A Virtual Dialogue Assistant for Conducting Remote Exams

Anton Matveev, Olesia Makhnytkina, Inna Lizunova, Taisiia Vinogradova,
Artem Chirkovskii, Aleksei Svischev, Nikita Mamaev
ITMO University
Saint-Petersburg, Russia
{aymatveev, makhnytkina, lizunova, tbvinogradova, chirkovskii, svischev, nmamaev}@itmo.ru

***Abstract*—In this paper, we demonstrate issues and possible solutions to building an Artificial Intelligence Dialogue Assistant for human-machine communication. We specialize it for conducting written exams at online education platforms, talk about the main logical components of the system: knowledge base, question encoder, question generation module, question analysis module. As a knowledge base we consider text fragments representing parts of the course of text fragments representing parts of a course and is a source for a format ontology; and is also sourced for neural network generation of fact-based questions in question generation module and building dependency trees for answer evaluation in question analysis module.**

I. Introduction

A. The Problem

The recent trends to automate education processes pose new difficult challenges for the developing digital society. Due to the limited capabilities of online education platforms, assessment of student performance is mostly done via multiple-choice questions, which is not ideal for all-round evaluation of skills and knowledge students acquire during a course. Unstructured interviews in the form of a written dialogue between the examiner and the examinee could provide a much more relevant experience for both parties and automation of such a tool is sought for by education providers looking for improvement and optimization of their processes.

Creating dialogue and Question and Answer systems is a rapidly developing field, however, most of the systems are built for figuring out an answer to the question of the user but few focus on generating questions based on some knowledge base and a dialogue context. While the desire of the market for such systems is raising, modern systems are yet to satisfy the performance requirements.

B. The Goals, the Requirements, and the Limitations

The scope of the problem spans across a variety of different fields of science, and many of them can have a legitimate claim to own it with respect to the history of the progress in the field. However, it is clear, that, ultimately, the problem lies within an intersection of multiple fields, which makes it difficult to give a formal description of the problem and criteria for evaluating discovered results. Thus, we feel that it is crucial to clearly state the environment we are working in and the kind of results we are seeking.

First, for the limitations, we believe it would be unproductive to either try to find purely theoretical solutions, which may or may not be competitive comparing to products of giants like Google given similar access to the amount of computation power they can have; and, on the other hand, to narrow the focus of the research to a very small portion of the problem and have a provable case of improvement for this particular subtask. This means the most interesting opportunity is to say that the limitation is that the system should be able to operate with the amount of computational resources of a home personal computer to a small company and should have a clear pathway to scaling. For this kind of a system, we are talking about both runtime and compile (training) time performance.

Second, for the requirements, the obvious case is to attempt to compete with state-of-the-art solutions. However, due to small number of such systems and ambiguity of what exactly the measurement of performance is, we believe the most reasonable approach is a combination of the system conforming to some clear basic criteria - at least working and producing results within the given limitations, - and a subjective evaluation of the results being meaningful and interesting for particular cases.

Third, the goal is to research and implement a number of methods, which combined in certain ways fit the limitations and the requirements, and to compare how different combinations behave.

II. Related work

Dialogue systems follow one of the three common routes: case- or rule-based approach; end-to-end deep learning; or a hybrid approach. Solutions based on rules provide high quality and - very important for certain cases - predictable results but require volumes of manual labor of highly qualified specialists in both the linguistic and the field of the problem; and also are very expensive for modification and extension when new information arrives. For online education platforms, this approach puts a heavy burden to rebuild or build a brand new collection of rules for each new course, while existing courses tend to update annually. End-to-end systems, based on methods of natural language processing and deep learning, have issues as well, mostly arising from the unpredictability of the results. The hybrid approach is an attempt to enjoy the strongest points of both methods by combining them in various ways. All of these methods are employed by modern

systems trying to answer the question of the user. Systems asking the user a question, like Eliza, exist for a while now but are yet to be able to account for a dialogue context and ”understand” the answer.

