NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS

Olga Vinogradova, Nikita Login

THE DESIGN OF TESTS WITH MULTIPLE CHOICE QUESTIONS AUTOMATICALLY GENERATED FROM ESSAYS IN A LEARNER CORPUS

BASIC RESEARCH PROGRAM

WORKING PAPERS

SERIES: LINGUISTICS
WP BRP 60/LNG/2017

Olga Vinogradova[1], *Nikita Login*[2]

THE DESIGN OF TESTS WITH MULTIPLE CHOICE QUESTIONS AUTOMATICALLY GENERATED FROM ESSAYS IN A LEARNER CORPUS[3]

Learner corpora have great potential as sources of educational material. If a corpus contains annotations of mistakes in student works, it can be of use for the recognition and analysis of the most common error patterns. The error-annotation system of the learner corpus REALEC makes it possible to automatically generate different types of test questions and thus form exercises from the corpus data. This paper describes the creation of an automatic multiple-choice generator which works with the specific types of the student errors annotated in the texts of examination essays.

Keywords: learner corpus; computer-assisted language learning; multiple choice questions; English as a second language; corpus methods in language teaching

JEL Classification Code: Z19

[1] National Research University Higher School of Economics (Moscow, Russia). School of Linguistics. Associate Professor; E-mail: ovinogradova@hse.ru
[2] National Research University Higher School of Economics (Moscow, Russia). School of Linguistics. E-mail: nvlogin@edu.hse.ru

[3] The article was prepared within the framework of the Academic Fund Program at the National Research University Higher School of Economics (HSE) in 2016–2017(grant №16-05-0057) and the Russian Academic Excellence Project '5-100'.

Introduction

The research is based on student texts in the Russian Error-Annotated Learner English Corpus (REALEC), which includes essays of two types written by second-year students of the Higher School of Economics (HSE) in the years 2014-2016. Errors in the texts were outlined by EFL experts and annotated with the help of a comprehensive hierarchical error categorisation scheme. In this paper we, first, describe the design of software solutions for automatic generation of multiple choice questions, and, second, analyse the results of the test made up of questions automatically generated on the basis of the annotated errors. The tools used for software design are Python 3.6 standalone environment, NLTK (Natural Language Toolkit), and Stanford Part-of-Speech Tagger.

The main difficulty in generating test questions from the corpus material is to predict the types of confusable options for a specific word. The process of test-making may involve multiple iterations of software development and above all a series of trials in order to create the answer options that are most likely to be challenging for students.

Similar approaches were considered by the following authors:

Volodina [2008] uses the Stocholm Umeå Corpus (SUC) to generate practice exercises, including multiple choice questions. The solution presented works according to a finite-state machine principle. At the first stage the target vocabulary is defined through either automatic or manual procedures. Then part-of-speech and morphosyntactic tags, and also the frequency band of the word are checked, and three words sharing the same characteristics are returned. This approach is ideal for lexical exercises as it uses the methods of distributive semantics. However, this approach does not involve the use of error annotation and is capable of generating only lexical choice exercises. Besides, our task is more specific: whereas SUC contains texts of students at different levels of language proficiency, and the author creates a tool for learning of Swedish in general, our direction of research lies towards preparation for a specific type of written examination.

Aldabe Arregi [2011] describes the creation and implementation of a system called ArikIturri, a computer-assisted learning tool for Basque and English languages, which generates questions automatically. This system uses a corpus-based approach and can generate questions of five types: error correction, fill-in-the-blank questions, word formation, multiple choice questions and short answer questions. Chapter VI.3 is dedicated to the use of ArikIturri for generating multiple choice questions involving the choice between verbs from the Academic Word List. Error annotation approach is not used in this work, instead, the author uses a cooperation of sentence selection and distraction generation modules. The distracters for the verbs are chosen using distributive semantics. The solution for Basque language involves a learner corpus, whereas the solution for English uses the British National Corpus as a dataset.

Mostow and Jang [2012] present the Diagnostic Question Generator, a system that checks reader comprehension of a given text. The authors do not use a learner corpus as they do not create a system for testing second-language learners' proficiency level. The suggested program inserts gaps at the end of paragraphs and works with sentences with length more than four words. The authors suggest three types of possible choice distracters: ungrammatical (implementation of

which would make the whole sentence ungrammatical), nonsensical (belonging to the same part of speech as the correct answer but making the whole sentence meaningless), and plausible (meaningful for sentence in isolation but not in context).

1. Corpus description

Russian Error-Annotated Learner English Corpus (REALEC) is a collection of English texts written by HSE students learning English, whose first language is Russian. It was set up at the School of Linguistics (HSE) and is freely available at http://realec.org/index.xhtml#/exam. The corpus is constantly updated with new essays written by HSE students in their English classes and while taking their English examination at the end of their second year at the university. It consists of 3,353 essays. Students who are involved in annotation practice also take part in various research projects and becoming familiar with the most common error patterns has the additional advantage of helping them to avoid similar mistakes in their own writing. A research group at the HSE oversees the research done on the basis of the corpus and for the corpus, and academic works have been written by the research group members using the corpus material (Kuzmenko, Kutuzov, 2014; Lyashevskaya, Vinogradova, Panteleeva, 2017; Vinogradova, Gerasimenko, 2017).

