiative to the system side is asking a question to the user directly after the just given answer without waiting for the user to make the next dialogue act. It could be not a question but a narration, however, a question is a better option, because a not completely closed to the question narration following the answer could be perceived as a rather strange part of the answer. However such phrases, if they are sometimes to narrations, could have a specific beginning such as "Do you know that there also exist such things as…" that can separate this phrase from the previously

going answer. Such acts may go not after every answer but appear from time to time. To keep the conversation subject these questions initiated by the program could be generated randomly but related to the main topic. If a narration type is to be chosen for such phrase it could randomly be selected from the related information obtained with the main answer in a manner described above.

If the initial phrase of the user is a narration situation is easier for the program. The system answer for this case could be other related narration or a question. Imperative phrases of the user are treated in the same way as questions.

Another case is when the program initiates the first meaningful phrase of the dialogue. It could be either question or narration. If it is a question, it is assumed that the user will answer it. How to use the user's answer and whether it is useful at all depends on the system destination and the current question. If, for instance, the answer was about the user's name, this name is probably could be extracted from the answer. In the following phrases, the program can use this name when contacting. After the user answers, the system is quite free about the type selection of the subsequent phrase. If the system starts dialogue from a narration it could expect from the user all the options of phrase type. Subsequent behaviour is determined by the user's phrase almost as it would be at the beginning.

The problem of the dialogue subject keeping for this type of system is the most crucial compared with the others. It could be divided into two parts: long-term topic keeping and short-term one. The problem long term being in a certain subject is that what has been mentioned above and is not hard to be solved but either is not very crucial. The possible solution is the division of the ontology on the thematic parts and phrases initiated by the system should be inside the parts which are the last involved. And there might not be a practical reason to force the user to keep only one topic through all the dialogue. The better way will be to select the most suitable ontology for the current user's phrase topic. But there might be some reason to give some preference to the last involved ontology for the case if there are two or more anthologies detected as equally or very closely suitable. Anyway, long-term subject keeping is rather more for

avoiding strange and funny situations when the system "jumps" from one topic to another several times through the dialogue.

Short-term topic keeping could be more crucial. It determines which subsequent behaviour is to expect from the user and how to treat his following phrase. The two main parts of this problem are pronouns substitution and semantic expectation. The first one has been already mentioned above and could be solved by the following principles: looking for the nouns and name groups in one or two previous phrases; if there are several candidates for the substitution, ask the user about the correct option; if there is no candidate for the substitution in the last two phrases, just ask the user, what he means saying the pronoun. For the phrases given the program would be a better way to avoid using pronouns without previously used corresponding nouns in the same dialogue act. It merely makes the phrases by the program more understandable for a human.

A solution for the problem of semantic expectation consists in remembering the last phrase of the program type and purpose, which determines the sufficient reaction on the following user's action. Let us consider typical cases:

1) The program asks a question to the user. In this case, the expectation will be the answer to this certain question. However, the user has free will and could type anything: his phrase may be narrative, imperative or interrogative. Normally an answer to a question ought to be a narration, but in real cases, it does not always happen. But there also could be confusion when the user's phrase in a form of a question or imperative is the answer. For example, the system asks the user "What is your favourite movie?", and the user answers "Tell me a story". Grammatically this phase is an imperative. So it could be treated like the user has ignored the question and asked the system to tell him a story. But this also could be the name of a movie. There could be others examples like this – names of books and films sometimes may be made in a form of question or appeal. Here appears a question: does for all the question types the answer can be not in narrative form? No, it may be not for all. Here is the following list of question semantic types which can expect an answer that looks like an imperative: pointing, address, incoming direction, outcoming direction, separation, unification, substitution, instrumental, object of action,

purpose, meaning, topic, object of characteristic. In the example above the semantic question type is "object of characteristic" (it is requested). The question explains that this object should be of category "movie" and it should have the characteristic "favourite" for the user. The expected answer – a name of a movie and sometimes could be in a form of imperative. A question like answers might be correct for the following semantic types: incoming direction, outcoming direction, separation, unification, object of action, purpose, reason, meaning, estimation, topic, object of characteristic. If the user's phrase following the question of the program is a question of the imperative of those types a good decision for the system will be asking a clarifying question about whether these phrases are the object of the expected answer. If the user's answer to the clarifying question is "yes" it will be processed like an answer, else, if the answer is "no" the phrase will be processed normally regarding its type and content. In the last case, the asked question will be assumed as left without an answer.

