RESEARCH PROJECT

Ontology-Based Multiple Choice Question Generation

Tahani Alsubait[1] · Bijan Parsia[1] · Ulrike Sattler[1]

Received: 15 May 2015 / Accepted: 5 October 2015 / Published online: 17 November 2015

Abstract Multiple choice questions (MCQs) are considered highly useful (being easy to take or mark) but quite difficult to create and large numbers are needed to form valid exams and associated practice materials. The idea of re-using an existing ontology to generate MCQs almost suggests itself and has been explored in various projects. In this project, we are applying suitable educational theory regarding assessments and related methods for their evaluation to ontology-based MCQ generation. In particular, we investigate whether we can measure the similarity of the concepts in an ontology with sufficient reliability so that this measure can be used to control the difficulty of the MCQs generated. In this report, we provide an overview of the background to this research, and describe the main steps taken and insights gained.

1 Introduction

Description logics (DLs) [4] are well-understood logics that form the logical basis of the web ontology language OWL. As a consequence of OWL's standardisation over the past 15 years and the related increase in tool support (e.g., reasoners, editors/IDEs, APIs), loads of people/communities have developed ontologies, i.e., DL knowledge bases that capture some domain of interest, in particular in biology and clinical sciences. Some of these ontologies have been developed by groups of experts over a long time, and so can be expected to provide a shared view of a broad, complex domain with a high level of detail (e.g., NCIt[1]).

Given these observations, a natural question arising is whether we cannot reuse these "nice" representations of domain knowledge for *teaching* in these domains, in particular for generating assessments. Of particular interest for this project are *multiple choice questions* (MCQs): they are well-suited for various kinds of questions, in particular those evaluating the students' understanding and knowledge of (the terms in) a subject area, automatically gradeable, and can be implemented in eLearning tools that give students immediate feedback.

MCQs are, however, rather difficult to design [9], regardless of whether we design them by hand or automatically: in addition to a good question[2] and a correct answer,[3] we need to generate suitable *distractors*, i.e., reasonable yet incorrect alternative answers. To generate a good *set* of MCQs, i.e., an *exam*, we need to control the *difficulty* of MCQs: a good exam has to have questions from a range of difficulties so that it faithfully assesses students' understanding. That is, only when an exam has questions of low, medium, and high difficulty can we use it to reliably identify students with a good understanding of the subject from those with a medium or weak one [13].

A large body of research exists on automatic question generation approaches from different types of knowledge sources, including ontologies. We carried out a systematic review of such approaches to gain a better understanding of the current state-of-the-art in question generation. Details of this systematic review can be found in [2]. An important deal of research effort has been devoted to

✉ Ulrike Sattler
ulrike.sattler@manchester.ac.uk

[1] School of Computer Science, University of Manchester, Manchester, UK

[1] http://ncit.nci.nih.gov.

[2] Referred to as the stem in MCQ terminology.

[3] Referred to as the key in MCQ terminology.

improve the distractor generation mechanism. Early MCQ generation approaches generate distractors based on syntactic or lexical features (e.g., same part of speech, same frequency, derivative words of the same prefix or suffix). While such mechanisms can be suitable for generating distractors for language testing, it is clearly not always suitable for other domains. Thus, an interest in semantics-based distractor generation mechanisms has developed, see for example [3, 6, 7, 14, 15]. These approaches utilise some notion of concept similarity to select suitable distractors just as we suggest in the current project, however, on the one hand, they do not examine the correlation between their utilised notion of similarity and human judgements of similarity (since they will be answering the generated questions). On the other hand, they do not examine the impact of varying the similarity between answers on the difficulty of the generated questions. We address the first issue in [1] and to address the second issue we conjecture that,

> given a suitable ontology and a suitable similarity measure on concepts, we can generate MCQs whose difficulty we can control by varying the similarity between the distractors and the key.

This conjecture involves two parameters: the suitability of

- a similarity measure: is it precise enough (and can it be implemented in a sufficiently performant way) so that we can base our distractor selection on it?
- an ontology: does it capture enough domain knowledge to base our MCQ generation on it? In particular, is the ontology detailed enough so that we can gauge concepts' commonalities and differences which, in turn, is a requirement for estimating similarity between concepts?