By October 2017, the length of the corpus was about 765,000 tokens. In calculations cited further we use the determination of a token as a 'sequence, that includes only alphabetic symbols, a hyphen or apostrophe'. The total number of annotation tags is 52,465.

The corpus uses BRAT (brat rapid annotation tool). This annotation system tokenizes uploaded texts and allows users to attach error tags and correction notes to selected fragments. The system of error tagging is based on a system devised by the research group members. This system, described in [Kuzmenko, Kutuzov, 2014], is a tree-form hierarchy and classifies possible mistakes into a set of categories. All annotations then are stored in special text files which can be downloaded from the corpus web page. The open text (and non-byte) format of the annotations allows programmers and analysts to easily work with them without requiring the installation of any specialized libraries or software packages.

2. Methods and strategies

The use of error annotations in the corpus simplifies the task of generating exercises, as there is no need for finding sentences whose parts will be gapped as in [Aldabe Arregi, 2011] and [Mostow, Jang, 2012] – only items tagged in BRAT as having mistakes will be gapped in our design.

The first part of the research is to define the contexts where students are most likely to make mistakes. These contexts were outlined from the *Advanced Grammar in Use Activities*[4] during previous work and resulted in formation of a list of cases to be programmed.

The five cases where the automatic generator of multiple choice questions is expected to work are the following:

[4] https://itunes.apple.com/de/app/advanced-grammar-in-use-activities/id436944159?l=en&mt=8

1. For a mistake tagged *Defining relative clause,* if the correction has a progressive form of the verb, the second option can be formed with the simple form of the same verb, and the third, with the passive present participle (being + 3rd form of the verb).
2. If the predicate is assigned *Agreement-number* tag, and the subject right before the predicate is a pronoun form the following list: some/someone/somebody/one/everyone/everybody/noone/no-one/nobody/something/everything/nothing, one option is the erroneous form of the verb, the second is the correction, and the third and the fourth are the negative forms – one in singular, and the other in plural.
3. For errors tagged *Prepositional noun/Prepositional adjective/Prepositional adverb/Prepositional verb* two other options are the same noun/adjective/adverb with prepositions *at* and *for*, and if the correction is *at* or *for*, then the fourth option is *on*.
4. If any one of the three expressions – *even if, even though, even* – is tagged with *Conjunction* or *Concession conjunction*, the other two are to be the two options for the testing question. If the correction does not coincide with the expressions on this list, then it will make the fourth option.
5. For errors tagged with *Choice of tense in conditionals* or *Incoherent tenses in conditionals*, if the correction has the word *would*, the four options will be the following: Past Simple of the verb – would+1st from of the verb, Past perfect of the verb – would+have+3rd for of the verb.

2.1. Software architecture

The software code written for this research[5] is implemented within the code of the REALEC English test maker[6], which is a program written as a part of work undertaken by the research group REALEC FOR REAL WORDS[7]. This program introducesgaps instead of the parts of the text which have a defined error tag and finds correction notes for them in the annotation files. Parts of these sentences with other tags are corrected automatically. The principles of the structure and work of this program are described in [Vinogradova, Gerasimenko, 2017]. Our code takes the sentence, the correct answer and the wrong answer as arguments and returns a list with two other options.

To perform the action for the **first** case, we need to create a tool for the automatic generation of verb forms. A module written by us to perform this specific task is named verb_forms_finder. It operates with a json-type database of derived and inflected forms of English words which was gathered previously by members of the research group using data from the British National Corpus. The module uses the following diagnostics to define if the analysed lexeme is a verb:

1. Does the lexeme have a gerund-like form?

2. Does the lexeme have a 3SG(V-s)-like form?

3. Does the lexeme have a PAST(V-d)-like form?

The conjunction of these three conditions is considered necessary and sufficient for postulating that the analysed lexeme is a verb. If a word does not have forms of type 2 and/or type 3, then the colon-separated text file containing irregular verb forms is checked.

[5] Available at https://github.com/nicklogin/realec_multiple_choice
[6] Available at https://github.com/kategerasimenko/realec-exercises/blob/master/realec_grammar_exercises.py
[7] https://ling.hse.ru/realec/

The algorithm that performs this task is featured in the MontyLingua[8] Python module, written at MIT in 2002-2004, but it is not compatible with Python 3, in which our software package was written. The same compatibility problem occurs with the NodeBox English Linguistics library[9]. Another reason for writing an independent solution was that we wanted the verb paradigm browsing module to be easy to use and not to be a black box inaccessible for future enhancements and improvements.

In the **second** case, an algorithm generating the negation of the verb is needed. Such an algorithm is presented by neg() function in the find_verb_forms module. As the first verb on the left is usually the head of the verb phrase in English, it is chosen as the default locus of negation. If the head verb form is one of the elements of the closed set (*'must', 'should', 'is', 'was', 'were', 'are', 'can', 'could', 'should', 'would', 'have', 'has', had'*), then the negation is formed simply by adding the '*n't*' suffix. If the head verb form is not in this set, then the negation is formed by taking the bare infinitive form of the verb (which is performed by find_verb_forms function in the verb_forms_finder module). Then, depending on the original form of the head verb '*doesn't', 'don't'* or *'didn't*' is added to the left of it. If there is only one verb form in the phrase and it is '*have', 'had*,' or '*has*,' then the same rule is applied to the head verb. To avoid cases when a noun is taken for a verb because their forms are homonymous, corrected sentences are analysed with the Stanford Part of Speech tagger[10]. Unfortunately, it can sometimes attach non-verb tags to the verb forms, but it helps to avoid processing noun forms with the neg() function.

