The Hybrid Chatbot System Combining Q&A and Knowledgebase Approaches

Yuriy Gapanyuk, Sergey Chernobrovkin, Aleksey Leontiev, Igor Latkin, Marina Belyanova, and Oleg Morozenkov

Bauman Moscow State Technical University, Moscow, Russia,
gapyu@bmstu.ru

Abstract. The paper discusses the development of hybrid chatbot system combining Q&A and knowledgebase approaches. The machine learning methods are used for answering question & answer pairs. The knowledgebase processing uses a metagraph model which requires hybridization of two approaches. The proposed approach of Q&A and knowledgebase hybridization is discussed.

Keywords: hybrid chatbot system, questions & answers, knowledgebase, metagraph, metavertex

1 Introduction

Nowadays chatbot systems became more and more popular. It is almost impossible to meet the company's website without chatbot. More and more efforts are being made to improve the quality of chatbots [1]. But the list of functions that chatbots can successfully perform is still restricted. Consider the most common of these functions.

The first function (function I) is question answering (Q&A). The most common case of Q&A is F.A.Q. (frequently asked questions). This is a list of question and answer pairs. If the user asks a question similar to the question in the list, then the corresponding answer from the list is returned to the user. There are many services implementing this function, for example [2], and also many research papers, for example [3].

The function II that is very close to function I is answering common phrases. A list of pairs is also used as the data model, differing in that in this case, there are no question and answer pairs but stimulus and response pairs. If the user sends phrase similar to stimulus in the list, then the corresponding response from the list is returned to the user. This function is implemented for example in chatterbot framework [4]. This function is not business-oriented but aimed to make chatbot system more human-like.

The function III is dialog scripts. Usually, dialog script is created in the form of the graph. The objective of chatbot is to route the user from the initial to the terminal vertex of the graph. The vertex of the graph is associated with some action. The action may be asking and answering user, calling web-service, etc.

Usually, the data from user input is parsed and stored in variables. These variables are substituted for the answering templates. There are also many services implementing this function, for example, flowxo [5] and rebotify [6].

The function IV is knowledgebase answering. In this case, dialog scripts are created dynamically based on user questions and data stored in the knowledgebase. The history of questions and answers is stored in the session. As far as we know, information systems implementing this function are in experimental phase.

The chatbot implementing function IV is the most general case including functions I, II and III. Indeed, if we can create scripts dynamically, it is unnecessary to draw static dialog script graph as in function III. The functions I and II may also be replaced with dynamical generation on the basis of knowledgebase.

From the business user point of view, the function IV is the most significant. Most users can provide price lists as a knowledgebase for questions answering. But function I also should not be abandoned because most users can also provide exact Q&A pairs.

The main goal of this article is to discuss the development of hybrid chatbot system implementing functions I for Q&A pairs answering and function IV for price list answering.

2 The hybrid chatbot system structure

The hybrid chatbot system structure is represented in Fig. 1.

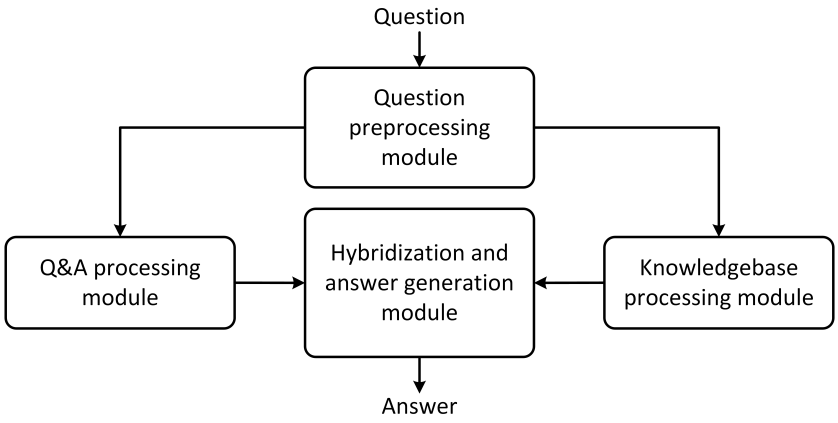

Fig. 1. The hybrid chatbot system structure.

First, the question is preprocessed using the question preprocessing module. The results of the preprocessing are passed to the Q&A and knowledgebase

processing modules that are executed simultaneously. Then the hybridization and answer generation module integrates the answers of two processing modules and returns the final answer to the user. Each module uses its own storage, and all modules have access to the user session storage. The storages are not shown in Fig. 1 in order not to confuse visualization of the figure.