A more general task - Automatic Question Generation (AQG) - has been tackled in many NLP works, most often with the purpose of knowledge assessment. As per the review by Kurdi et al. [1], the majority of them describe using ext as input and operate with template-based or rule-based algorithms for generating questions. The attempts to use deep learning have only started after the release of SQuAD and MS MARCO datasets in 2016. For example, Du et al. [2] use an encoder-decoder network to create factual questions based on the SQuAD corpus in an end-to-end fashion. Zhao et al. [3] use encoder-decoder architecture with self-attention, copy and maxout mechanisms to construct factual questions. Similarly, Scialom et al [4] use a small Transformer network with a copying mechanism to generate factual questions. The task of conversational question generation is new and is largely supported by the CoQA dataset [5]. For instance, Gao et al. [6] [6] jointly encode the passage and the dialogue history with an encoder-decoder model; Pan et al. [7] propose a reasoning mechanism for reading the conversation history iteratively. Pan et al. were also the first to introduce a method to improve the quality of generated questions via policy gradient. A similar concept (using a QG system to improve a QA system) is seen in the work by Duan et al. [8]

III. Review of the system

The dialogue assistant, developed by the authors of this paper and intended to conduct a written examination, can work in a number of ways:

- By generating a set of questions to a text corpus.
- By generating supplementary questions based on the ser’s answers.
- By evaluating relevancy and correctness of the answers and figuring out the way to continue the dialogue or conclude it.
- By controlling the flow of the dialogue to adhere to the desired length of a dialogue.

A. Technical aspects

To conform to the requirements it is important not only to focus on more scientific subtasks but also to pay attention to the general design of the system and technical implementation of it.

The implementation of the system is designed for both the simplicity of external interfaces and clarity of internal processes, see Fig. 1.

The top level of the system is split between the compo-nent, offering an interface to the system, and the subsystem implementing business-logic.

The client of the system only interacts with the former and receives from it specification of datatypes and scenarios supported by the system.

The business-logic subsystem consists of modules implementing business-logic subroutines, a persistent storage, and a coordination module responsible for the composition root of the supported scenarios.

The system is modular and distributed, providing agility and scalability of the solution, clear visualization of the processes, and a possibility to employ parts of the system separately for a particular task. The system is packaged as a set of modules (libraries) with interfaces that can be run on a machine or a set of machines connected in a local network or via the internet.

B. Logic Components

Bridging between technical details and the nature of the problem, the following concepts provide a way to design the next level of abstraction of the system.

Knowledge/Text Database: Knowledgebase consists of text corpora and formal ontologies.

Question Encoder: In the simplest case it is just a question in a natural language, however, another approach is to represent it as a construction with a higher or a lower number of formal restrictions.

Question Generation Module: Generates a question in a natural language in the context of a text fragment either via rules or by a trained neural network.’

Question Analysis Module: An evaluation (a measure) of an answer. In our case, is directly mapped to the module calculating ”distances” between Dependency Trees.

The ”business-process” running inside the system is demonstrated in Fig. 2.

1) Knowledge/Text Database: The knowledge base consists of text fragments representing parts of a course and is a source for a format ontology; and is also sourced for neural network generation of fact-based questions and building dependency trees for answer evaluation. More complex questions are generated with the support of a formal ontology. The obstacles for creating a virtual assistant built on top of a knowledge base are: 1) the result must be in a natural language and should make sense 2) it requires the input text fragment to be parsed into a formal representation for the knowledge base 3) the knowledge base architecture itself requires a careful design. On the basic level, the first problem can be solved by employing existing template-based solutions for natural language generation; and for the third, there are several commonly accepted standards for ontology structure. The second problem, however, - extracting facts and terms from a text fragment - still is a present problem in natural language processing and is of critical importance for creating an ontology-based virtual dialogue assistant since it is both needed to avoid expensive manual labor of highly qualified experts, and to analyze user’s answers dynamically during their interactions with the system.