In the **third** case the preposition is found in the correction and is later substituted with one of the options predefined in the case description. The usage of the Stanford Part of Speech tagger would be redundant in this task because instead of attaching a general preposition tag to the word, it may attach one of the functional tags (such as the 'IN' or 'TO' tags) to the preposition. English prepositions are listed in the file attached to the program code. Having found a token that it is similar to one of the elements on the prepositions list, we substitute it with '*at', 'for,'* and *'on*' in the generated options. Code sections which work in the third and **fourth** cases are written the same way, except for the fact that elements needing replacement belong to a more limited set.

In the **fifth** case the first thing to do is to find the word *would* (the head of a *would*-phrase). After that the program continues scanning the correction and finds verb forms in the phrase. Either the last right verb (in case of the active voice) or '*be*' + the last right verb (in case of the passive voice) is written in the lex_verb variable. If *would* is under negation in the phrase, then the neg variable is getting True value. Then the options for the verb are formed following the listed models: '*would'* + 2nd form of the verb, '*would have*' + 3rd form of the verb and '*would*' + 1st form. If the phrase is in the passive voice, then variants are formed with these models: '*was/were*'+3rd form of the verb, '*would have been*'+ 3rd form of the verb and '*would be*'+3rd form of the verb. Word forms are taken from the Python dictionary object, which is returned as an output of the function find_verb_forms from the verb_forms_finder module. Then the Boolean value neg is checked. If it equals 'True', then all the generated forms undergo the neg() function of the verb_forms_finder module and are written to the multiple choice question options list.

[8] http://alumni.media.mit.edu/~hugo/montylingua/
[9] https://www.nodebox.net/code/index.php/Linguistics
[10] https://nlp.stanford.edu/software/tagger.shtml

Test and assignments based on the corpus material are located at http://web-corpora.net/realec, a learning resource based on the Moodle web engine.

2.2. Administration of the test

The program generates questions for the five cases. If there are not enough generated questions of any type then the corpus files are analysed to define the cause of such a result and see whether a hypothesis formed for each case is proved to be true. The generated questions are uploaded to the server where language experts check the appropriateness of each question. Inappropriate questions for use in the automatic generation can fall under one or more of these definitions:

1. There are some mistakes not corrected by annotators.

2. The annotator's suggested correction is mistaken, as a result, all options in the testing question are incorrect.

These two factors lead to the situation when the full automatization of the test-generating process is replaced with semi-automated generation, which includes manual editing. However, this still has advantages over the manual creation of multiple choices as it includes the existing contexts for potential mistakes from the student work in the corpus.

3. Analysis of generated exercises

The realec_grammar_exercises computer program performs the following actions:

- Opens a raw text file
- Scans the annotated text for error tags signified by the user
- Adds corrections to the raw texts for future use as one of the options
- Replaces mistakes of the specified type with gaps and corrects all the other mistakes as the annotator instructed
- Saves generated contexts in files of TXT and Moodle XML format.

If multiple mistakes of the same type are found in the same context, only one of them is gapped.

We tested the program on two versions of the corpus, one dated from 30.05.2017 and one from 05.11.2017. With the first version 39 agreement-number, 39 prepositional and 5 conditional questions were generated. For the newer version there were generated 61, 64 and 16 questions of each type respectively. Examples of the exercises of these types are presented in Figures 1-3. These questions are opened in the Moodle web engine on the server where REALEC testing is administered (http://web-corpora.net/realec).