The system is implemented in Python 3. The MongoDB database is used as storage backend. The mongoengine document-object mapper is used to simplify the work with MongoDB.

The principles of the operation of the modules will be discussed in details in the following sections of the article.

3 The question preprocessing module

The question preprocessing module prepares question text for Q&A and knowledgebase processing. The main library used in this module is NLTK. The Q&A preprocessing includes:

- tokenization;
- lemmatization;
- vectorization for TF-IDF and Doc2Vec [7] processing.

The vectorized question is passed to the Q&A processing module.

The main preprocessing task for knowledgebase processing module is concept recognition. The idea of concept recognition is based on NLTK sentence parsing. If all the words in the concept are in the same subtree of the parse tree, then the concept is considered to be recognized in the sentence. The set of recognized concepts is passed to the knowledgebase processing module.

4 The Q&A processing module

The input value of Q&A processing module is user question that is vectorized using TF-IDF and Doc2Vec processing. The module tries to find the closest question in the Q&A pairs list. Experiments were carried out with TF-IDF and Doc2Vec processing and with cosine and Euclidean distances.

Technically, there are two approaches to solve this task. The first one is to create a huge matrix $T(N \times M)$, where N is an amount of the questions in the database, M is the length of the dictionary. Since most of the questions won't contain that many words, the matrix will be too sparse. The advantage of this approach is its ability to use highly efficient linear algebra libraries. But there is also a very serious practical disadvantage: each time administrator adds new questions we need to check whether or not there are new words in them. And if there are, we need to rebuild the whole matrix.

The second approach is way agiler. Since we use the resulting vectors to compare between each others using metrics like cosine or Euclidean - we don't need to compare zeroes, so we don't need to vectorize all the questions in one

space (with same dimensions). Actually we could create small distinct spaces for each question in the database. Each vector space would have a size of $|q_i \cup q_a|$, where q_i is a question from the database, and q_a is the asked question. In this case, we don't need to create any matrices; we just compare asked question with the database questions one at a time. If the new questions are being added to the database, the only thing needed is to adjust IDF weights of the given words, no database transformation needed at all.

There are a lot of algorithm metrics which are matter for the practical use of the system such as accuracy, train and evaluation time. Below one can see experiments carried out with a goal to maximize accuracy and minimize train/evaluation time. Corpora used in the experiments was obtained from the Wikipedia Monolingual Corpora [8] project page. It contains about 4,5 millions of English Wikipedia articles. The Q&A dataset we've used contained not just question-answer pairs, but also some variations of the same questions. We've divided these question variants as 70/30 between the train and test sets.

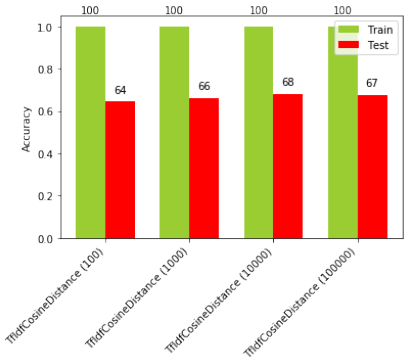

Fig. 2. Accuracy vs corpora size.

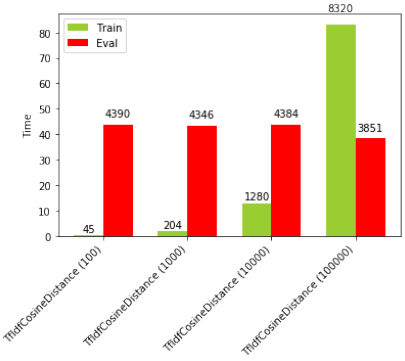

Fig. 3. Train/evaluation time vs corpora size.

	Corpora Size (sentences)	Train Accuracy	Test Accuracy	Train Time (sec.)	Evaluation Time (sec.)
0	100	1.0	0.644	0.458	43.905
1	1000	1.0	0.663	2.041	43.462
2	10000	1.0	0.681	12.801	43.844
3	100000	1.0	0.675	83.207	38.517

Table 1. Q&A experiments results.

As one can see in Fig. 2 and 3 and Table 1 the optimal corpora size for given dataset is 10000 sentences. The later enlargement makes the algorithm slower and doesn't improve accuracy.