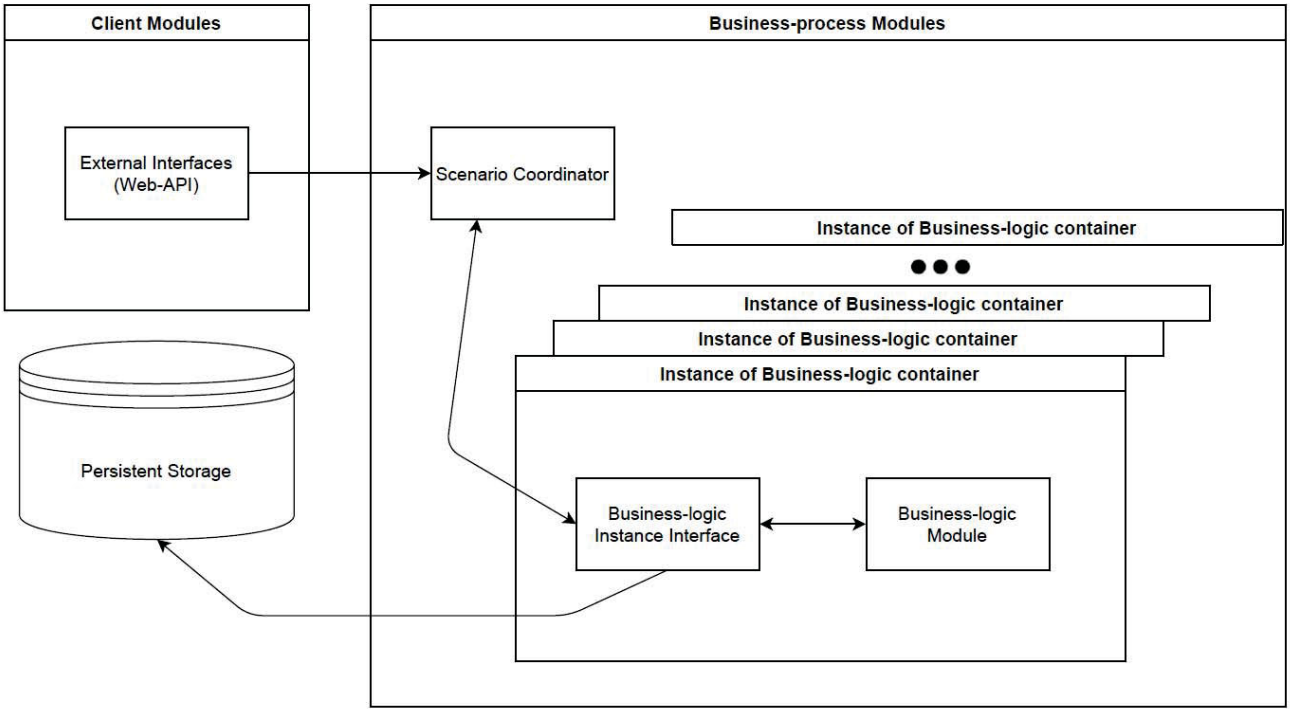

Fig. 1. High-level architecture of the system

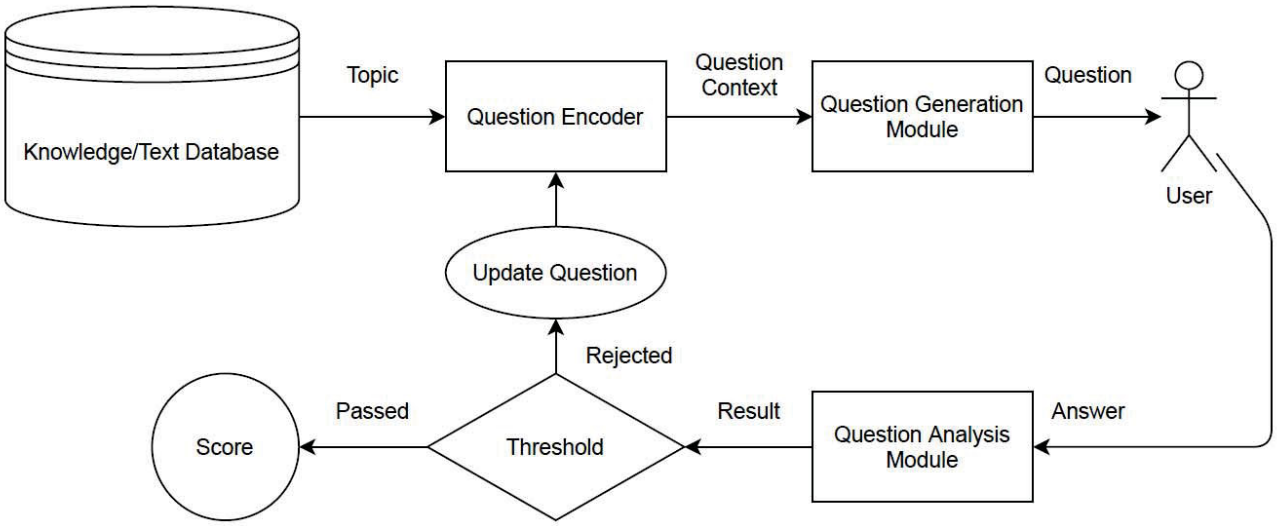

Fig. 2. The outline of the scenario, supported by the system

Terms and facts extraction is based on lexical properties such as part of speech labeling, named entities labeling, and coreferent relations. Unfortunately, for Russian language, which is famous for its fusional nature and flexible word order, there are no existing databases for labeled named entities of significant enough volume to train a strong model, therefore we suggest to rely on characteristics on the kind of input data we target: text fragments in natural language, but with high density of particular cliche which intended to improve readability. This way it is possible to match such cliche with the formal ontology, extract facts and terms, normalize them, and add them to the ontology along with the information about their connections. An improvement to this method is to employ language models to unify synonymical entities and empirical knowledge about specific structures found in a particular media, for example, certain text corpus

may have important terms highlighted, or that textbooks tend to have lists of terms placed by the end.

The process of formalization of a particular field of knowledge is called Ontology Learning. Obtaining such ontology takes several steps involving various methods of semantical analysis. The formal structure of a field of knowledge allows to:

1. Group entities no just on the level of words and phrases, but concepts, i.e. sets of terms referring to the same idea (analogous to WordNet synsets). Here we call terms the commonly used phrases referring to a singular idea, e.g. ”natural language processing”, ”machine learning”, etc..