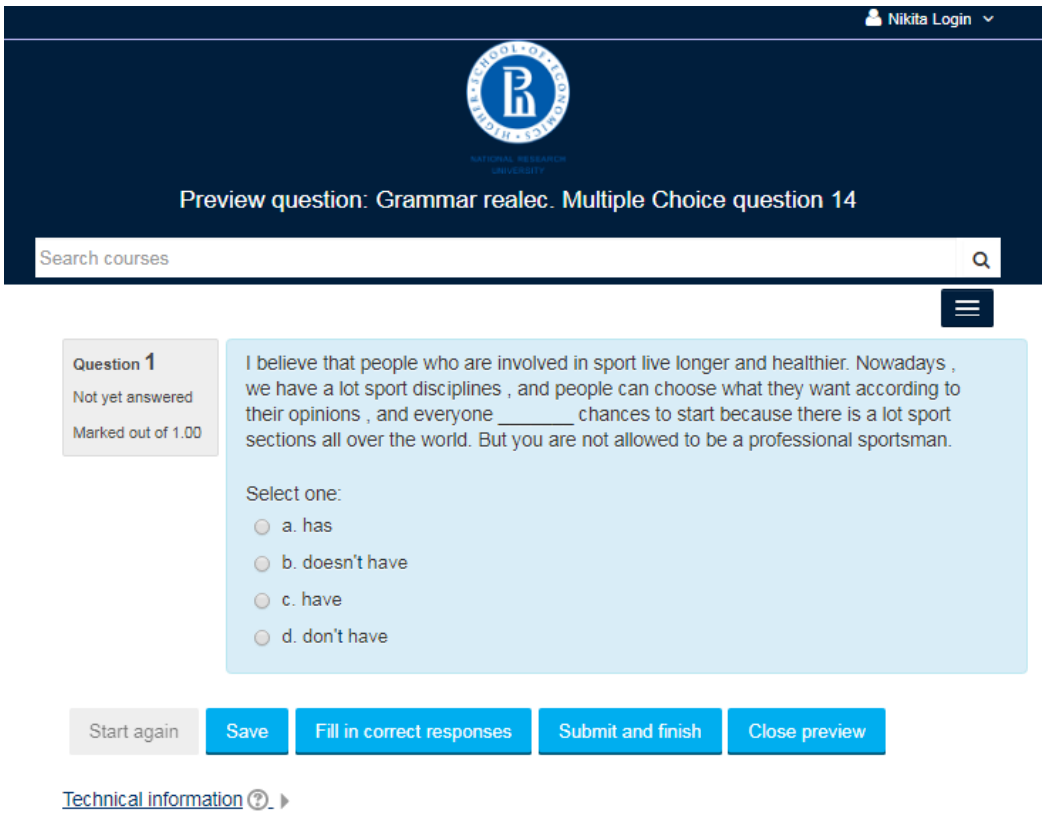

Figure 1. *An example of automatically-generated question testing the area of agreement*

Questions of the type shown in Figure 1 test the ability of students to compare information from the context of the sentence to that the given in answer options. This task demands an understanding of not only the number characteristics of the subject but also of the connections between sentences.

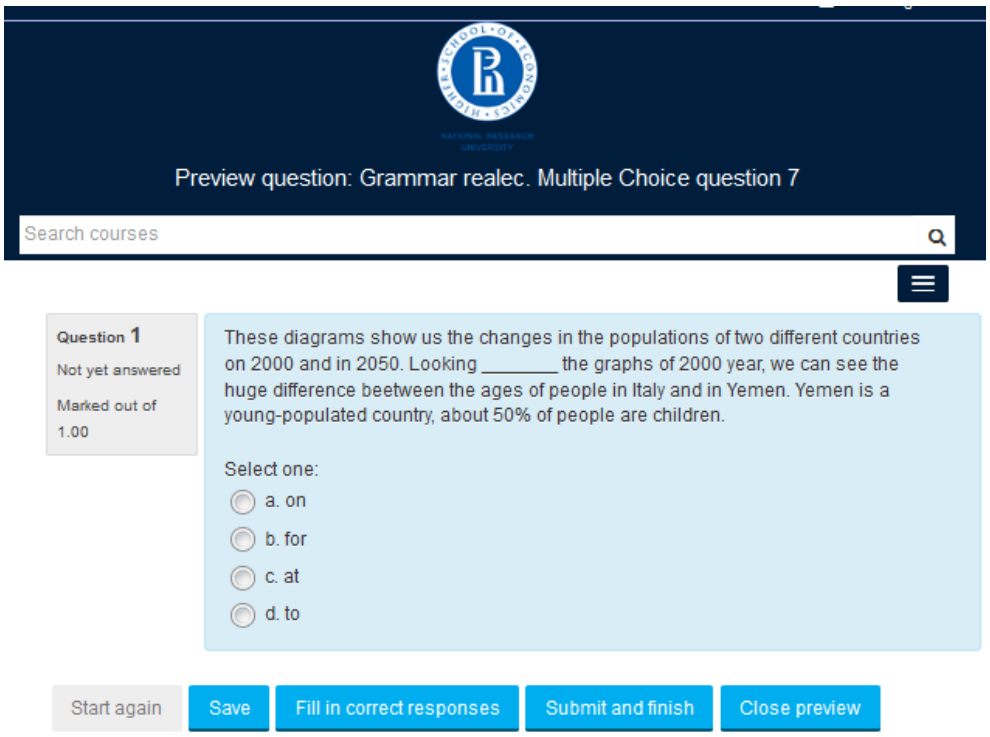

Figure 2. *An example of automatically-generated question testing the area of preposition choice*

Questions of this type contain locative prepositions '*at', 'for', 'on*' which are misused by English learners. Extracting data from the annotation files shows that mistakes in the use of these prepositions make up 15% of all prepositional mistakes (*prepositional nouns, prepositional adjectives, prepositional verbs, prepositional adverbs).*

Table 1. *The number of preposition misuse occurrences in REALEC texts*

of	for	to	on	with	from	in	about	by	at	up
26	21	20	12	10	10	9	4	2	2	1

Table 1 indicates that the preposition '*for*' is the second most frequently misused preposition. However, in the majority of the cases of tags dealing with prepositions (103 out of 220) the mistake was the absence of a preposition.