The results of the experiments showed that the best quality is achieved when using cosine distance and ensemble model with TF-IDF (weight is 0.8) and Doc2Vec (weight is 0.2).

The Q&A processing module returns the corresponding answer for the closest question and the classifier confidence level.

5 The knowledgebase processing module

5.1 Why metagraph approach

The main idea of this module is answering price list questions. We assume that typical representation for price list is a denormalized table. The rows of the table are objects, and the columns of the table are attributes.

The class of machine learning algorithms that is most relevant for this task is recurrent neural networks (RNNs). LSTM [9] and Seq2Seq [10] neural networks topologies are actively used in language processing and machine translation. Now RNN-based non-factoid question answering neural models are under active research, for example, Neural Generative Question Answering (GenQA) [11] and attention-based LSTM [12] models. But these models also assume that there is a trainset of questions and answers which is not a case for price list questions answering.

It is especially important to note the DeepPavlov library [13] which combines several approaches and allows to create goal-oriented chatbots. It is possible that in the future this library will allow to abandon the knowledgebase processing module. But this library is now in a really early Alpha release and also heavily dependent on trainsets.

Thus, although machine learning algorithms are well suited for Q&A processing, they are not suitable enough for price list questions answering.

Therefore, to solve price list questions answering task, we propose to use methods based on knowledge and its processing.

Though the denormalized table is a good logical data model for knowledge representation, it is not good enough as a physical model because it does not allow to store and process data effectively. The first experiments were carried out with RDF storages. But the main problem of RDF model is poor N-ary relationship representation [14]. According to [15] the metagraph model can solve this issue. Therefore the metagraph model is used as a physical data model for a knowledgebase.

5.2 The brief description of the metagraph approach

A metagraph is a kind of complex graph model aimed for hierarchical graph description. The metagraph model simplifies N-ary relationship representation and complex contexts description.

According to [15] the metagraph is described as follows: $MG = \langle V, MV, E \rangle$, where MG – metagraph; V – set of metagraph vertices; MV – set of metagraph metavertices; E – set of metagraph edges.

The metagraph vertex is described by set of attributes: $v_i = \{atr_k\}, v_i \in V$, where v_i – metagraph vertex and atr_k – attribute.

The metagraph edge is described by set of attributes, the source and destination vertices (or metavertices): $e_i = \langle v_S, v_E, \{atr_k\} \rangle, e_i \in E$, where e_i – metagraph edge; v_S – source vertex (metavertex) of the edge; v_E – destination vertex (metavertex) of the edge; atr_k – attribute.

The metagraph fragment is defined as $MG_f = \{ev_j\}, ev_j \in (V \cup E \cup MV)$, where MG_f – metagraph fragment; ev_j – an element that belongs to the union of vertices, edges and metavertices.

The metagraph metavertex: $mv_i = \langle \{atr_k\}, MG_f \rangle, mv_i \in MV$, where mv_i – metagraph metavertex; atr_k – attribute, MG_f – metagraph fragment.

The example of metagraph representation is given in Fig. 4.

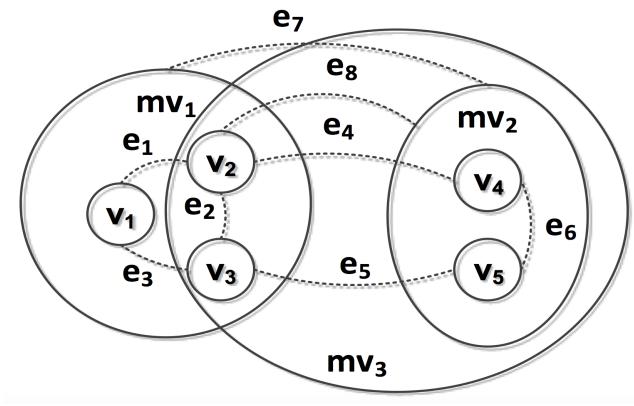

Fig. 4. The example of metagraph representation.

The example contains three metavertices: mv_1, mv_2, and mv_3. Metavertex mv_1 contains vertices v_1, v_2, v_3 and connecting them edges e_1, e_2, e_3. Metavertex mv_2 contains vertices v_4, v_5 and connecting them edge e_6. Edges e_4, e_5 are examples of edges connecting vertices v_2–v_4 and v_3-v_5 are contained in different metavertices mv_1 and mv_2. Edge e_7 is an example of edge connecting metavertices mv_1 and mv_2. Edge e_8 is an example of the edge connecting vertex v_2 and metavertex mv_2. Metavertex mv_3 contains metavertex mv_2, vertices v_2, v_3 and edge e_2 from metavertex mv_1 and also edges e_4, e_5, e_8 showing emergent nature of metagraph structure.

