Ontology Structure and Processing in the Test Generation System

Gushchanskiy, Dmitry
Saint-Petersburg State University
Saint-Petersburg, Russian Federation
e-mail: dmitry.gushchanskiy@gmail.com

ABSTRACT
Usage of tests is a popular way of gaining information about learner's knowledge, and test generation systems provide a possible method of facilitation of test process. There are different approaches to such systems, and one of them suggests the usage of special representations of knowledge and describing the systems as knowledge handlers or the whole test process as ontology-based. In this work a possible way of representing a test generation system as an ontology is described. Methods of handling of such structure are described with a focus on question selection and test score. The proposed approach uses generic ontological characteristics, therefore it is domain-independent. It provides a great variability of test variants and knowledge reuse between test cases.

Keywords
Test generation, ontology, question selection, test scoring

1. INTRODUCTION
Gaining information about learners knowledge is crucial for educational process of any kind. Today tests become the most popular method of knowledge checking. A testing approach suits many different domains on different levels of observation. Tests are applicable for both classic learning and e-learning. The test size and questions types can be changed to match a goal of testing. Thereby, it's possible to use it on different parts of education: checking residual knowledge, intermediate progress or acting like final exam.
A wide spreading of tests makes them a noticeable target for automation, which involves every side of using tests: test creation (both questions and test structure), test process and evaluation of results. One of the approaches is to use ontologies as a way to describe entities in testing and existing relations between them. The ontology allows formalizing their description and standardizing methods of testing parts interaction thus making universal approach for test processing possible. In this paper a method of representing ontology of test knowledge is introduced. The method offers a domain-independent, scalable on different levels way of knowledge representation for test systems. It supports both question generation and using predefined questions, thus making possible to use previously gained knowledge of domain tests. A short answer question type is used as a base type of question because it suits different types of knowledge checking [1] and some another question types can be represented as short answer questions. Methods of test generation and test answers scoring for this knowledge representation are also presented.

2. RELATED WORK
In [1] the idea of using ontology as a core of test generation system is described. The main characteristics of ontology-based test generation systems are shown. Moreover, the authors discussed an impact of approaches of scaling and students knowledge level on selection of scoring schemes, question types and test structure.
The previously mentioned work is developed in [2], where its ideas are extended and the full test generation system is described both as a part of a learning system and a sepa- rate system. The current work proposes a different way of ontologies organization with another view of domain-independent part of knowledge. The approach suggests unification of different domains knowledge in a single knowledge base for better usage of cross-domain relations. The domain knowledge base excludes any knowledge about learners and learning process.
The authors of [3] propose a method of generation of text-based multiple choice questions using ontologies in OWL format as the base. Usage of the widespread method of knowledge representation assists authors in simplification of ontology processing and allows the test generation system to use widely expressed relationships and properties of ontology elements for producing distractors for multiple choice questions. The domain representation described in the current paper relies more on network architecture than on document-oriented approach. Moreover, the test generation system is using a short answer open question instead of multiple choice questions proposed in [3].

3. SYSTEM DESCRIPTION
3.1 Knowledge Representation
A heterogeneous semantic network $G = (V, A)$ is chosen as a core structure for knowledge representation. V is a set of nodes, an item or a concept of the domain is associated for each. A is a set of typed arcs. They are ordered pairs of nodes with a relation type associated. It's possible to have a pair of nodes connected by several arcs with different types.
Furthermore, for additional description each node has a set of parameters. These parameters contain special information about nodes place in the domain and help to process knowledge base during test generation and answers evaluation:
• Node class t. Classes generalize items and concepts in nodes and mark nodes with something in common. They create a node classification, which is applied in test generation.
• S — natural language processing related information about the node for correct question text generation and