2. Evaluate relations between concepts and their hierarchy. In terms of Formal Concept Analysis mechanism, a concept can be represented as a set of objects (i.e. terms in our case) and a set of attributes shared by all of those objects.

3. Resolve coreference, i.e. all relations and attributes of a synonym or another reference to the target concept and are also applied to the concept itself.

Extracting concepts is implemented by searching a text for specific objects (entities) and attaching respective attributes to them (properties). At this moment, the solution is based on classical methods with syntactic parsers and rule systems, and improvement by utilizing neural networks and linguistical models like BERT is planned.

The valuable property of modeling a field of knowledge as concepts is the possibility to analytically find more general or more specific concepts. This allows us to build a hierarchy of concepts, for example, the concept of ”languages” can be split into concepts of ”formal”, ”artificial”, and ”natural” languages. Additionally, it is possible to describe the category of constructed languages (artificial and formal) and human languages (artificial and natural). Having the hierarchy of concepts we can start working with logical constructs of a higher level of abstraction; for example, all descendants of a concept have all its attributes, and, if two terms do not belong to the same concept, we can determine the difference and generate a question regarding that.

A sidestep from the area where methods of Formal Concept Analysis would be to attempt to label relations between concepts not supported by the method. Any such relationship can be expressed as an object/attributes pair. Let us look at an example in notation of Prolog language:

A text fragment: ”Natural language (NL) is a language used by people to communicate and not intentionally created. Examples would be Russian, English, Chinese, etc.”