2 Preliminaries

We assume the reader to be familiar with DLs [4]; for those who are not, it should suffice to know that DLs are decidable fragments of first order logic with unary predicates, called *concepts* and binary predicates, called *roles*. An *ontology* is a finite set of axioms, e.g., of the form:

$$\begin{array}{lll} \mathit{Orangutan} & \sqsubseteq & \mathit{Frugivore} \sqcap \exists \mathit{livesIn}.\mathit{Forest} \\ \mathit{Koala} & \sqsubseteq & \mathit{Herbivore} \sqcap \exists \mathit{livesIn}.\mathit{WoodLand} \\ \mathit{Giraffe} & \sqsubseteq & \mathit{Herbivore} \sqcap \exists \mathit{eats}.\mathit{Tree} \\ \mathit{Frugivore} & \sqsubseteq & \mathit{Herbivore} \\ \mathit{WoodLand} & \sqsubseteq & \mathit{Forest} \end{array}$$

where the first and last axioms, for example, would read in their first order logic form as follows:

$$\begin{array}{lll} \forall x.\mathit{Orangutan}(x) & \Rightarrow & (\mathit{Frugivore}(x) \wedge \exists y.\mathit{Forest}(y) \wedge \\ & & \mathit{livesIn}(x,y)) \\ \forall x.\mathit{WoodLand}(x) & \Rightarrow & (\mathit{Forest}(x)) \end{array}$$

A concept can be atomic (i.e., a predicate name) or complex (e.g., a conjunction). For this paper, the specific kind of DL does not matter; we are, however, only concerned with *TBox* axioms, i.e., terminological axioms of the form above and do not consider ABox assertions or ground facts.

The model theory and entailment relation $\models$ are classic, e.g., the example ontology above entails that $\mathit{Koala} \sqsubseteq \exists \mathit{livesIn}.\mathit{Forest}$. A DL *reasoner* implements a decision procedure for this entailment relation.

Given an ontology $\mathcal{O}$, a set of concepts $\mathcal{L}$, and a concept C. We define *the subsumers of* C *in* $\mathcal{L}$ w.r.t. $\mathcal{O}$, $\mathit{subs}(C,\mathcal{L},\mathcal{O})$ as follows:

$$\mathit{subs}(C,\mathcal{L},\mathcal{O}) = \{D \in \mathcal{L} \mid \mathcal{O} \models C \sqsubseteq D\}.$$

Throughout the paper, we use $\widetilde{\mathcal{O}}$ to refer to the signature of $\mathcal{O}$.

3 Approach and Challenges

Intuitively, one way that the difficulty of an MCQ can vary is with the *similarity* between the key and distractors. The more similar they are, the more knowledge you need to discriminate between them and thus select the correct one. When generating conceptual knowledge multiple choice questions from an ontology, stem and answers are derived from (possibly complex) concepts. Hence if we have an appropriate similarity measure for concepts, we can attempt to generate MCQs with predictable difficulty.

As an example, consider the ontology presented in Sect. 2. One of the questions that can be generated from this ontology is: "Which of the following is a Frugivore?" where the correct answer is *Orangutan*. Both *Koala* and *Giraffe* can be used as plausible distractors but, if we want to generate a difficult question, then we can pick *Koala* as a distractor and if we want an easy question, then we can pick *Giraffe* as a distractor. This is because *Koala* and *Orangutan* share more "properties" than *Giraffe* and *Orangutan* (both *Koala* and *Orangutan* live in a *Forest*). Here, we are applying similarity measures over those concepts (the key and distractors) to control the difficulty of this question.

A core challenge of the proposed approach is finding a suitable way to measure similarity of concepts in ontologies: namely one that correlates well with human judgements of similarity. Various similarity measures have been developed for DLs [8, 11, 16], and we need to find one that is both computationally feasible so that we can use it to pick distractors, and informative enough so that this choice is suitable. Due to the computational restrictions (e.g., limited expressivity, acyclic TBoxes, reliance on representative ABoxes) of existing similarity measures, we have made the choice to build a new family of similarity measures that are suitable to be used with any ontology.

To evaluate the presented approach to control the difficulty of MCQs we have designed two case studies. In both studies, we make use of purpose-built ontologies and evaluate the approach via domain experts or students. The two studies make use of three modules: (1) a similarity measurer, (2) an MCQ generator and (3) an MCQ checker.