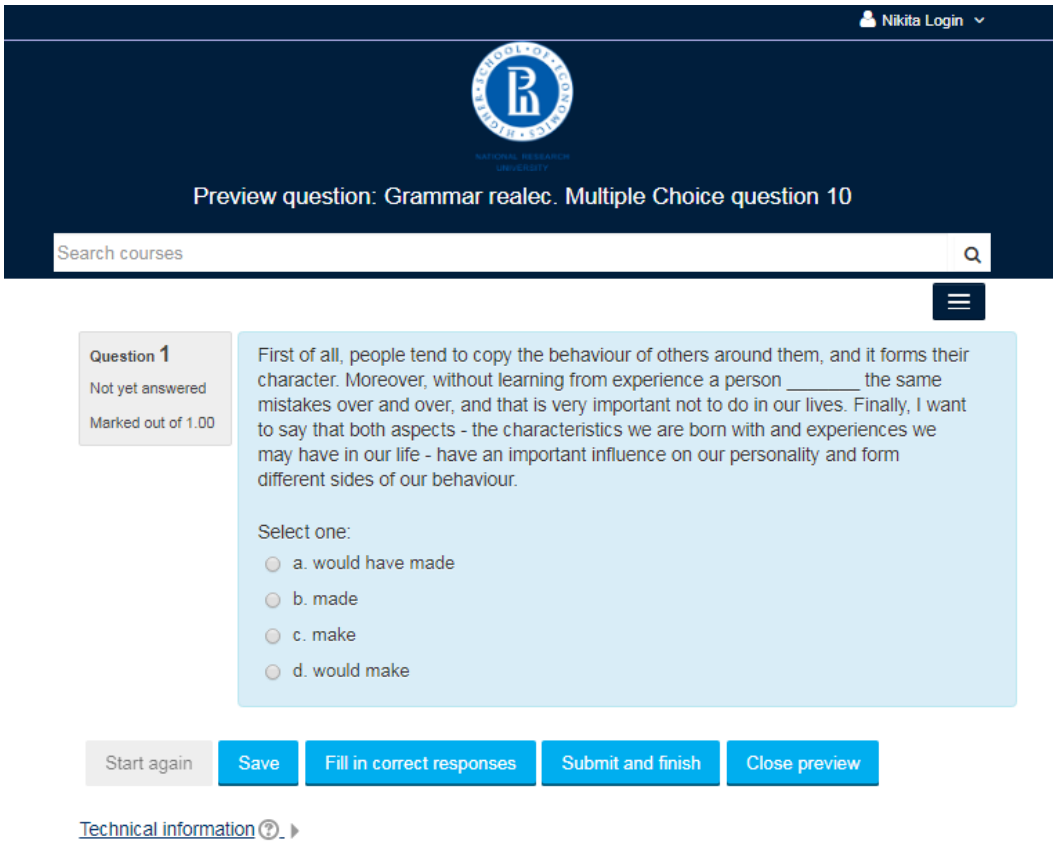

Figure 3. *An example of automatically-generated question testing the area of conditionals*

The conjunctional constructions we were looking for are *even, even if, even though, except, besides,* and *but for*. These constructions are featured in 642 essays, which make nearly 19,15% of texts and contain 21,35% of corpus tokens. To define the level of students who use them, we need to compare the length (in tokens) of works containing these items with the general text length in corpus, as the length is referenced as one of the characteristics of the level of written language in [Crossley MacNamara, McCarthy, 2010]. Thus, we have to separate IELTS *Task1*-type (graph description) from *Task2*-type (argumentative essay) texts, as they have different requirements for both length and genre. Otherwise, we will not be able to distinguish the impact

of language level from the impact of the assignment type. The two assignment types in the corpus account for two modes of essay length distribution (Figure 4). The graphs in Figures 4-6 have been built with the help of the interval estimation following Sturge's rule. By a sub-corpus we define a part of the corpus that follows a specified condition.

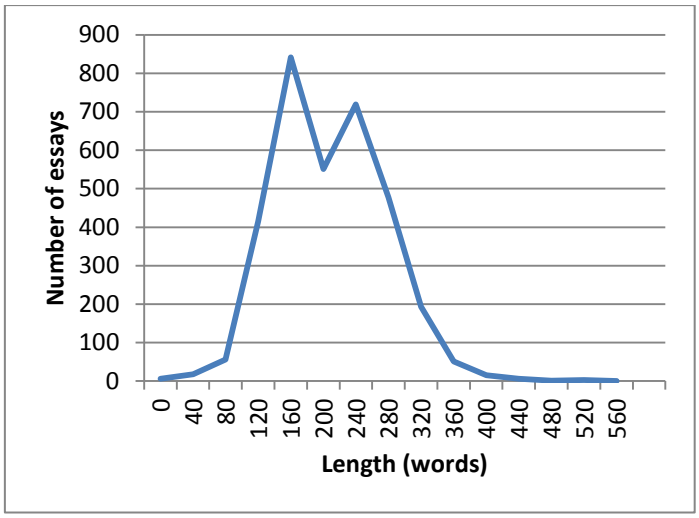

Figure 4. *Distribution of text length among the whole corpus*

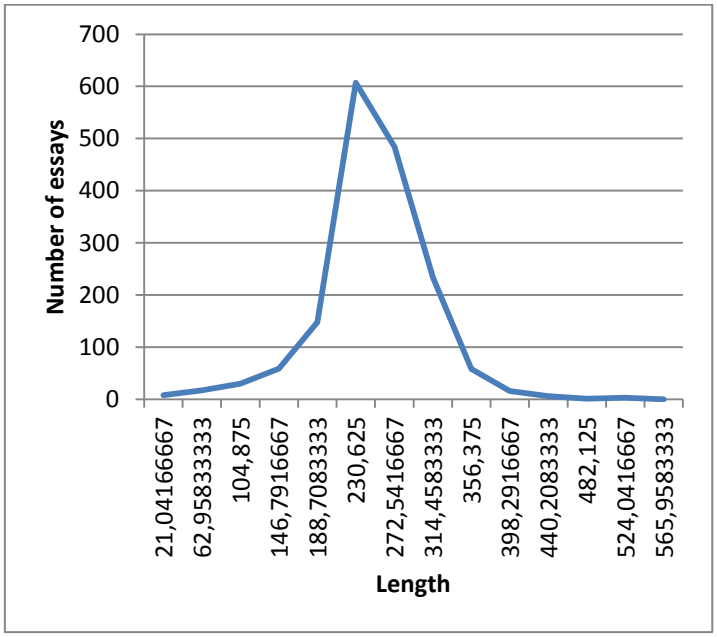

Figure 5. *Distribution of length among the argumentative essay (Task2-type) sub-corpus*