An example of an ontology for this fragment might be:

Term - term(”Natural language”)

Synonymy relation - syn(”Natural language”, ”NL”)

Hypernymy/hyponymy relation - hyp(”Language”, ”Natural language”)

Sample (instance of relation) - ekz(”Natural language”, ”Russian”)

Attribute association - attr(”Natural language”, ”used by people to communicate”)

Attribute association - attr(”Natural language”, ”is not intentionally constructed”)

Also such knowledge base requires rules for automatical relation inference, for example:

hyper(X, Y) :- hyp(Y, X) (if Y is hypernym of X, than X is hyponym of Y)

attr(Y, Z) :- attr(X, Z), syn(X, Y) (if X has an attribute Z and Y is a synonym of X, than Y has an attribute Z as well)

2) Question generation module: The question generation module is implemented for two modes.

The first mode is a question generator based on an ontology. Ontology allows not only to ask factual questions - which most of the neural networks trained for QnA model text generation do - but also to generate descriptive questions (e.g. ”Describe X”, that is, list its attributes), discriminative questions (e.g. ”What is the difference between X and Y, that is, which of their attributes are not shared between them), closed questions (e.g. ”Is it true that X has the property Y), and even inverse (e.g. ”Which properties X does not have?”)

Having the ability to generate this variety of questions is crucial for an effective virtual assistant since it may provide the user with tips, which more closely resembles a natural interview.

Question: ”What are the characteristics of a natural language?”

Query: ? - attr(”Natural language”, X)

Response:

X = ”is used by people to communicate”

X = ”is not intentionally constructed”

Question: ”What are some examples of artificial languages?”

Query: ? - ekz(”Artificial language”, X)

Response:

X = ”Esperanto”

X = ”Lojban”

X = ”Toki Pona”

X = ”Sindarin”

Question: ”What is common between formal and artificial languages?”

Query: attr(”Formal language”, X), attr(”Artificial language”, X)

Response:

X = ”intentionally constructed”

Another implementation is based on deep neural networks.

CQG (Conversational Question Generation) attempts to generate the next question based on a text fragment and a dialogue context, that is question/answer pairs to the text fragment. It is closely connected to the CQA (Conversational Question Answering) problem, which, in turn, is a field of the QA (Question Answering) problem. However, while from QA and QG points of view generating a question or an answer is a static solution, CQA and CQG consider the problem as a time series with attached dialogue context and coreferent connection and requires a much deeper understanding of what part of the available information should be the focus to source from.

The first step was to prepare data by semi-automatic translation of English training data to Russian. The second step was to train a ReDR (Reinforced Dynamic Reasoning) network.

3) Question Analysis Module: To prepare data for evaluation of the correctness of answers by fuzzy comparison of grammar dependency trees we have to perform graphematical, morphological, and syntactic analysis. For graphematical analysis, tokenization by sentences and then by words is performed. For morphological analysis, we perform lemmatization, i.e. figuring out stems and labeling with parts of speech and extracting morphological features. For syntactic analysis, for each sentence, we build a dependency tree with words for nodes and types of semantical connections for edges.

In this work we use the semantical parser UDify [9], which produces a result adhering to a unified system of labeling parts of speech, morpho-syntactical attributes and types of syntactical connections, developed by Universal Dependencies [10], to support the development of a multilingual syntactic analyzer, inter-language training, and analysis from the point of view of typology of a language. This markup is universal for all modern languages and brings together knowledge of various linguistic theories. According to this markup system, labels for parts of speech are separated into three categories: open class words, closed class words, and others. The first class consists of 6 tags, which are attached to independent parts of speech, typical for most of the languages. The second class has 8 tags, which are attributed to supporting parts of speech. The third class contains 3 tags for symbols, not representation particular words, or some special cases. There are 24 tags offered for labeling morphological attributes, which are split into lexical and inflectional groups. In the inflectional group, there are nominal and verbal features, however, this separation is fluid since there is no universal rule deciding a specific feature can only attach to nouns or only to verbs.

For semantical dependencies, there are 37 tags, which correspond to basic linguistic principles of sentence structure building or describe supplemental relations. Fig. 3 demonstrates examples of building dependency trees for student's answers to the question "What kind of a language can be called natural?"