2) The program phrase is a narration. Any type of phrase type may come from the user and it will be treated according to its type.

3) The program phrase is imperative. Here reaction on the following user's phrase depends on what has been asked. It the imperative assumes an action like to tell some information to the system it is equal to a question. If it suggests the user perform some action in the real world it is probably advisable, which might be given as a reaction to the previous phrases. In this case, there should not be any influence on the following user's phrase perception.

This type of dialogue system could also be staffed with a decision tree-like it was described above for the third (questionnaire) type. So the user's answers to the system's questions become bifurcation criteria that determine the following phrases the system gives to the user.

The last described dialogue system type is useful more for intersegment but it also could perform the functions of any one of these types. Moreover, the ability both to ask and answer the questions could make the working with the dialogue system more user's friendly and less bearing. Also, it allows the program to obtain information from the user in an unobtrusive matter.

Conclusions

Considered the types of natural language dialogue systems depending on the involvement and roles of the user and the conversation program: reference system, where all the initiative belongs to the user, who merely asking the system questions; reference system, where the system can ask the user technical and clarification questions; questionnaire system that is aimed rather address questions to the user than answer them; active initiative system which assumes narrative, interrogative and imperative phrases both from the user and the system sides; natural language control system.

Were analyzed two types of ontology used in our dialogue systems: simpler one, where categorized concepts combined into intersections are bound to the corresponding references on ready answer texts and more semantically structured, where concepts are represented by rather multiword terms bind to answers, definitions, comments and with each other by a wide range of semantic meaningful predicates. The second type is considered usage of standard OWL and RDF/XML or Neo4j data structures with arbitrary predicates approaches. It was shown that the method used for RDF/XML or Neo4j data structures with arbitrary predicates leads to simplification of the ontology formal structure and could simplify queries with the same purpose compared with standard OWL predicates usage only. However, for the first, simpler, type standard OWL is rather useful and convenient. The advantage of the first ontology type is its simplicity, ability to create atomization, universality for different languages and subject areas, and no need for deep semantic analysis. The advantage of the second ontology type is semantically precision and the potential probability of more certain and relevant answers obtained.

A tree-based semantic analysis method is proposed for the determination of the semantic type of the input user's phrase. 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 only one or might be several, but ranked semantic types. The semantic

type determines the corresponding appropriate formal query template. The concepts substituted into the template are to be taken from the defined parts of the initial considered words list considered on the certain tree positions, which depend on both the tree way and the chosen template.

A method was proposed for derivation from the sets of words obtained from the initial phrase to ones stored in the nodes of the graph database. It can both reduce and expand the context to the most appropriate one. The essence of the technique is the following. There is a set of all the terms from the ontology with the corresponding lists of lemmatized words and their parts of speech. The word set is compared with the words of these concepts. Both additional and missing words are taken into account. To estimate the similarity degree of the input words set and one from the file is used a specific metric, which gives different scores to matching, additional and missed words depending on their part of speech. That concept is to be selected for which the score will be the highest. Obtained in such manner terms are used to be substituted into the query template. The method is useful for ontology with long multiword terms.

An approach of preliminary queries ranking is considered. In the final node of the decision tree for the semantic type, determinations could be given several possible ones each of them corresponding to a certain query template. The essence is that those types are given in their possible relevance hierarchy. So if the previous one becomes unable to obtain any result it probably could be obtained by the following one, however, perhaps less adequate.

Considered in this work is the problem of the dialogue context keeping for different types of dialogue systems. It was shown that the most crucial this issue is for active initiative system type. To its solution is proposed the following approach. There should be both long-term and short-term contexts keeping procedures. The long-term one is aimed at giving preference to the ontology used when dealing with previous dialogue acts. If there is only a narrow subjected ontology in the system, this problem solves itself. If there are several different ontologies, following the last used one is needed for the program initiatives only, and the general dialogue subject should be able to be changed by the user's will. The short-term context keeping is the most actual after the dialogue act when the system asks the user a question or generates an

imperative phrase with the meaning of the question. If after that from the user comes a phrase of interrogative or imperative type and its type is incoming direction, out coming direction, separation, unification, an object of action, purpose, reason, meaning, estimation, topic or object of characteristic (for question) or pointing, address, incoming direction, out coming direction, separation, unification, substitution, instrumental, object of action, purpose, meaning, topic, object of characteristic (for an imperative phrase), the system should ask the user a clarifying question, whether this phrase the answer or the user merely has ignored to answer, but asking for something. The following activity of the system should depend on the user's answer to this query. For other cases, users' questions and imperative phrases are to be treated just as questions or imperative phrases without any clarifying. Also, short-term context keeping includes pronouns substitution. Possible ambiguities are also should be resolved by asking clarifying questions or by having an additional ontology where possible actions and characteristics of the concepts are stored.