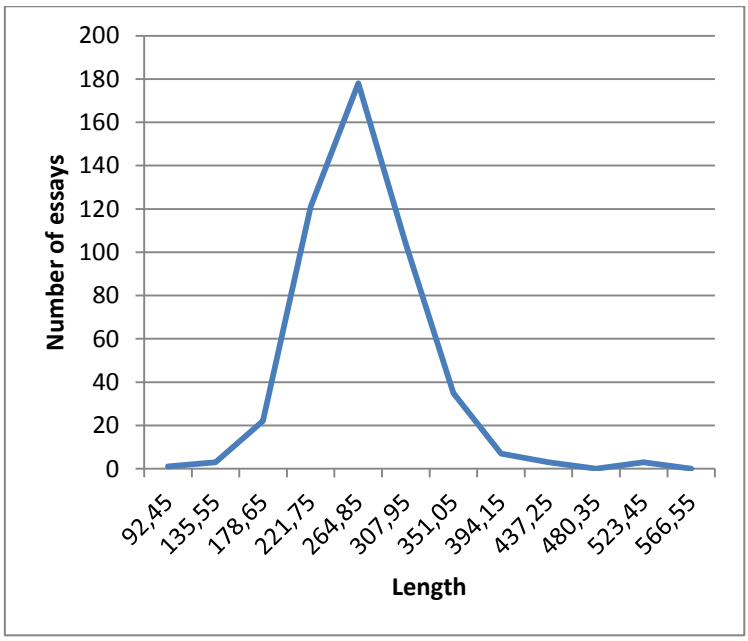

Figure 6. *Distribution of length among the essay (Task2-type) sub-corpus containing items from the list ('even', 'even if', 'even though', 'besides', 'except', 'but for')*

Among the essay sub-corpus these items are featured in 477 texts (28,6%). The distribution of the length of the essays containing these conjunctions is close to the essay length distribution in the general sub-corpus. The average length of these essays is 290,4 words, whereas in the general sub-corpus the average essay length is 271,3 words. The standard deviation of the length among all *Task2*-type essays is 57,0 words, whereas among texts containing these conjunctions it is 50,9, which means that texts containing them show less difference in the number of words. The mode length of the essay sub-corpus is 270 and the median is 271. Among the essays containing these conjunctions the mode is also 270 and the median is 284. The distribution of these conjunctions in the graph description (*Task1*-type) sub-corpus is not analysed in this paper because there are only 165 texts (9,8%) falling under the specified condition. This implies that existence of these constructions is more connected with the assignment type than with the level of student.

The word *even* occurs in 428 (10,9%) of all texts from the corpus, and the average frequency of this unit is 0,13 tokens per text. The bigram *even if* came up in 72 (2,1%) texts and occurred 0,02 times per text. The bigram *even though* is even less frequent – it occurred only in 24 (0,7%) of texts and 0,007 times per text. The word *except* was observed in 65 (1,9%) texts and 0,02 times per text. The conjunction *besides* was found in 151 (4,5%) texts and occurred 0,045 times per text. The bigram *but for* was found in 39 (1,1%) texts and occurred 0,012 times per text.

However, the special Python script written for scanning the annotation files found 78 mistakes containing items from the class. In 56 of them mistakes were made in the use of the item from the class, and 23 of them contained the mistake of not using the appropriate item from the class.

Table 2. *Distribution of mistakes among the selected conjunctions*

	even	even if	besides	except	**Total**
Spelling	1	0	1	11	13
Punctuation	1	0	1	3	5
Confusion of categories	0	0	0	1	1
Absence of detail or explanation	4	1	0	1	6
Capitalisation	0	0	1	0	1
Form of conditionals	1	0	0	0	1
Standard word order	4	0	0	0	4
Words often confused	0	0	1	0	1
Choice of a lexical item	1	0	0	0	1
Agreement - number	0	0	0	1	1
Absence of component in a sentence or clause	3	0	0	4	7
Absence of certain components in a collocation	0	1	0	0	1
Formational suffix	0	0	0	1	1
Choice of a part of lexical item	1	0	0	0	1
Tautology	1	0	0	0	1
Discourse	1	0	0	0	1
Word choice	2	0	0	1	3
Redundant component in clause or sentence	1	0	1	0	2
Word order	2	0	0	0	2
Coherence	1	0	0	0	1
Inappropriate register	1	0	0	0	1
Total	25	2	5	23	55
Percentage, %	5,567928731	2,666667	3,205128	32,85714	0,0711

Nearly 32,7% of the mistakes containing these items are spelling and punctuation mistakes which are not included in the multiple-choice exercises. The table shows that students do not usually make mistakes in the use of '*even*' (only 5,6% of all '*even'* contexts), *'even if'* (2,7%) and '*besides'* (3,2%). For the item *'except'* mistakes are in 32,9% of contexts, but nearly half of them are mistakes in the spelling of this word, and the others are distributed mostly among non-lexical error tags.