The similarity measurer computes the similarity between (possibly complex) concepts w.r.t. an ontology $\mathcal{O}$. The similarity measurer encompasses a range of similarity measures with different granularities (i.e., semantic sensitivity) and computational costs. All measures are inspired by Jaccard's similarity coefficient and thus the similarity $Sim(C, D, \mathcal{L}, \mathcal{O})$ of two concepts C, D in a concept language $\mathcal{L}$ w.r.t. an ontology $\mathcal{O}$ is defined as follows:

$$Sim(C, D, \mathcal{L}, \mathcal{O}) = \frac{|subs(C, \mathcal{L}, \mathcal{O}) \cap subs(D, \mathcal{L}, \mathcal{O})|}{|subs(C, \mathcal{L}, \mathcal{O}) \cup subs(D, \mathcal{L}, \mathcal{O})|}.$$

The above formula would be trivialised unless we define $\mathcal{L}$ such that $subs(C, \mathcal{L}, \mathcal{O})$ is finite. For the current study we set $\mathcal{L}$ to be either $\mathcal{L}_{sub}$ or $\mathcal{L}_{gram}$. $\mathcal{L}_{sub}$ is defined as:

$$\mathcal{L}_{sub} = SubCE(\mathcal{O})$$

where $SubCE(\mathcal{O})$ is the set of (possibly complex) concept expressions in $\mathcal{O}$, and $\mathcal{L}_{gram}$ is defined as:

$$\mathcal{L}_{gram} = \{E \mid (E \in SubCE(\mathcal{O}) \vee (\exists r \in \widetilde{\mathcal{O}} \wedge \\ \exists F \in SubCE(\mathcal{O}) \cdot E = \exists r \cdot F)\}$$

We refer to the similarity measure that makes use of $\mathcal{L}_{sub}$ as $SubSim(\cdot)$ and the measure that makes use of $\mathcal{L}_{gram}$ as $GrammarSim(\cdot)$. Other options to design alternative measures are presented in [1]. The similarity measurer module computes the pairwise similarity of all concept names in $\mathcal{O}$ using $SubSim(\cdot)$ and the pairwise similarity of all concept expressions in $\mathcal{O}$ using $GrammarSim(\cdot)$. The rationality behind that is that $GrammarSim(\cdot)$ is more semantically sensitive (i.e., precise), but much more expensive than $SubSim(\cdot)$; thus we only use the former to compute the similarity of complex concepts where commonalities and differences are usually expressed at a high expressivity level. It should be noted that if computation cost is not an issue, then $GrammarSim(\cdot)$ can be used to compute the similarity of both concept names and complex concepts and if (high) semantic sensitivity is not an issue then $SubSim(\cdot)$ can be used as a (cheap) approximation of $GrammarSim(\cdot)$. A detailed comparison between the two measures (and other measures) can be found in [1]. It is also worth mentioning that the presented measures are sensitive to the syntax used to describe the concepts in the underlying ontology. Hence, if we have two equivalent ontologies ($\mathcal{O}_1 \equiv \mathcal{O}_2$) over the same signatures ($\widetilde{\mathcal{O}_1} = \widetilde{\mathcal{O}_2}$) then $Sim(C_1, C_2, \mathcal{O}_1)$ does not necessarily equal $Sim(C_1, C_2, \mathcal{O}_2)$. This syntactic-sensitivity is not an issue when computing similarity over a single ontology, as we do in the case studies presented below.

The MCQ generator takes as input an ontology and the pairwise similarities returned by the similarity measurer. It returns as output a set of MCQs taking one of the following forms: "What is X?", "Which of the following is an X?", "What is the following definition describing?", "Which of the following is the odd one out?". Each MCQ has a stem, a key, a varying number of distractors and a difficulty level (i.e., easy or difficult). The same stem and key can be used to generate two questions of different difficulty levels depending on the similarity degree between the key and the selected distractors.

Two ontologies have been built for the two studies. The first ontology, which has been built by one of the authors who is familiar with ontology development strategies and tools, models a selected part from a knowledge representation (KR) course presented to third year students at the University of Manchester. The second ontology, which has been built by an instructor who has no prior experience in building ontologies (with guidance from one of the authors), covers the conceptual part of a self-study Java course presented to Masters students at the same university. In terms of development cost, the two ontologies required 14 and 25 h (spread over multiple days) of development time, respectively. In terms of size, the KR ontology has 504 axioms, 151 classes, 7 object properties, and the Java ontology has 1151 axioms, 319 classes and 107 object properties.