users answers handling. This information supports communication between the user and the system in natural language. For instance, the prototype of test generation system uses a list of synonyms as a parameter.
Moreover, for the sake of the ability to use previously created tasks and questions in tests and to express special tasks related specifically to the item each node has a set of predefined questions. Each question is represented as a number of parameters:
• t_q — a text of the question;
• A fuzzy set a which stores possible variants of an answer to the question and a membership function acting like a measure of accuracy of these variants: 1 means its the most fitting answer and 0 means it's wrong. Only the support set is needed for proper storing.
• c — a numerical characteristic of complexity of the question relative to other questions related to the node. $c \in [0; 1]$, where 1 means the hardest question. It is used for answer scoring.
• w^- — a numerical characteristic of significance of the knowledge behind the question relative to other ones connected with the node. $w^- \in [0; 1]$, where 1 means the knowledge has an absolute value. It is used for answer scoring, too.
• T — a list of tags for questions selection for test. For instance, a tag may mean that this question is suitable for entering exams or for finals, or that the question should be asked in certain courses.
Moreover, it's possible to store templates of domain-related tasks in the question set. In this case, instead of the question text a task template is stored and the fuzzy set with answer information and characteristic of complexity are replaced by descriptions of how to check and evaluate the answer and how to value complexity of the generated task. Description of such templates relates to domains and details of the system implementing and so it is omitted here.
The structure has advantages. It provides the domain independent way to store knowledge for test generation. Knowledge can be expanded by adding new nodes that describe domain details or even new domains. Thus, it is possible to keep interconnected knowledge about several domains in one place and to construct both domain-centered and inter-domain tests and only an accurate placing of tags is needed. The most of the information about specific tests is stored as a list of parameters:
• Name of the test N.
• Number of questions n_q.
• (A, v_0) — a link for a subnet of the knowledge semantic network which is going to be used in the test with allocation of a central node. The central node v_0 is the node of the knowledge semantic network representing the essential idea of test theme. It serves as an initial point during test generation.
• W_n — a list of numerical characteristics of the importance of the nodes of the selected parts of the network. The importance of node w_n may vary from test to test therefore parameters aren't stored in the knowledge base. The characteristics are also applied in test generation.$w_n \in [0; 1]$, where 0 means that there will be no questions about this item, but it can participate in question generation for another nodes; 1 means that this item is essential for the test.
• The same list W_e of parameters of importance for the arcs w_e.
• The inportance bottom limit L.
• Q_r — a list of question generation and answer evaluation rules. They will be discussed further.
• A list of tags T_N suggested for the test. The test generation system should use only previously described questions with listed tags.
• S — a scoring scheme for test results. The scheme describes how to interpret values which are received during answer checking.
Question generation rules are the core part of a test case. They determine how questions are created and processed, how their complexity and importance are calculated. As ontologies incorporate a reasoning mechanism to derive facts from explicitly refined knowledge [4], a generated question as well as a result of its answer checking may be considered as a reasoned fact. The choice of the reasoning method and its implementation strongly affects rules structure and capabilities. In the test prototype a rule-based inference engine is applied. It uses nodes and arcs as set of axioms thereby processing test question generation as proof of a theorem. Therefore an item of Q_r may be considered as a tuple (t_x, e, w_r, w_c), where t_x is a rule of generation of a question text, e — a rule of answer evaluation and w_r , c_r — rules of evaluation of question importance and complexity, correspondingly. An example of rules is shown in Implementation part of the paper.

3.2 Question Selection

The base mechanic of test generation system is to choose a question from the network, ask the question and receive an answer, check answer, repeat the process several times and finally get the score for a student. The process of question choice can be divided into two parts: selection of the node to ask about and selection or generation of a question about the node.
The node selection process is similar to applications of Monte-Carlo methods to Bayesian networks [5] in behavior:

Step 1. Start in the v_0.

Step 2. Check the network availability to ask questions: suppose n_c — current number of asked questions, if $n_c = n_q$ or $\sum_{w_n \in W_n} w_n = 0$, end the test.

Step 3. Check node availability: if $w_n \neq 0$ then try to get access to the node in the next step, else go to Step 4.

Step 4. Nodes access: w_e acts like the probability to be chosen (i.e. $w_n = 1$ means it will be chosen absolutely), and a random generated value $r_n \in [0; 1]$ is compared with it. If $r_n \leq w_n$, pick the node and end the procedure, in another case continue to the next step.

Step 5. Get a list E of all arcs which have the current node as a tail. Suppose $\bar{W}_e = \sum_{e \in E} w_e(e)$ Construct a categorical distribution with a probability mass function $f(x), f(x = e_i) = w_e(e_i)/\bar{W}_e$.