The metagraph itself is not more than a complex data structure. To process and transform metagraph data the metagraph agents are used.

The metagraph agent uses rule-based approach: $ag^R = \langle MG, R, AG^{ST} \rangle, R = \{r_i\}, r_i : MG_j \to OP^{MG}$, where ag^R – metagraph rule agent; MG – working metagraph, a metagraph on the basis of which the rules of agent are performed; R – set of rules r_i; AG^{ST} – start condition (metagraph fragment for start rule check or start rule); MG_j – a metagraph fragment on the basis of which the rule is performed; OP^{MG} – set of actions performed on metagraph.

The antecedent of a rule is a condition over metagraph fragment, the consequent of rule is a set of actions performed on metagraph. Rules can be divided into open and closed. If the agent contains only open rules it is called open agent. If the agent contains only closed rules it is called closed agent.

The consequent of an open rule is not permitted to change metagraph fragment occurring in rule antecedent. In this case, the input and output metagraph fragments may be separated. The open rule is similar to the template that generates the output metagraph based on the input metagraph.

The consequent of closed rule is permitted to change metagraph fragment occurring in rule antecedent. The metagraph fragment changing in rule consequent cause to trigger the antecedents of other rules bound to the same metagraph fragment. But incorrectly designed closed rules system can cause an infinite loop of metagraph rule agent.

Thus metagraph agent can generate the output metagraph based on the input metagraph (using open rules) or can modify the single metagraph (using closed rules).

The distinguishing feature of the metagraph agent is its homoiconicity which means that it can be a data structure for itself. This is due to the fact that according to definition metagraph agent may be represented as a set of metagraph fragments and this set can be combined in a single metagraph. Thus, the metagraph agent can change the structure of other metagraph agents.

The example of metagraph agent is shown in Fig. 5. The metagraph agent "metagraph rule agent 1" is represented as metagraph metavertex. According to definition, it is bound to the working metagraph MG_1, which is shown with edge e_4.

The metagraph agent description contains inner metavertices corresponds to agent rules (rule 1 ... rule N). Each rule metavertex contains antecedent and consequent inner vertices. In given example mv_2 metavertex bound with antecedent which is shown with edge e_2 and mv_3 metavertex bound with consequent which is shown with edge e_3. Antecedent conditions and consequent actions are defined in form of attributes bound to antecedent and consequent corresponding vertices.

The start condition is given in form of attribute "start=true". If the start condition is defined as a start metagraph fragment then the edge bound start metagraph fragment to agent metavertex (edge e_1 in given example) is annotated with attribute "start=true". If the start condition is defined as a start rule then the rule metavertex is annotated with attribute "start=true" (rule 1 in given example), Fig. 5 shows both cases corresponding to the start metagraph fragment and to the start rule.

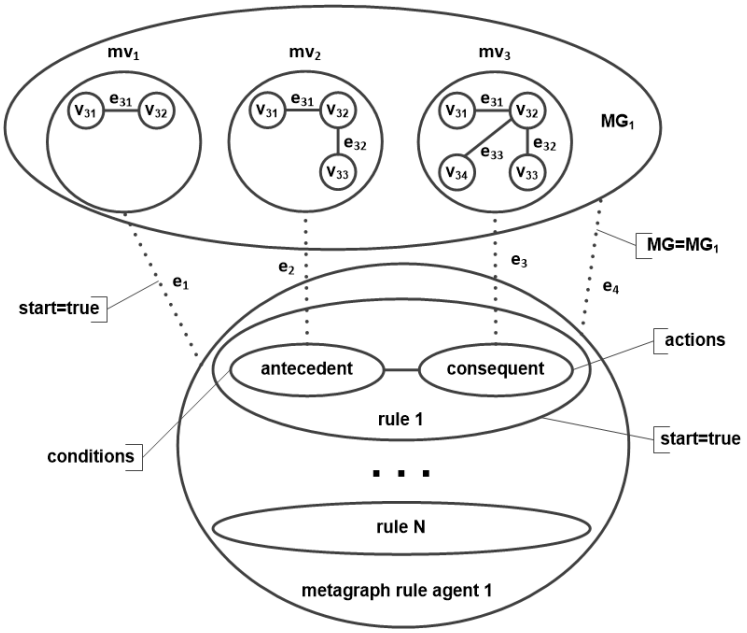

Fig. 5. The example of the metagraph agent.