After that, we extract features from the vector space of the pre-trained language model RuBERT [11], which is trained on Russian Wikipedia and news data. The basis for this method of building embeddings is the Transformer architecture and the idea of training a predictor of words, obscured by a mask.

To evaluate the user's answer we apply a measurement of the relevancy of the answer. The "relevancy" term in natural language processing often refers to how well the results provided by a search engine corresponding to the query of the user, i.e. how closely do the information fit to the needs of the user.

This approach fits a question-answering system and can be employed to evaluate the user's answer to the question asked by the system. For documents ranking a function is taken which reaches extremum fitting to the definition of relevance. To use is for a single document, though, requires to calculate a threshold for deciding whether the answer is relevant or not.

For this project, to evaluate a user's answer we perform a comparison of dependency trees for text fragments, which allows comparing whole phrases/sentences: a text fragment from the text corpora and a user's answer.

The grammar dependency tree is a way to illustrate syntactical structures of a sentence, where all connections are considered subordinate, the root of the tree is the predicate, and the prepositions are described by how they control a form of a noun.

There are various ways to measure distances between dependency trees, and they are much easier to work with comparing to ontologies or raw texts.

A simple way to compare dependency trees for which there already are built children-trees is to calculate intersections of such relationships. In this case, the distance between two strings can be calculated as:

$$E = \frac{|Q \cap T|}{|Q|} \tag{1}$$

where E is a measure of proximity,

Q is the set of tuples of dependencies in the "correct answer",

T is the set of tuples of dependencies in the user's answer.

To evaluate how close the user's answer to one of the correct answers we need to figure out a threshold for making a decision.

A more complicated, but also more precise method for comparing dependency trees is predicate matching. This measure is a more detailed version of the one before.

Now, we are not only looking at triplets, constructed from two nodes, but also at all semantical relationships of a semantical root (predicate relationships) at the verb position. We compare the predicate of the correct answer to the predicate, extracted from the user's answer.