Step 6. Apply a random generated value $r_n \in [0; 1]$ to the distribution thus receive one of the arcs.

Step 7. Pick the head of the chosen arc and go to the Step 3.

The importance bottom limit L is an important part of node selection handling. After several reductions the importance of a node may become low, but still nonzero. The node has a very small chance to be selected and is

used only as intermediate connection in the network. Therefore an importance bottom limit is introduced in the system. If the importance of a node is smaller than the limit, it becomes zero thus excluding the most of the knowledge about the node except its name and type. Such reduction helps to better use test cases and allows the system to use quit condition from the node selection procedure.

The guarantee of variety of selected nodes is crucial for the test generation system. To provide it the importance parameter of the node will decrease before jumping to the next question generation. In general, reduction of the importance should be done by a contraction on $[0; 1]$ with Euclidean metric with zero as a fixed point. In the current work a function $z(x) = x/\beta$ is selected. The parameter β is selected by analysis of the network structure and the number of questions. The main criterion of selection is to make the sum of importance parameters of all nodes in the network close to 0 by the end of test. This idea produces the formula for β:

$$\beta = e^{\frac{\sum\limits_{w_n \in W_n, w_n > 0} \ln w_n - \alpha \ln L}{n_q - \alpha}}, \quad (1)$$

where α is the number of participating nodes for which it is possible to receive at least one question.

Next step is to choose a question for the node. The filtering by tags selects all appropriate previously created questions and task templates from the node. Additionally the system selects suitable for node task generation rules from the test case. During rules selections the system receives their importance parameters. They could be preliminary added to the rules or be computed on the run. Then, like in the node selection case, a categorical distribution is constructed and one question option is chosen. The question is marked as used. The text of the task is generated if needed and presented to a user who gives an answer to it. Since the test generation system has a generality, the users answer is a string and it's needed to be checked.

3.3 Answer Checking

Short answer evaluation is a complex study which seizes several scientific areas with different methods of evaluation [6]. In this work a simple comparison with templates is used for predefined questions. It requires a manual explanation of answers, but guarantee stable and correct work of the system. Therefore, the suitable options are sought in the fuzzy set attached to the question. The best result is returned as an answer evaluation. In the case the answer isn't in the set, 0 is returned. If the answer is for the question generated by the test case rules, the rule e from its tuple is applied and the numerical score in range $[0; 1]$ is received.

3.4 Test Scoring

For each answered question four parameters received: answer checking result a, question importance $\bar{w}$, question complexity c and node importance w_n. Every parameter is in range $[0; 1]$. The key idea is to create a function $F(a, \bar{w}, c, w_n)$ which can evaluate the answer investment in the total result. F has to satisfy a list of requirements:

1. Completely wrong answer can't receive a positive result:
$$F(0, \bar{w}, c, w_n) = 0 \; \forall \bar{w}, c, w_n. \quad (2)$$
2. A right answer for an elementary question still should be rewarded:
$$F(a, \bar{w}, 0, w_n) \neq 0 \; \forall \bar{w}, w_n; \; a > 0. \quad (3)$$
3. The best result can be achieved only with the top conditions:
$$F(1, 1, 1, 1) = 1. \quad (4)$$
4. More precise answer is better:
$$\forall a_1, a_2; a_1 < a_2 : \; F(a_1, \bar{w}, c, w_n) < \\ < F(a_2, \bar{w}, c, w_n) \; \forall \bar{w}, c, w_n. \quad (5)$$
5. More difficult question is more rewarded:
$$\forall c_1, c_2; c_1 < c_2 : \; F(a, \bar{w}, c_1, w_n) < \\ < F(a, \bar{w}, c_2, w_n) \; \forall a, \bar{w}, w_n. \quad (6)$$
6. More important question is more rewarded
$$\forall \bar{w}_1, \bar{w}_2; \bar{w}_1 < \bar{w}_2 : \; F(a, \bar{w}_1, c, w_n) < \\ < F(a, \bar{w}_2, c, w_n) \; \forall a, c, w_n. \quad (7)$$
7. A question for more important item is more rewarded:
$$\forall \bar{w}_{n1}, \bar{w}_{n2}; \bar{w}_{n1} < \bar{w}_{n2} : \; F(a, \bar{w}, c, w_{n1}) < \\ < F(a, \bar{w}, c, w_{n2}) \; \forall a, \bar{w}, c. \quad (8)$$