Acknowledgements

This paper and the research behind it would not have been possible without the financial support of the National research foundation of Ukraine under the contract № 159/01/0245 from 07.05.2021, project title: "Transdisciplinary intelligent information and analytical system for the rehabilitation processes support in a pandemic (TISP)".

References

[Algorithmia, 2021] Algorithmia: Developer centre. Available from: https://algorithmia.com/developers/clients/python.

[Aylein, 2021] Aylein: NLP-enriched News Intelligence Platform. Available from: https://aylien.com/product/news-api?redirect=portal.

[Stsvrianou et al, 2007] A. Stsvrianou, P. Andritsos, N. Nicoloyannis. Overview and semantic issues of text mining. SIGMOID Record. Vol. 36, No. 3. 2007. pp. 23 – 34.

[Rajman et al, 1999] M. Rajman, R. Besancon. Stochastic distributional models for textual information retrieval. Proc. of 9th ASMDA. Lisbon, Portugal. 1999. pp. 80 – 85.

[Nenadic and Ananiadou, 2006] G. Nenadic, S. Ananiadou. Mining semantically related terms from biomedical literature. ACM TALIP Special issue on text mining and management in biomedicine. Vol. 5, No. 1. 2006. pp. 22 – 43.

[Caropreso at al, 2001] M. F. Caropreso, S. Matwin, F. Sebastiani. A leaner-independent evaluation of the usefulness of statistical phrases for automated text categorization. Text databases and document management: theory and practice. Hershy, PA. 2001. pp. 78 – 102.

[Kehagias et al, 2001] A. Kehagias, V. Petridis, V. G. Kaburlasos, P. Fragkou. A cooperation of word- and sense-based text categorization using several classification algorithms. Journal of intelligent information systems. Vol. 21, No. 3. 2001. pp. 227 – 247.

[Carenini and Zwart, 2005] G. Carenini, E. Zwart. Extracting knowledge from evaluative text. 3rd KCAP. Alberta, Canada. 2005. pp. 11 – 18.

[Antonellis and Gallopoulos, 2006] I. Antonellis, E. Gallopoulos. Exploring term-document matrices from matrix in text mining. Proc. of the SIAM Text mining workshop. Maryland. 2006.

[Yang and Liu, 1999] Y. Yang, X. Liu. A re-examination of text categorization methods. Proc. of SIGIR. Berkeley, CA. 1999. pp. 42 – 49.

[Dumais et al, 1998] S. Dumais, J. Platt, D. Heckerman, M. Sahami. Inductive learning algorithms and representations for text categorization. Proc. of the 7th CIKM. Bethesda. 1998. pp. 148 – 155.

[Sebastiani, 2002] F. Sebastiani. Machine learning in automated text categorization. ACM Computing Surveys. Vol. 43, No. 1. 2002. pp. 1 – 47.

[Jindal and Bing, 2006] N. Jindal, L. Bing. Identifying comparative sentences in text documents. Proc. of 29th SIGIR. Seattle, USA. 2006. pp. 244 – 251.

[Hatzivassiloglou and Mckeown, 1997] V. Hatzivassiloglou, K. R. Mckeown. Predicting the semantic orientation of adjectives. Proc. of the 35th ACL and

8th Conference of the European chapter of ACL. Nee Brunswick, NJ. 1997. pp. 174 – 181.

[Shaik et al, 2016] S. Shaik, P. Kanakam, S.M. Hussain, D. Suryanarayana Transforming natural language query to SPARQL for semantic information retrieval. International Journal of Engineering Trends and Technology. No. 7. 2016. pp. 347–350. DOI: 10.14445/22315381/IJETT-V41P263.

[Ochieng, 2020] P. Ochieng PAROT: Translating natural language to SPARQL. Expert Systems with Applications. No. 5. 2020. pp. 1–16. DOI: 10.1016/j.eswa.2021.114712.