We can conclude that HSE students use conjunctions with a higher or lower frequency across the texts, and they make mistakes related to the use of conjunctions, however, the majority of mistakes do not belong to the VOCABULARY tags.

3. Analysis of the multiple-option test administration and overview of the generated exercises

The exercises generated were exported to the Moodle XML format and uploaded to http://web-corpora.net/realec and 63 questions were chosen for the testing session. The participants of the testing were 14 first- and second-year students of the School of Linguistics and the Department of Foreign Languages (HSE).

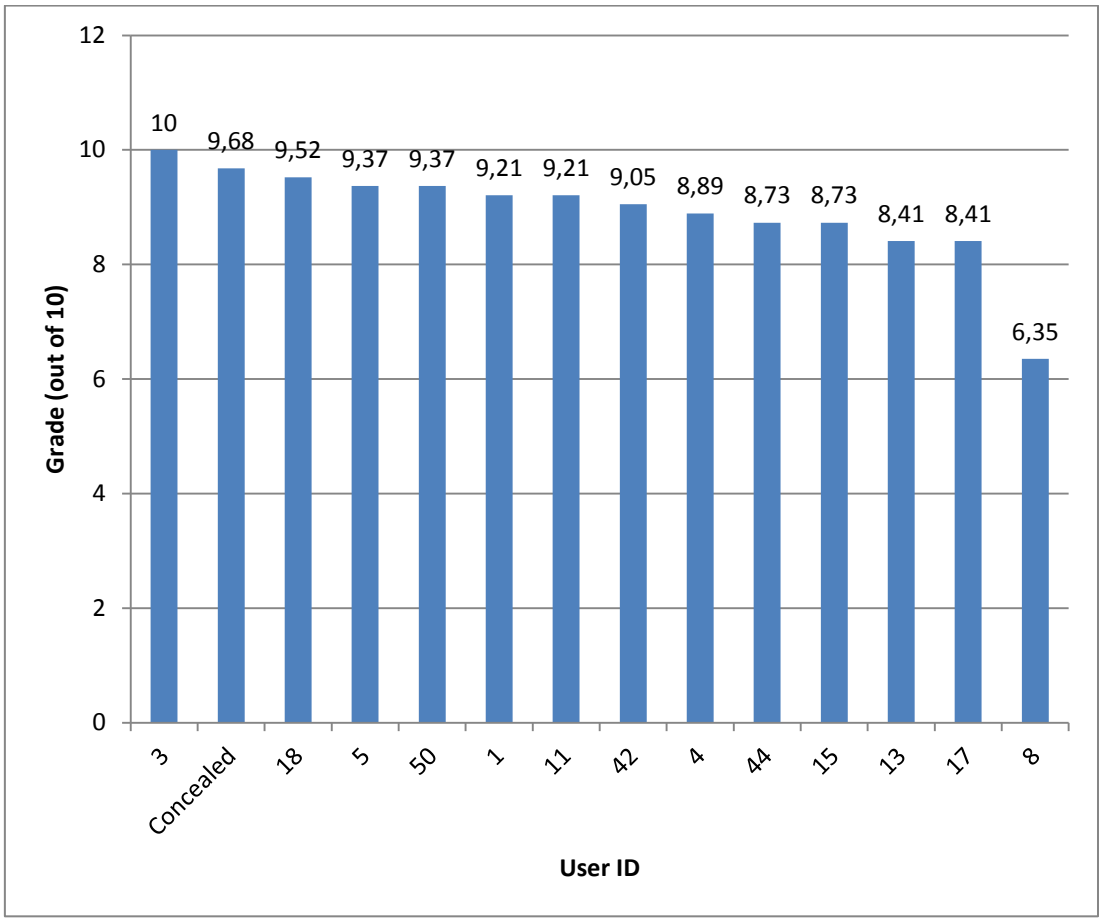

Figure 7. *Results of completing the automatically-generated test*

The students showed good results in a set of 63 question, which means that the exercises are easy enough for students – the average score is 89,2% and the standard deviation is 7%. Each question was completed by no less than 50% of participants (Figure 7). Every agreement question was completed on average by 13,7 students, whereas for conditionals and prepositional questions this measure was 12 and 11, respectively.