Thus, the metagraph approach is a good basis for complex graph structures and their transformations representation. Now we can describe the knowledgebase in terms of metagraph approach.

5.3 The knowledgebase representation using the metagraph approach

The knowledgebase fragment in the form of metagraph is represented in Fig. 6. This example focuses on sales of office supplies. The price list is a table, each line of which contains the goods, and each column contains the characteristic of the goods.

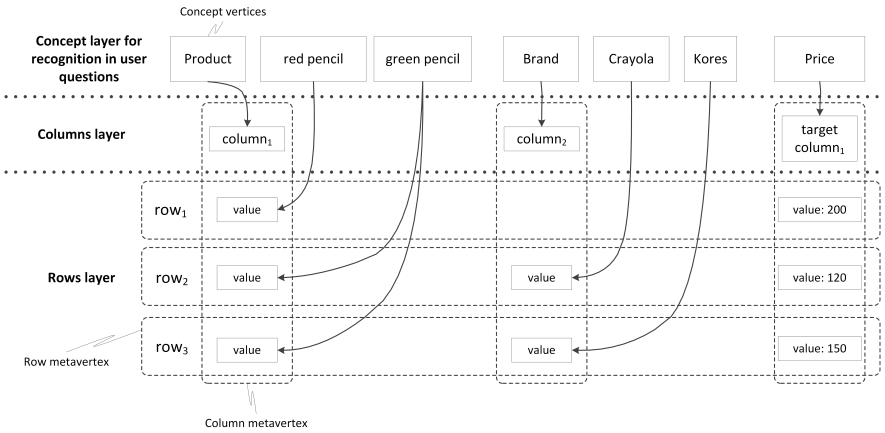

Fig. 6. The knowledgebase fragment in the form of metagraph.

The knowledgebase processing module solves two main tasks: questions answering and an active dialogue with the user. Each solution is represented using metagraph rule agent.

The questions answering agent uses the following rules:

- If the user's question contains concepts that match the data in the table, the corresponding column name is searched for each of these concepts, and the pair "column name: data" is saved in the user's session. If the session contained a pair corresponding to a column name, then this pair is deleted before insertion of a new pair.
- If the user's question contains concepts that correspond to the column name, a set of values for that column are displayed based on the session data. For example, if the user's question contains the concept of "products", the system will display a list of products in response: "red pencil", "green pencil". But if the session contains a pair {product: green pencil}, the user will be given a message that he orders green pencil.

An active dialogue agent is used to help the user choose a product. Then the main goal of active dialogue is to filter table in order to choose a single row. An active dialogue agent uses the following rules:

- After processing the user's question and saving the recognized concepts in the session, table rows are filtered based on the session data, and the number of filtered rows is estimated.
- If the single row is found, the goal is considered to be completed, and the user is shown information about target columns of the single row.
- If more than one row is found, then the system implements an active dialogue with the user and tries to complete the goal by asking auxiliary questions.

For this, a column with minimal diversity is defined, that is, it determines which column contains the minimum number of possible concepts, taking into account the filter based on the data of the current session. The active dialog will continue until the goal is completed, that is until the single row of the table is found.
– If the user specifies an incompatible set of concepts during the dialog process that leads to zero rows as a result of the filtering, a message is displayed indicating that no data has been found for incompatible set of session parameters and the user session is cleared.

The described process can be considered as a special kind of forward chaining based on table data. The usage of metagraph approach for knowledgebase representation allows using metavertices both as data elements used for question answering and information elements used for active dialog implementation.

It should be noted that it is not correct to talk about the accuracy of the knowledgebase processing module, because it does not use machine learning algorithms. The correctness of the module operation depends entirely on the accuracy of concept recognition.

6 Hybridization and answer generation module

Actually, the Q&A and knowledgebase processing modules are two different chatbot systems, and we need to implement their seamless integration. The main idea of integration is that the Q&A processing module is aimed to answer general purpose question while the knowledgebase processing module is aimed to answer specific questions based on knowledgebase data.

After the preprocessing phase, the Q&A and knowledgebase processing modules are called simultaneously in order to reduce the response time of the system. However, when analyzing module answers, the Q&A module responsible for general purpose questions has a higher priority.