[Yin et al, 2021] X. Yin, D. Gromann, S. Rudolph. Neural machine translation from natural language to SPARQL. Future Generation Computer Systems. Vol. 117. 2021. pp. 510–519. DOI: 10.1016/j.future.2020.12.013.

[Damljanovic et al, 2011] D. Damljanovic, M. Agatonovic, H. Cunningham. 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.

[Sun, 2018] C. Sun. A Natural Language Interface for Querying Graph Databases: master's thesis in computer science and engineering. USA: Massachusetts Institute of Technology, 2018.

[CypherConverter] GIT-hub Convert English sentences to Cypher queries documentation. Available from: https://github.com/gsssrao/english2cypher

[Litvin et al, 2020] A. A. Litvin, V. Yu. Velychko, V. V. Kaverynskyi. Method of information obtaining from ontology on the basis of a natural language phrase analysis. Problems in programming. No. 2-3. 2020. pp. 322 – 330. DOI: 10.15407/pp2020.02-03.322

[Sudakov and Malyokin, 2012] B. N. Sudakov. A.S. Malenkin. Metody sinteza estestvenno-yazykovykh tekstov v ekspertnykh sistemakh. Vestnik NTU «KhPI». (In Russian).

[Bolshakova, 2014] I. E. Bolshakova. Yazyk leksiko-sintaksicheskikh shablonov LSPL: opyt ispolzovaniya i puti razvitiya. Programmnyye sistemy i instrumenty: Tematicheskiy sbornik. Vol. 15. 2014. pp. 15 – 26. (In Russian).

[Tarasov, 2015] Д. Тарасов. Chatbot на нейронных сетях. (In Russian). Available from: https://habr.com/ru/company/meanotek/blog/256987/

[Levin, 2016] I. Levin. Chto mozhet i chego ne mozhet neyroset. (In Russian). Available from: https://habr.com/ru/company/neurodatalab/blog/335238/

[Kuki] Kuki - Chat with me. Available from: https://chat.kuki.ai.

[AIML] AIML Artificial Intelligence Markup Language. Available from: http://www.aiml.foundation/doc.html

[Gomez-Perez et al, 2002] A. Gomez-Perez, J. Angele, M. Fernandez-Lopez, V. Christophides, A. Stutt. A survey on ontology tools. Onto wed deliverable. Vol. 1. 2002.

[Antoniou, 2016] G. Antoniou. Semantic web. Moscow: DKM-Press, 2016.

[Goel, 2015] A. Goel. Neo4j Cookbook. Birmingham, 2015.

[DBEngines, 2021] DB-Engines ranking of graph DBMS. 2021. Available from: https://db-engines.com/en/ranking/graph+dbms.

[Litvin, 2020] A. A. Litvin, V. Yu. Velychko, V. V. Kaverinsky. Synthesis of chat-bot responses in the inflecting natural language based on the results of queries to ontology and analysis of the chat previous phrase. Information theories and applications. Vol. 27, No. 2. 2020. pp. 152 – 199.

[Litvin et al, 2021] A. A. Litvin, V. Yu. Velychko, V. V. Kaverynskyi. Tree-based semantic analysis method for natural language phrase to formal query conversion. Radio Electronics, Computer Science, Control, 2021, No. 2, pp. 105–113. https://doi.org/10.15588/1607-3274-2021-2-11.

Authors' Information

Anna Litvin *– V. M. Glushkov Institute of Cybernetics NAS of Ukraine; graduate student. Glushkova Avenue 40, Kyiv, Ukraine, 03187; e-mail: litvin_any@ukr.net*

The main directions of scientific research: natural language processing for analysis and synthesis of text in inflected languages, intelligent systems, microservices, chat-bots

Vitalii Velychko *– V. M. Glushkov institute of Cybernetics NAS of Ukraine; PhD, Senior Researcher. Glushkova avenue 40, Kiev, Ukraine, 03187; e-mail: aduisukr@gmail.com*

The main directions of scientific research: computer ontologies and their use in the educational process; information-oriented information systems with processing of natural language objects: ontological approach

Vladislav Kaverinsky *– Institute of Materials Science, NAS of Ukraine; Candidate of Technical Sciences. Senior Researcher, ul. Krzhizhanovsky 3, Kyiv, Ukraine; e-mail: insamhlaithe@gmail.com*

The main areas of scientific research: theory and mathematical modelling of phase and structural transformation processes, the evolution of the distribution function of dispersed systems, modification of cast metal, natural language processing for analysis, and synthesis of text in natural languages.