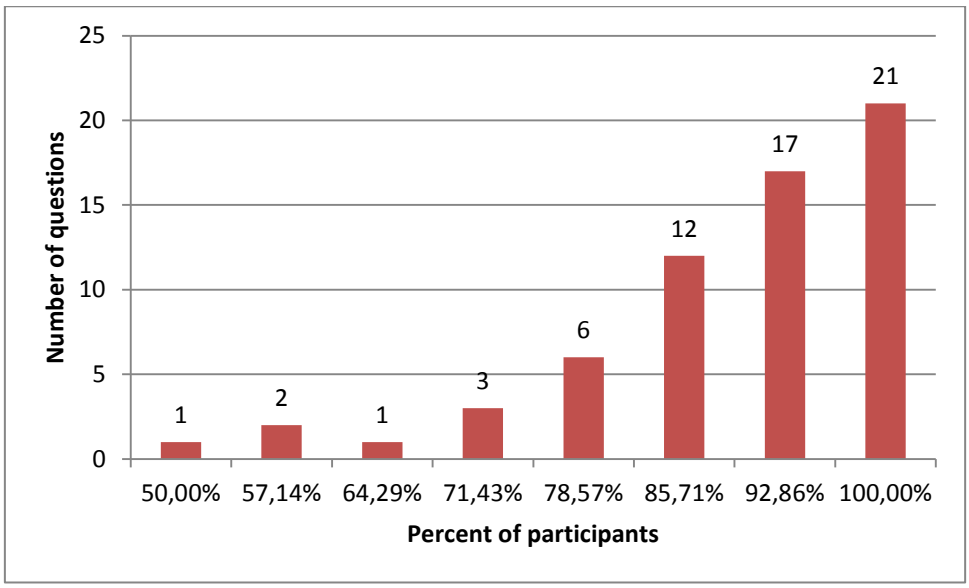

Figure 8. *Distribution of 'answerability' across the set of questions*

From here we define the hardest question as that correctly answered by the least number of students. The hardest agreement question was completed each by 12 students out of 14, and the hardest prepositional question was completed only by 7 students. The most problematic case with the misuse of prepositions occurred in the context after the verb *amount*. Among the conditionals the hardest question was correctly answered by 9 people. From 20 agreement questions, 14 (70%) were completed by all testees. Out of 31 prepositional questions 5 (16,1%) were given the right answers by all the participants, while from 11 conditional exercises only 2 (18,2%) showed 100% level of completion.

Unfortunately, there were only two results for case 1, neither of which was representative. The main reason for the lack of results for case 1 is that annotators include only '(no) comma + conjunction' sequences in the correction note – and not the whole clause – if the correction requires the deletion or addition of only one comma, namely, the one introducing the clause. At the current stage of work such occurrences were left out from the pool of possible testing questions.

Very few results that could be appropriate for use in tests were found for cases 4 and 5. To see whether HSE students use the conjunctions from cases 4 and 5 in their writing, or whether they are just unlikely to make mistakes in them, we had to carry out a statistical research, the results of which are presented in Figs. 7 and 8.

The results achieved during this experiment show not all the values of the annotation note attached to a selected error tag are equal in view of the task difficulty, even in a defined context. Thus, we may need to develop some statistical approach to estimate the distracting power of the possible multiple-choice options.

Overall, the first administration of the automatically generated questions has indicated that the level of difficulty of generated questions has to be higher.

4. Future work

The results of the multiple-choice test demonstrated in the previous section show that to make the solutions more challenging for use in real exams, we need more variety in question-forming strategies. The main ideas for the future development are the following:

1. Form a new case list for generating exercises depending on the statistical description of the collection of mistakes. Include in the cases mistakes that are most frequently made by student learners of English.

2. Use a word-vectoring tool to define lexical items that are most likely to be confused in texts of the students' essays.

3. Perform comparative testing on groups of different academic advancement and from different educational programs.

4. Implement automatic evaluation of question appropriateness.

Acknowledgement
The article was prepared within the framework of the Academic Fund Program at the National Research University Higher School of Economics (HSE) in 2016–2017(grant №16-05-0057) and the Russian Academic Excellence Project '5-100'.

References

Aldabe Arregi, I., 2011: 'Automatic exercise generation based on corpora and language processing techniques', Bilbao, Spain: Euskal Herriko Unibertsitatea, Informatika Fakultatea Lengoaia eta Sistema Informatikoak Saila.

Crossley, S.A., MacNamara, D.S., McCarthy, P.M., 2010: 'Linguistic features of writing quality', Written Communications 27(I), p. 57-86, SAGE Publications, 2010.

Kuzmenko, E., Kutuzov, A., 2014: Russian error-annotated learner English corpus: a tool

for computer-assisted language learning. NEALT Proceedings Series Vol. 22 (2014), p.87.

Liu, H., 2004: MontyLingua: An end-to-end natural language processor with common sense. Online publication at web.media.mit.edu/~hugo/montylingua.

Lyashevskaya, O., Vinogradova, O., Panteleeva, I. 2017: 'Automated student essay feedback in a learner corpus' - 'Диалог', Международная конференция по компьютерной лингвистике, т. 1, стр. 370-383, Москва

Mostow, J., Jang, H., 2012: 'Generating Diagnostic Multiple Choice Comprehension Cloze Questions', The 7th Workshop on the Innovative Use of NLP for Building Educational Applications, pages 136–146, Montreal, Canada, June 3-8, 2012.

Toutanova, K., Klein, D., Manning, C., Singer, Y., 2003: Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In Proceedings of HLT-NAACL 2003, pp. 252-259.

Vinogradova, O., Gerasimenko, E., 2017: 'Design of test-making tools for the learner corpus', Corpus Linguistics 2017 Abstracts, 2017.

Volodina, E., 2008: Corpora in Language Classroom: Reusing Stockholm Umeå Corpus in a vocabulary exercise generator, Master's Thesis, University of Gothenburg, 2008.

Voutilainen, A., 2005: 'Part-of-Speech Tagging', The Oxford Handbook of Computational Linguistics, 2005.

Olga Vinogradova
National Research University Higher School of Economics (Moscow, Russia). School of Linguistics. Associate Professor; E-mail: ovinogradova@hse.ru

Nikita Login
National Research University Higher School of Economics (Moscow, Russia). School of Linguistics. E-mail: nvlogin@edu.hse.ru