The MCQ checker presents a set of MCQs to domain experts and asks them to (a) attempt to answer the questions, (b) rate their difficulty and (c) rate their usefulness. Each expert is asked to rate the difficulty of the question by choosing one of the options: (1) too easy, (2) reasonably easy, (3) reasonably difficult and (4) too difficult. Then the expert is asked to rate the usefulness of the question by choosing one of the options: (1) not useful at all, (2) useful as a seed for another question, (3) useful but requires major improvements, (4) useful but requires minor improvements or (5) useful as it is. Additionally, in the second evaluation study, the experts have been asked to determine whether the following properties hold for each question: (1) it is relevant to the course content, (2) it has exactly one key, (3) it contains no clues to the key, (4) it requires more than common knowledge to be answered correctly, and (5) it is grammatically correct.

Two samples of the KR questions which have been rated by at least 2 reviewers as useful (or useful with minor improvements) have been presented to the students enrolled in the KR course unit during the academic year 2013/2014 and who were about to sit their final exam. To increase participation rate, the two samples were delivered through different channels; the first sample of 6 questions was delivered using a traditional (closed-book) paper-and-pencil method during a revision session and the second sample of (different) 6 questions was delivered via the universities' eLearning portal. The first round was time-limited, i.e., students were instructed to hand back their answers after 10 min. In contrast, in the second round, students were allowed to finish the test during the week preceding their final exam. A total of 19 and 7 students participated in the two rounds, respectively.

4 Results and Discussion

913 and 428 questions were generated from the KR and Java ontologies, respectively. Among these are 535, 344 questions that have at least 3 distractors for the KR and Java ontologies, respectively. Among the 913 KR questions, 50 questions were selected randomly to be reviewed by domain experts. Java questions were first fed to an automatic filter that filters out any question where there is an overlap, i.e., a string of more than three characters, that appears in both the stem and the key. This is to eliminate the chances of providing a clue to the students to figure out the correct answer without actually having the required knowledge to answer it correctly. Among the 264 Java questions that have no overlap, 65 questions were selected for the reviewing phase. These questions were not selected in a pure random way, rather, each question has been selected randomly, then checked manually to make sure that it is not redundant, i.e., similar in terms of content to a previously selected question but different in presentation.

A question is considered "useful" if it is rated as either "useful as it is" or "useful but requires minor improvements" by a reviewer. 46 out of the 50 KR questions were considered useful by at least one reviewer. 63 out of the 65 Java questions have been rated as useful by at least one reviewer. A given distractor is considered "useful" if it has been functional (i.e., picked by at least one student). For the questions delivered in the first round, at least two out of three distractors were useful. For the questions delivered in the second round, at least one distractor was useful except for one question which has been answered correctly by all the seven students.

We used Pearson's coefficient to compute item discrimination to show the correlation between students' performance on a given question and the overall performance of each student on all questions. The range of item discrimination is $[-1, +1]$. A good discrimination value is greater than 0.4 [10]. For the questions administered on paper and four out of the six questions administered online, item discrimination was greater than 0.4. For one question administered online, item discrimination could not be calculated as 100 % of students answered that question correctly. One of the reviewers pointed out that the question that has poor discrimination is highly guessable because of the conceptual similarities between the stem and the key.

The quality of Java questions was further evaluated by adherence to five rules. Figure 1 shows the number of questions adhering to each rule as evaluated by each reviewer. In general, almost all questions have been found by all three reviewers to adhere to Rules 1 (relevance), 2 (exactly 1 key), and 4 (requires domain knowledge). Almost all questions were also found to adhere to Rule 3 (question contains no clues) by two reviewers. Rule 5 (grammatical correctness) was the only rule which got low ratings. According to reviewers' comments, this is mainly due to the lack of appropriate articles (i.e., the, a, an).

A question is too difficult for a particular group of students if it is answered correctly by less than 30 % of the students and is too easy if answered by more than 90 % of the students [9]. In both cases, the question needs to be reviewed and improved. Accordingly, we consider a question to be difficult if it is answered correctly by 30–60 % and easy if it is answered correctly by 60–90 % of the students. With regard to the six questions delivered on paper, two questions were reasonably difficult and two were reasonably easy for the students. These four questions were in line with difficulty estimations by the MCQ Generator tool. One out of the six questions was too difficult for the students. Remarkably, the tool and the three reviewers