An infinite set of functions satisfies the requirements, for instance, a family of functions

$$F(a, \bar{w}, c, w_n) = w_n^{n_1} \cdot a^{n_2} \cdot e^{\bar{w}^{n_3}} \cdot e^{c^{n_4}}, \\ n_1, n_2, n_3, n_4 > 0. \quad (9)$$

is suitable for scoring.

Then $F(a, w^-, c, w_n)$ is chosen it is possible to find a total score for the test. Let S_t be a list tuples (a, w^-, c, w_n) from each answer evaluation. Then a total score $Ts, Tsin[0; 1]$ may be found as

$$Ts = \frac{\sum\limits_{(a, \bar{w}, c, w_n) \in S_t} F(a, \bar{w}, c, w_n)}{\sum\limits_{(a, \bar{w}, c, w_n) \in S_t} F(1, \bar{w}, c, w_n)}, \quad (10)$$

and interpreted according to the test description. The current approach has an important disadvantage which can restrict the test evaluation. For instance, let the domain have an item with a question which is considered as quiet simple, but crucial for the whole test theme. It's natural to suppose that the right answer should not be high evaluated due to simplicity, however, an incorrect answer should greatly affect total score, thus impeach the test results.

As a possible solution the usage of additional score is suggested. In addition to $F(a, w^-, c, w_n)$ a penalty function $K(a, w^-, c, w_n)$ is added. The total score formula is modified as follows:

$$Ts = \frac{\sum\limits_{(a, \bar{w}, c, w_n) \in S_t} F(a, \bar{w}, c, w_n)}{\sum\limits_{(a, \bar{w}, c, w_n) \in S_t} (F(1, \bar{w}, c, w_n) + K(a, \bar{w}, c, w_n))}. \quad (11)$$

$K(a, \bar{w}, c, w_n)$ has to satisfy some requirements:

1. A penalty may have no upper bound:
$$K(a, \bar{w}, c, w_n) \geq 0 \; \forall a, \bar{w}, c, w_n. \quad (12)$$
2. An absolutely correct asnwer has no penalty:
$$K(1, \bar{w}, c, w_n) = 0 \; \forall \bar{w}, c, w_n. \quad (13)$$

3. More precise answer has a lesser penalty:

$$\forall a_1, a_2; a_1 < a_2 : K(a_1, \bar{w}, c, w_n) \geq \geq K(a_2, \bar{w}, c, w_n) \, \forall \bar{w}, c, w_n. \quad (14)$$

4. More difficult question has a lesser penalty:

$$\forall c_1, c_2; c_1 < c_2 : K(a, \bar{w}, c_1, w_n) \geq \geq K(a, \bar{w}, c_2, w_n) \, \forall a, \bar{w}, w_n. \quad (15)$$

5. More important question is more fined:

$$\forall \bar{w}_1, \bar{w}_2; \bar{w}_1 < \bar{w}_2 : K(a, \bar{w}_1, c, w_n) \leq \leq K(a, \bar{w}_2, c, w_n) \, \forall a, c, w_n. \quad (16)$$

6. A question for more important item is more fined:

$$\forall \bar{w}_{n1}, \bar{w}_{n2}; \bar{w}_{n1} < \bar{w}_{n2} : K(a, \bar{w}, c, w_{n1}) \leq \leq K(a, \bar{w}, c, w_{n2}) \, \forall a, \bar{w}, c. \quad (17)$$

As for $F(a, \bar{w}, c, w_n)$, an infinite set of functions satisfies the requirements. The function family:

$$K(a, \bar{w}, c, w_n) = w_n^{n_1} \cdot (1-a)^{n_2} \cdot \cdot \bar{w}^{n_3} \cdot (1-c)^{n_4}, \quad n_1, n_2, n_3, n_4 > 0 \quad (18)$$

is an example of such functions.