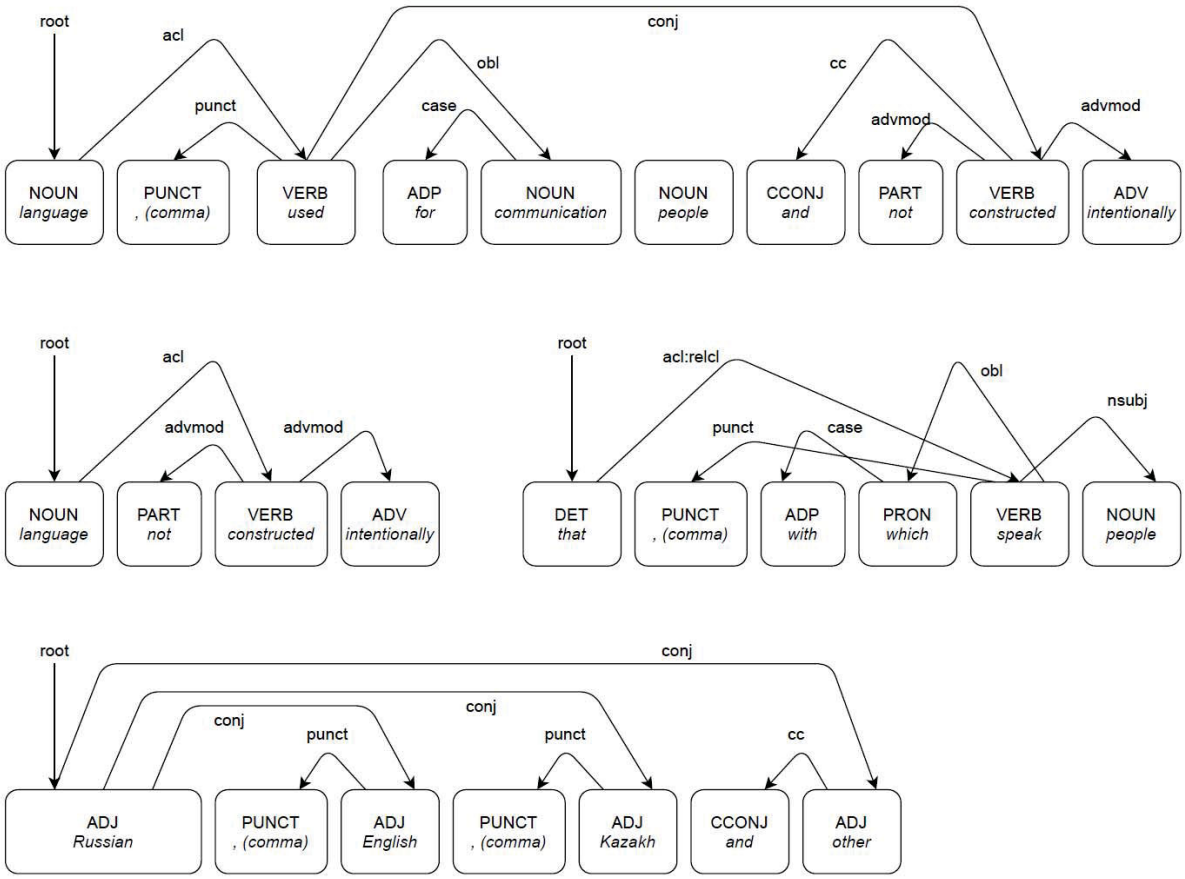

Fig. 3. Examples of dependency trees

The measure of similarity $Sim_{Term}(t_1, t_2)$ between terms t_1 and t_2.

$$Sim_{Term}(t_1, t_2) = J(W_1, W_2) = \frac{|W_1 \cap W_2|}{|W_1 \cup W_2|} \qquad (2)$$

Where W_1 and W_2 are sets of context words from WordNet dictionary to the terms t_1 and t_2 respectively.

Then, if t_u is a term from the user's answer and t_a is a term from the correct answer, T_u and T_a - sets of terms from the user's and the correct answers, the evaluation of the similarity between the user's predicate p_u and the correct predicate p_a is calculated by

$$Sim_{\arg s}(p_u, p_a) = \frac{\sum_{t_u \in T_u} \max_{t_a \in T_a} \left(Sim_{ExprTerm}(t_u, t_a) \right)}{|T_a| + \left| \{ t_u \in T_u : \max_{t_a \in T_a} \left(Sim_{ExprTerm}(t_u, t_a) \right) = 0 \} \right|} \qquad (3)$$

The total similarity measurement of the whole predicate is calculated as a product of measurement from 3 and the similarity of verb-terms. Such measure accounts for context words, which deals with issues of, for example, Homonymy.

Another method is the depth-first fuzzy search, which works this way:

1. Follow the same paths for both graphs from the roots and picking the same edges and nodes with identical labels.
2. Every time nodes match, add points to a cumulative score:

2.1. Edge match

2.1.1. Edges of different types can be assigned different weights

2.1.2. Some edges and nodes can be dropped in one of the graphs

2.2. Node match

2.2.1. Letter-by-letter match - 1 point

2.2.2. Matched lemmas - 0.5 point

2.2.3. One lemma is a substring of another - 0.5 point

3. Compare the cumulative score to the threshold

Node connection types (coordination, subordination, etc.) determine weights assigned to edges. The more important the connection for preserving the semantic structure of the sentence, the larger the weight should be assigned to it.

This method expands on the depth-first fuzzy search similar to how predicate matching expands on relationships sets intersections counting.

To compare the user's answer to the correct answer, we suggest the following:

Given text phrases as dependency trees, first simultaneously path along the dependency trees, similarly to the depth-first fuzzy searching. Each time an edge is skipped a set amount of points is deducted. Each step, compare nodes letter-by-letter, compare lemmas, and in the vector space, created by a neural network, calculate dot product of the two nodes. Meanwhile, collect all paths, and for each one arriving to the final node, calculate the cumulative score. The best score would indicate the best path and is accepted as the measure of similarity between the two text fragments.