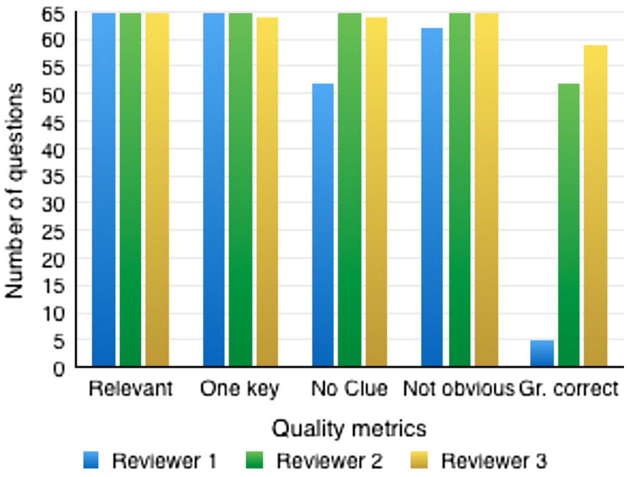

Fig. 1 Quality of questions according to reviewers' evaluations

have rated this item as easy. Finally, one question was too easy for the students, however it was rated as difficult by the tool. This is due to having a clue in the stem. Similarly, for the questions administered online, one question was reasonably difficult and one question was reasonably easy for the students; just in line with tool estimations. One out of the six questions was too easy for the students (100 % correct answers). This question was rated as easy by the tool. Again, one question was rated as difficult by the tool but was easy for the students due to having a clue in the stem. Two questions were not in line with tool estimations but were in line with estimations of at least two reviewers. For 51 out of the 65 Java questions, there has been an agreement between the tool and at least one reviewer. The degree of agreement is much higher with easy questions reaching 100 % agreements with at least one reviewer. This could mean that the generated distractors for difficult questions were not plausible enough which can be due to many reasons such as having lexical or semantic clues. Figure 2 shows the number of questions for which there is an agreement between the tool and at least one, two or three reviewers for the 50 KR questions and the 65 Java questions that have been reviewed by the domain experts.

5 Related Work

Due to space limitations, a selected set of related work is presented in this section. For a detailed review of the related literature, the interested reader is referred to [2]. A number of ontology-based question generation approaches have been proposed [3, 6, 7, 12, 15, 19, 20]. For example, Zitko et al. [19] proposed templates and algorithms for the automatic generation of objective questions from ontologies. The focus in their work was to extend the functionality of a previously implemented tutoring system (TexSys) by concentrating on the assessment component. The main difference between this approach and our approach is in the distractor selection mechanism. The mechanism adopted by Zitko et al. is to generate a set of *random* distractors for each MCQ without an attempt to filter them according to their pedagogical appropriateness.

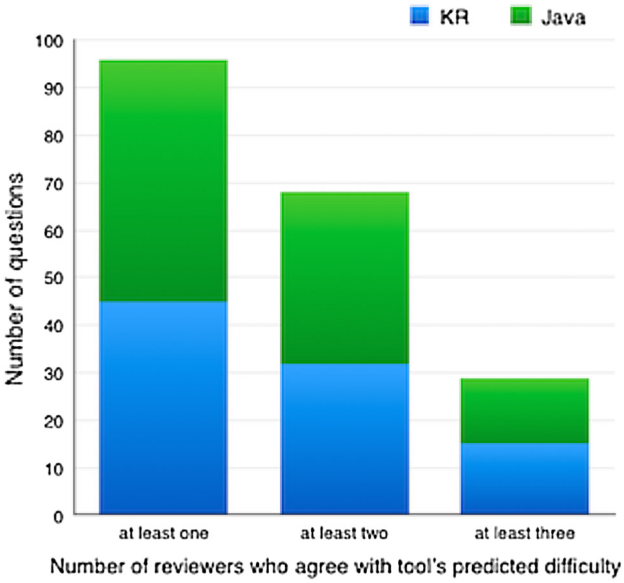

Fig. 2 Tool-reviewers' agreements on difficulty predictions

The distractor selection mechanism was enhanced by Papasalouros et al. [15] who presented various ontology-based strategies for the automatic generation of MCQs. These strategies are used for selecting keys and distractors. The evaluation of the produced questions by domain experts shows that the questions are satisfactory for assessment but not all of them are syntactically correct. The major problem related to this approach is the use of highly constrained rules with no theory backing that would motivate the selection of these rules. For example, the distractors in each MCQ are mainly picked from the set of siblings of the correct answer while there might be other plausible distractors. Later, Cubric and Tosic [6] reported on their experience in implementing a *Protégé* plugin for question generation based on the strategies proposed by Papasalouros et al. [15]. More recently, Cubric and Tosic [7] extended their previous work by considering new ontology elements, e.g., annotations. In addition, they suggested employing question templates to avoid syntactical problems in the generated questions. They have also illustrated, by some examples, that their method is suitable for generating questions of both lower and higher levels of Bloom's taxonomy [5].

