УДК 004.9

ELABORATION OF A TEST GENERATOR BASED ON A VECTORIZED SEMANTIC CLASSIFICATION

Rybkov M.V., Lichargin D.V.,
Scientific supervisor Ph.D. Lichargin D.V.
Siberian federal university

Introduction. One of important problems of the modern computational linguistics is the problem of automatic generation of the educational tests. In respect to the study of the natural and machine languages structure the problem of generating the natural language is very topical, including the problem of generating grammatically and semantically meaningful phrases and texts on natural languages, satisfying definite criteria of meaningfulness, for the purpose of contributing to the Turing test passing. The latter is topical today, because of the importance of such applied tasks as building natural-language interfaces, developing expert systems, electronic translators, electronic summarizing systems, e-learning systems, advertisement software of dialogue with the user, etc. The problem of generating educational tests is important due to wide usage of tests in education, and universities' need to have a big amount of different version of the tests of several types in order to avoid the risks of the tests to be similar and not individual enough.

In general case this problem can be expanded to an issue of semi-automatic textbook generation. Today almost all the teachers and tutors have to compose textbooks for students. And they usually have theoretical material, some tasks and methodical recommendations, but it is quite a difficult and routine process to gather all the materials together, forming original texts without involving materials breaking the copyright law and to perform text formatting. Therefore, developing semi-automatic textbook generation software, where the user can compose a textbook by adjusting a set of parameters (type of formatting, extension of output document, etc.) is a topical problem.

In this paper, it is necessary to assume that *a task* is a question with four or five, for example, possible answers, only one of which is right. And *a test* is a set of tasks of different type. The purpose of the work is to develop a test generator which is used for generating big amount of different English grammatical tasks. The basic task includes determining of the types of questions, developing a special way of generating wrong answers, creating the algorithms of meaningful speech generation based on the a specific semantic classification (Lichargin D.V., «Elaboration of a vector-based semantic classification over the words and notions of the natural language»). The novelty of the work is reduced to applying a vector based semantic classification over the words and notions of the English language for developing a test generator.

Nowadays there are a lot of different test generators that can produce quite a big amount of different English tests. But they usually use a base of ready tasks that was not generated, but prepared by people in advance. So the diversity of tests depends only on the size of this base. Algorithm of generating, for example, a test of 10 tasks includes just choice of 10 questions from the base and input to user in a random way. Besides, in every task the possible answers are mixed every time as well. A new algorithm of generating tasks that uses a vector based semantic classification over the words and notions of the English language is offered in the article.

Semantic classification over the words and notions. Let's consider a semantic classification of natural language words and notions, reduced to 16 classes of language semes (semantic, meaning «atoms»). Based on the natural language semes classification a natural language notions classification vector of five coordinates is offered. The values of the G vector coordinates are assigned by means of a generative grammar of the following form:

1. The first level of the notions classification corresponds to the coordinate G_1 of the vector G. Let G_1 = {SOMETHING, RELATION, MIND, IDEA, INFORMATION, PLACE, THING, CREATURE}.
2. The second level of the notions classification is presented by the coordinate G_2. A set G_2 of the coordinates value for the classification is assigned by a set of generative grammar rules: {S$\rightarrow$Fd, S$\rightarrow$Fx, d$\rightarrow$ALIVE, d$\rightarrow$NOT ALIVE, x $\rightarrow$ WHICH ALIVE, x $\rightarrow$ WHICH NOT ALIVE, F$\rightarrow$OF, F$\rightarrow$IN, F$\rightarrow$ON, F$\rightarrow$AT}, where notion AT means any not zero distance between objects.
3. The third level of the notions classification is determined by the coordinate G_3, G_3={X-y (*ESSENCE*), X-X-y (*ESSENCE OF ESSENCE*), ОТНОШЕНИЕ-X-y (*PROPERTY*), ОТНОШЕНИЕ-X-X-y (*CONNECTION*), ОТНОШЕНИЕ-СУЩЕСТВО-X-y (*ACTION*), ОТНОШЕНИЕ-СУЩЕСТВО-X-X-y (*JOINING*), ОТНОШЕНИЕ-СУЩЕСТВО- СУЩЕСТВО-X-y (*PRESENTING*), ОТНОШЕНИЕ-СУЩЕСТВО-СУЩЕСТВО-X-X-y (*EXCHANGE*)}, where X is any of the basic semes, determined on the first level of the classification, while y is any sequence of such semes. X is determined as the seme, main by its meaning. Sign «-» is used in the given case for concatenation notation. Essential explanations are shown in the round brackets.
4. A set of G_4 values of the coordinate G is assigned by a set of generative grammar rules: {S$\rightarrow P_1 \cdot P_2 \cdot P_3 \cdot P_4 \cdot P_5 \cdot P_6 \cdot P_7 \cdot P_8$, $P_1\rightarrow$g·QUANTITY, $P_1\rightarrow\lambda$, $P_2\rightarrow$ g·STABILITY, $P_2\rightarrow\lambda$, $P_3\rightarrow$ g·POSITIVITY, $P_3\rightarrow\lambda$, $P_4\rightarrow$ g·SPECTRUM, $P_4\rightarrow\lambda$, $P_5\rightarrow$ g·INFORMATION CONTENT, $P_5\rightarrow\lambda$, $P_6\rightarrow$ g·LOCATION, $P_6\rightarrow\lambda$, $P_7\rightarrow$ g·SIZE, $P_7\rightarrow\lambda$, $P_8\rightarrow$ g·BEING ARTIFICIAL, $P_8\rightarrow\lambda$}, where g is a linguistic scale value like: {minimal, … ,little, …, medium, …, big,…, maximal, λ}. Here λ is an empty symbol.
5. A set G_5 of the coordinate values G is assigned by a set of generative grammar rules: {S$\rightarrow$x, x$\rightarrow$(xFx), x$\rightarrow$xFx, x$\rightarrow$1 (*EXISTING*), x$\rightarrow$0 (*NON-EXISTING*), x$\rightarrow\Diamond$ (*POSSIBLE*), x$\rightarrow$□ (*NECESSARY*), F$\rightarrow$INCLUDES, F $\rightarrow$ IS INCLUDED IN, F $\rightarrow$ INCLUDES AND IS INCLUDED IN, F $\rightarrow$ PARTIALLY INCLUDES, F $\rightarrow$ MORE THAN, F $\rightarrow$ LESS THAN, F $\rightarrow$ EQUAL TO, F $\rightarrow$ SIMILAR TO, F $\rightarrow$ BECOMES, F $\rightarrow$ IS DERIVED FROM, F $\rightarrow$IS SIMULTANEOUS WITH, F $\rightarrow$ IS NOT SIMULTANEOUS WITH, F $\rightarrow$ IMPLIES, F $\rightarrow$IS DETERMINED BY, F $\rightarrow$ CORRESPONDS TO, F $\rightarrow$ IS CONNECTED WITH}.
6. All further levels of the classification are formed by means of the recursive repetition of the offered five levels of classification. The index of the level can be calculated by the formula $G_i=G_{mod(i,5)}$, where i belongs to the set of natural numbers. Any notion or class of notions for the natural language corresponds to a definite classification vector.

For example, the group of words {take, give, buy, sell, accept, present, …} correspond to the such a vector as [THING\\RELATION-CREATURE-CREATURE-X]. The group of words {shop, kiosk, supermarket, …} correspond to such a vector as [THING\IN WHICH ALIVE\X]+[THING\\RELATION-CREATURE-CREATURE-X]. The word «transport» corresponds to a vector: [THING\IN WHICH ALIVE\X]+[PLACE\\RELATION-CREATURE-X]. Each word corresponds to a set of semantic notions – points of the notions space. However, using the five coordinates of the multidimensional classification vector is a definite simplification. In the most complete form the classification can be based on 16 coordinates of a recursively repeating vector of values.

Test generator developing. According to this classification a database of words that was programmed in Delphi was created. The database has a structure of a tree of words, and it is possible to move between the branches of the tree in semantized directions, because firstly, all the notions are ordered by basic semes. And, secondly, the words are separated into topics, so this separation into semantically combinable classes is very useful with respect to test generating. The general algorithm of generating a task is mainly based on the traversal of the tree of words (the database) and creating sentence patterns, while it consists of the following sequential steps:

STEP 1. Load from the file the database of words and database of patterns into the program.
STEP 2. Select the topic that is necessary for the task generating. As a rule, the topic is chosen by user.
STEP 3. Select the type of task (for example, "choose the best word or phrase to complete the sentence", "arrange the words to make sentences", "answer the following question").
STEP 4. Select the pattern of generating phrase according to the type of task chosen at step 2.
STEP 5. Generate grammatically and semantically meaningful phrase using selected pattern.
STEP 6. Compose a task question from the phrase obtained at step 4.
STEP 7. Generate possible answers taking into account the chosen topic and the most common mistakes of students.
STEP 8. Write a task ready to be used into the file. The ready task should include description of the task, question and possible answers.