IV. Conclusion

This work proposes a virtual dialogue assistant for conducting remote exams to enable supports the entire process of taking the exam in the form of an interview. The online exam system can create a large number of simple questions based on the texts of course lectures and compare the student's answer with the reference answer, which in our opinion is the best option for conducting the exam, then even test. Improvement of the system is represented in the form of implementing the possibility of maintaining a student-system dialogue, when the system is able to ask clarifying questions, and the response analysis module takes into account all the student's responses when calculating the final grade. However, when creating and improving such systems, one of the problems is the complexity of evaluating the quality of its work.

While the "subjective test" proves to be a solid choice as an evaluation system for a solution to a problem in terms of it fitting to "sane" requirements, all according to Occam's razor, the complexity of the problem makes it very difficult to reason about when trying to produce some sort of comparison to similar solutions to similar problems. Artificial Intelligence solutions, and in particular dialog systems, taking a place at the intersection of variety of fundamental disciplines, enjoy a significant number of degrees of freedom, allowing for exponential growth of opportunities, but blurring the sense of measurement of performance of a system due to increasing number of aspect towards which the performance can be evaluated.

It appears building systems such as the one described in this paper, more than anything requires very careful and strict formulation of the aim and criteria of achieving the goal. Since the comprehensive evaluation of such systems takes a large amount of testing data, preparation of the data should be accounted for when planning the research and development cycles of a project.

In spite of difficulties with evaluating results, nevertheless, the potential of such systems definitely is there, and, even though subjectively evaluated, the results prove to be enlightening and interesting.

Acknowledgment

This work was partially financially supported by the Government of the Russian Federation (Grant 08-08).

References

[1] G. Kurdi, J. Leo, B. Parsia, and S. Al-Emari, "Asystematic review of automatic question generation for educational purposes", *International Journal of Artificial Intelligence in Education*, Nov. 2019.

[2] X. Du, J. Shao, and C. Cardie, "Learning to ask: Neural question generation for reading comprehension", *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1342–1352, Jan. 2017.

[3] Y. Zhao, X. Ni, Y. Ding, and Q. Ke, "Paragraph-level neural question generation with maxout pointer and gated self-attention networks", *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3901–3910, Brussels, Belgium, Oct.-Nov. 2018, Association for Computational Linguistics.

[4] T. Scialom, B. Piwowarski, and J. Staiano, "Self-attention architectures for answer-agnostic neural question generation", *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6027–6032, Florence, Italy, July 2019, Association for Computational Linguistics.

[5] S. Reddy, D. Chen, and C. D. Manning, "Coqa: A conversational question answering challenge", *Transactions of the Association for Computational Linguistics*, 7:249–266, Mar. 2019.

[6] Y. Gao, P. Li, I. King, and M. R. Lyu, "Interconnected question generation with coreference alignment and conversation flow modeling", *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4853–4862, Florence, Italy, July 2019, Association for Computational Linguistics.

[7] B. Pan, H. Li, Z. Yao, D. Cai, and H. Sun, "Reinforced dynamic reasoning for conversational question generation", *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[8] N. Duan, D. Tang, P. Chen, and M. Zhou, "Question generation for question answering", *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 866–874, Copenhagen, Denmark, Sep. 2017, Association for Computational Linguistics.

[9] D. Kondratyuk and M. Straka, "75 languages, 1 model: Parsing univer-sal dependencies universally", *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th Inter-national Joint Conference on Natural Language Processing (EMNLPI-JCNLP)*, pp. 2779–2795, Hong Kong, China, 2019, Association for Computational Linguistics.

[10] M. Marneffe, T. Dozat, N. Silveira, K. Haverinen, F. Ginter, J. Nivre, C.D. Manning, "Universal Stanford dependencies: A cross-linguistic typology", *Proceedings of the Ninth International Conference on Lan-guage Resources and Evaluation (LREC'14)*, 2014, pp. 4585–4592.

[11] Y. Kuratov, M. Arkhipov, "Adaptation of Deep Bidirectional Multilingual Transformers for Russian Language", arXiv:1905.07213, 2019, Web: https://arxiv.org/abs/1905.07213