In addition to the distractor selection mechanism, it is important to consider some presentation issues that might affect the quality of the generated questions, e.g., naturalness and fluency of the language. Consistent with this, Williams [17] extends the use of SWAT[4] natural language tools to verbalise ontology terms which are used in the generated questions. For example, "has a height of" can be derived from the data property "has_height".

6 Outlook

The current project presents a range of contributions including a theory to control MCQs difficulty and a protocol to evaluate automatically generated assessment questions. The presented methodology has been externally validated over 4 existing ontologies in [18] to show the suitability of the approach for existing ontologies and not

[4] http://swat.open.ac.uk/tools/.

only handcrafted ones. However, there are still some open issues that need to be addressed. We suggest a few possible future directions. For example, we are currently exploring alternative models to control the difficulty of different classes of MCQs that may not fit the current similarity-based theory. In addition, we are exploring the suitability of using the generated MCQs to validate the ontology they are generated from.

References

1. Alsubait T, Parsia B, Sattler U (2014) Measuring similarity in ontologies: a new family of measures. In: Proceedings of the 19th international conference on knowledge engineering and knowledge management. Sweden
2. Alsubait T (2015) Ontology-based multiple-choice question generation. PhD thesis, School of Computer Science, The University of Manchester
3. Al-Yahya M (2014) Ontology-based multiple choice question generation. Sci World J. doi:10.1155/2014/274949
4. Baader F, Calvanese D, McGuinness DL, Nardi D, Patel-Schneider PF (eds) (2007) The description logic handbook: Theory, implementation and applications. Cambridge University Press, Cambridge
5. Bloom BS, Krathwohl DR (1956) Taxonomy of educational objectives: the classification of educational goals by a committee of college and university examiners. Handbook 1. Cognitive domain. Addison-Wesley, New York
6. Cubric M, Tosic M (2009) SeMCQ—*Protégé* Conference. Poster and Demo Session
7. Cubric M, Tosic M (2010) Towards automatic generation of e-assessment using semantic web technologies. In: Proceedings of the 2010 international computer assisted assessment conference, University of Southampton
8. d'Amato C, Staab S, Fanizzi N (2008) On the influence of description logics ontologies on conceptual similarity. In: Proceedings of the 16th international conference on knowledge engineering: Practice and Patterns EKAW'08
9. Davis BG (2001) Tools for Teaching. Jossey-Bass, San Francisco
10. Ebel RL (1954) Procedures for the analysis of classroom tests. Educ Psychol Measurement 14:352–364
11. Lehmann K, Turhan A (2012) A framework for semantic-based similarity measures for ELH-concepts. In: del Cerro LF, Herzig A, Mengin J (eds) Logics in artificial intelligence. Lecture Notes in Computer Science, vol 7519. Springer, Berlin, pp 307–319
12. Holohan E, Melia M, McMullen D, Pahl C (2005) Adaptive e-learning content generation based on semantic web technology. In: Proceedings of workshop on applications of semantic web technologies for e-learning
13. Lowman J (1995) Mastering the techniques of teaching, 2nd edn. Jossey-Bass, San Francisco
14. Mitkov R, An Ha L, Karamani N (2006) A computer-aided environment for generating multiple-choice test items. In: Natural language engineering, vol 12(2). Cambridge University Press, Cambridge, pp 177–194
15. Papasalouros A, Kotis K, Kanaris K (2008) Automatic generation of multiple-choice questions from domain ontologies. In: IADIS e-learning 2008 conference. Amsterdam
16. Tongphu S, Suntisrivaraporn B (2014) On desirable properties of the structural subsumption-based similarity measure. In: Proceedings of the 4th Joint international semantic technology conference. Thailand
17. Williams S (2011) Generating mathematical word problems. In: 2011 AAAI Fall symposium series
18. Vinu EV, Sreenivasa Kumar P (2015) A novel approach to generate MCQs from domain ontology: considering DL semantics and open-world assumption. In: Journal of Web Semantics: Science, Services and Agents on the World Wide Web
19. Zitko B, Stankov S, Rosic M, Grubisic A (2008) Dynamic test generation over ontology-based knowledge representation in authoring shell. Expert Syst Appl Int J 36(4):8185–8196
20. Zoumpatianos K, Papasalouros A, Kotis K (2011) Automated transformation of SWRL rules into multiple-choice questions. In: FLAIRS conference'11