If the text of the question is found in Q&A pairs with a high degree of conformity, then we recognize that question is general purpose and the corresponding answer from the Q&A processing module is returned to the user. A high degree of conformity means that the confidence level of the Q&A classifier exceeds the threshold. The value of confidence level threshold based on the results of experiments is about 85%.

If the confidence level of the Q&A classifier does not exceed the threshold, then we apologize that question is knowledgebase specific, and the answer from the knowledgebase processing module should be returned to the user. If the set of recognized concepts passed to the knowledgebase processing module is empty or the knowledgebase processing module returned empty answer then standard answer "Please, clarify your question" is returned to the user.

7 Experiments - dialog example

In this section we consider simple dialog example:

User: I want to buy a green beautiful pencil.

Comment: The concept 'green pencil' is recognized and stored in the session because words 'green' and 'pencil' are in the same subtree of the NLTK parse tree. The knowledgebase contains several rows with this concept. The system tries to offer the brand for the selected product.

Bot: We have green pencils of the following brands: 'Crayola' and 'Kores'.

User: What are the advantages of Crayola pencils?

Comment: The concept 'Crayola' is recognized. But the confidence level of the Q&A classifier exceeds the threshold, and the corresponding answer from the Q&A processing module is returned to the user.

Bot: The advantages of Crayola pencils ... (answer from Q&A).

User: Then I want to buy a Crayola pencil.

Comment: The concept 'Crayola' is recognized and stored in the session. The confidence level of the Q&A classifier does not exceed the threshold. The knowledgebase contains only one row corresponding to the session concepts. Then the value of the target column (price) is returned to the user.

Bot: It costs 120 price units.

8 Conclusion

The main result of our development is the chatbot system that integrates Q&A and knowledgebase processing functions.

The developed system can be considered as an automated consultant who provides information about the goods, answers detailed questions about the characteristics of the goods and helps to choose the goods.

The Q&A processing module uses machine learning methods while the knowledgebase processing module uses metagraph approach.

The Q&A processing module finding the closest question in the Q&A pairs list. The best quality is achieved when using cosine distance and ensemble model with TF-IDF and Doc2Vec.

The knowledgebase processing module uses metagraph model for a denormalized table data storage. The metagraph rule agents are used for knowledgebase questions answering and an active dialogue with the user.

As the future work, we plan to improve the quality of Q&A processing and add more question answering features to the knowledgebase processing module.

References

1. Khan, R., Das. A.: Build Better Chatbots. Apress (2018)
2. Microsoft QnA Maker official site, available at: https://qnamaker.ai/
3. Ranoliya, B.R., Raghuwanshi, N., Singh, S.: Chatbot for university related FAQs. International Conference on Advances in Computing, Communications and Informatics (ICACCI2017), 1525 - 1530 (2017)
4. ChatterBot conversational dialog engine official site, available at: https://github.com/gunthercox/ChatterBot

5. FlowXO official site, available at: https://flowxo.com/
6. Rebotify official site, available at: https://www.rebotify.com/
7. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. The 31st International Conference on Machine Learning (ICML 2014), 1188 - 1196 (2014)
8. Wikipedia Monolingual Corpora, available at: http://linguatools.org/tools/corpora/wikipedia-monolingual-corpora/
9. Sundermeyer, M., Ney, H., Schlter, R.: From feedforward to recurrent LSTM neural networks for language modeling. IEEE/ACM Trans. Audio, Speech and Lang. Proc. 23(3), 517 - 529 (2015)
10. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. Advances in neural information processing systems, 3104 - 3112 (2014)
11. AU - Huang, Y., Zhong, T.: Multitask learning for neural generative question answering. Machine Vision and Applications, 1432 - 1769 (2018)
12. Tan, M., Xiang, B., Zhou, B.: LSTM-based Deep Learning Models for non-factoid answer selection, available at: https://arxiv.org/pdf/1511.04108.pdf
13. DeepPavlov official site, available at: http://deeppavlov.ai/
14. Defining N-ary Relations on the Semantic Web. W3C Working Group Note 12 April 2006, available at: http://www.w3.org/TR/swbp-n-aryRelations/
15. Chernenkiy, V., Gapanyuk, Yu., Nardid, A., Skvortsova, M., Gushcha, A., Fedorenko, Yu., Picking, R.: Using the metagraph approach for addressing RDF knowledge representation limitations. Internet Technologies and Applications (ITA) 2017, 47 - 52 (2017)