4. IMPLEMENTATION

A prototype of the test generation system is implemented using Python with Pyke [7] as a knowledge engine for question generation. Pyke natively supports multiple rule bases with forward-chaining and backward-chaining rules and code snippets within rules thus allows the system to express complex relationships between items in different domains.

As a base for test generation the knowledge base about geography of Ireland is used. The knowledge base is a semantic network with 74 nodes and 127 vertices, 13 nodes have predefined questions. The test case has 15 questions, 14 rules of question generation in Q_r and a score scheme described above with

$$F(a, \bar{w}, c, w_n) = a \cdot w_n \cdot e^{\bar{w}-1} \cdot e^{\sqrt{c}-1}, \quad K(a, \bar{w}, c, w_n) = w_n \cdot (1-a) \cdot \bar{w} \cdot (1-c)^2. \quad (19)$$

Question generation rules use similarities in the network structure and allow for a variety of options of the same type of questions. For example, such diversity provides more than a thousand different test variants for a test with 15 questions, and each of the variants has questions from different parts of the domain.

Each element of Q_r is described as follows. (t_x, e, w_r , w_c) t_x is a pair of text generation template and a rule for an applicability the question to a node and receiving parameters from the node. e is a rule for answer checking, which substitutes answer in the Pyke rule and tries to resolve it. w_r is similar to the node significance and w_c = $1-w_r$. For example, a question: "What county is Ennis situated in?" is generated by the rule:

situated_in($what, $where), $type = $where.type

and a text template:

What $type is $what situated in?

For answer checking a rule

situated_in($what, $x), $x.type = $type is applied. It gives 1 as a score if the rule application is finished cor- rectly and 0 otherwise.

5. CONCLUSION AND FURTHER RESEARCH

The ontology representation of a test generation sys- tem was presented. The representation allows storage of domain independent knowledge in the same connected network with applying different test cases, where each of them may use different parts of the network and have different test-oriented purposes. Some methods of handling such representation were presented. The method of question selection provides the question variability and coverage of a test case. The method of complex answer scoring was also described. It showed an appropriate behavior during the test case runs. The test generation system prototype was created. It shows the ability to diversify test varieties of the test case. Rule-based question generation creates a variability of tests, each test is quite unique.

The research suggests many ways of its development, but some options are considered. Firstly, the test scoring method should be compared with other methods in larger test case runs. Different kinds of functions should be examined during the tests. Secondly, the method of question selection has a lack of user feedback, which can vary test even more. Possible ways to add a feedback in the method behavior should be considered. Finally, effective knowledge storage and processing should be examined. In the case of multi-domain knowledge the number of items in the network can be increased significantly and effective knowledge handling becomes crucial for the system.

REFERENCES

[1] Soldatova, Larisa, Mizoguchi, Riichiro, "Ontology of test", *Proc. Computers and Advanced Technology in Education*, pp. 173–180, 2003.

[2] Soldatova, Larisa, Mizoguchi, Riichiro, "An Ontology-Based Test Generation System" *Semantic Web Technologies for E-Learning*, pp. 96–110, 2009.

[3] Papasalouros, Andreas, Kotis, Konstantinos, Konstantinos, Kanaris, "Automatic generation of tests from domain and multimedia ontologies", ıInteractive Learning Environments 19, no. 1 ,pp. 5–23, 2011.

[4] McGuinness, Debora L., Patel-Schneider, Peter F., "From Description Logic Provers to Knowledge Representation Systems", *The description logic handbook: theory, implementation, and applications*, pp.271–283, 2003.

[5] Heckerman, David, "A Tutorial on Learning with Bayesian Networks", *Innovations in Bayesian Networks. Studies in Computational Intelligence Volume 156*, pp. 33–82, 2008.

[6] Ziai, Ramon, Ott, Niels, Meures, Detmar, "Short Answer Assessment: Establishing Links Between Research Strands", *The 7th Workshop on the Innovative Use of NLP for Building Educational Applications*, pp. 190–200, 2012.

[7] Frederiksen, Bruce, "Python Knowledge Engine", *http://pyke.sourceforge.net.*