The above algorithm was carried out in the program «English test generator», where tests can be generated on more than 10 topics (Food, Clothes, Place, etc) and three types of tasks are implemented.

Task generation example. Two examples of tasks that were generated by «English test generator» are given below:

Task 1. Put the words in the right order.

my publish client plans to the book

(The right answer is «My client plans to publish the book»).
Task 2. Insert the proper word into the following sentence:

My boss ___________ *correct the price list.*

a) wants to
b) becomes to
c) amends to
d) feels to
(The right answer is "a) wants to".)

It is necessary to notice that there can be several right answers on the first question (except the latter, «The client plans to publish my book» is possible too). So in this case we face the problem of several suitable answers when it should be only one according to the test description.

When generating task 2 and similar tasks, the software selects two different topics: the one is for the question and the right answer and another (one or more) is for the wrong answers. So sometimes when these two topics are situated «far» from each other in the semantic classification tree, it is very easy to find out the right answer and the difficulty of the task is rather low. This problem can be solved by using not full database of words but its *clone* in the program. The clone of database is a reduced version of full database, where we retain only those topics that are situated «close to» each other in the semantic classification tree. The usage of database clones has shown the efficiency of this approach.

Semi-automatic textbook generation. Having developed the test generator, now the purpose is to expand its interface and create multifunctional integrated system that includes the following capabilities:

– handling the dictionary (the database, which is implemented as a tree of words);
– test generating (herewith there should be implemented generation of different types of tasks (grammatical, lexical, etc.));
– accumulating and saving all the tests and materials in standard file format, which were generated by the system or were downloaded by a user;
– operating *a textbook tree*;
– operating *a tree of text formatting*;
– generating a ready textbook by multiplying the textbook tree and the tree of text formatting (by using a special algorithm).

These capabilities can be implemented by means of imperative programming language and relational database managing system application. Although performing a presentation of a text in the form of a tree is the most complicated problem. When building a text parsing algorithm, some key features should be taken into account such as type of a text, text structure (chapter, section, paragraph), table of contents. The program module that performs a text tree parsing algorithm is supposed to have a user-friendly interface, because text book generation is a semi-automatic procedure and the usability plays very important role. All the actions such as applying a pattern to a text, creating a new chapter, adding some tasks into a section etc should be carried out. After the tree of textbook having been composed, it is necessary to perform the formatting of the text in order to all the headlines, paragraphs, references and other “parts” of the document have the needed format. This problem can be solved by creating a tree of text formatting. This tree contains formats of every element of the general text structure, i.e. every node of the tree includes specific pattern of text formatting that, for example, can be written in tags of the HTML markup language:

<font face="Arial" size = 18> <b><i> the chapter headline </i></b></font>

The tree of text formatting is fulfilled or exported by the user, in this way it will be very useful to develop some standard trees that can be downloaded and then possibly edited in the program. Obviously, the tree editor should allow changing formats using some buttons or menus, because users do not usually know HTML language. Therefore having fulfilled the tree of text formatting and the textbook tree, ready textbook generating system is reduced to the «multiplication» of these trees. During this procedure the program sets specific text format from the first tree for every element from the second tree. Thus, the user gets a ready textbook on the output of the program.

Conclusions. In the article the observation of a vector based semantic classification over the words and notions of the English language is given. It is quite effective to apply semantic vectorized classification of words and notions to build generators of semantically and grammatically meaningful phrases. Particularly, this classification was used in the real test mode to develop a test generator that can be applied by teachers of English to prepare the tests for their students. Nowadays, this program can generate tasks of only several types, so the main purpose now is to expand its possibilities in order to build a more expanded test generator. On the way to reach the purpose important factors should be taken into consideration, including classification of errors, word compatibility and patterns multiplication to expand the diversity of patterns.