Undefined 1 (2015) 1–5
IOS Press

Automated Generation of Assessment Tests from Domain Ontologies

Editor(s): Name Surname, University, Country
Solicited review(s): Name Surname, University, Country
Open review(s): Name Surname, University, Country

Vinu E.V * and P Sreenivasa Kumar
Department of Computer Science and Engineering, Indian Institute of Technology Madras, Chennai, India
E-mail: {vinuev,psk}@cse.iitm.ac.in

Abstract. We investigate the effectiveness of OWL-DL ontologies in generating multiple choice questions (MCQs) that can be employed for conducting large scale assessments. The details of a prototype system called *Automatic Test Generation* (ATG) system and its extended version called Extended-ATG system are elaborated in this paper. The ATG system was useful in generating multiple choice question-sets of a required cardinality, from a given formal ontology. This system is further enhanced to include features such as finding the difficulty values of generated MCQs and controlling the overall difficulty-level of question-sets, to form Extended-ATG system. This paper discusses the novel methods adopted to address these new features. While the ATG system uses at most two predicates for generating the stems of MCQs, the E-ATG system has no such limitations and employs several interesting predicate based patterns for stem generation. These predicate patterns are obtained from a detailed empirical study of large real-world question-sets. In addition, the system also incorporates a specific non-pattern based approach which make use of aggregation-like operations, to generate questions that involve superlatives (e.g., *highest* mountain, *largest* river etc.). We have tested the applicability and efficacy of the proposed methods by generating MCQs from several online available ontologies, and verified our results in a classroom setup — incorporating real students and domain experts.

Keywords: MCQ Generation, Question Generation, Ontologies, E-learning System

1. Introduction

Web Ontology Language (OWL) ontologies are knowledge representation structures that are designed to represent rich and complex knowledge about things, groups of things, and relations between things [23,22]. These ontologies have widely flourished in recent years due to the advancement of Semantic Web technologies and, due to ease of publishing knowledge in online repositories. The use of knowledge captured in these ontologies, by e-learning systems, to improve learning and teaching process in a particular domain, is an advancing area of research.

Assessment authoring modules are the first to be implemented and currently the most accepted component in e-learning systems [14,13]. The majority of the existing e-learning systems (such as Moodle[1] and WebCT[2]), support only the administration of manually constructed question-sets. Further upgrades on these systems were mainly focusing on introducing new question types, improving the aesthetics of the interface to make the question authoring process easier and on enhancing the overall management of the system. Despite all these upgrades, enormous amount of time, money and skill are required for setting up a

*Corresponding author. E-mail: vinuev@cse.iitm.ac.in, mvsquare1729@gmail.com

[1] http://www.moodle.org/
[2] http://www.cuhk.edu.hk/eLearning/c_systems/webct6/

question-set [10,31] — making it necessary to have an automated solution.

This problem of automated generation of assessment tests has recently attracted notable attention in the research as well as education communities [18]. This is particularly due to its importance in the new emerging education styles; for instance, online courses like the MOOCs (Massive Open Online Courses) conduct multiple choice quizzes at regular intervals, to evaluate the mastery of their students [32,6].

One possible solution to this problem is to incorporate an intelligent module in the e-learning system which can generate possible domain related question items, from a given knowledge source. Knowledge sources like ontologies are of great use here, since they can represent the knowledge of a domain in the form of logical axioms. But, having a question generation module does not fully overcome the underlying problem. There should be effective mechanisms for selecting those questions which are apt for conducting an assessment test.

In a pedagogical environment, an e-learning system should be intelligent enough to serve various assessment goals. For example, the system should be able to tackle the scenarios like selecting the top ten students who are having high domain knowledge proficiency, using a question-set of limited cardinality, say 20 or 25. To tackle such common scenarios, an e-learning system should be able to predetermine the difficulty-levels of the question items which they generate. Also, there should be provisions for controlling the count of questions in the final question-set and its overall difficulty-level.

In our research, we studied the effectiveness of Web Ontology Language (OWL) ontologies in generating question-sets which can be employed for conducting large-scale multiple choice questions (MCQs) based assessment tests. This is achieved by implementing a prototype system called *Automatic Test Generation* (ATG) system. The details of the ATG system is given in [35].

There are several works in the literature, which describe the usefulness of OWL ontologies in generating MCQs [27,17,8,3,38]. Studies in [2] have shown that ontologies are good for generating *factual* (or *knowledge-level*) MCQs. These knowledge-level questions help in testing the first level of Bloom's taxonomy [12], a taxonomy of educational objectives for a cognitive domain. Throughout this paper, by MCQs we mean factual-MCQs.

Recently, publications like [2,1,35], show that MCQs can be generated from the assertional facts (ABox axioms) associated with the ontology. In this paper, we categorize the approaches that use ABox axioms to generate MCQs into two types: 1) Pattern-based factual question generation and 2) Non-Pattern-based factual question generation. In our earlier work [35], we focused mainly on the first approach and did not explore the second approach fully. We termed the second approach as *Ontology-specific* question generation approach in our earlier work, but later determined that it that *Non-Pattern-based* approach as the appropriate term for it. In this work, we explore a sub-category of Non-Pattern-based questions, called *Aggregation-based* questions, and its generation technique.

In the ATG system, we introduced a systematic method for generating Pattern-based MCQs. The proposed method considers predicate (or property) patterns associated with *individuals* in an ontology, for generating MCQ stems. Two major drawbacks associated with the practicality of this question generation approach were: (1) human intervention is needed to screen the irrelevant or out-of-domain questions, (2) the approach generates thousands of questions, making it difficult even for a human expert to make the selection of a small question-set.

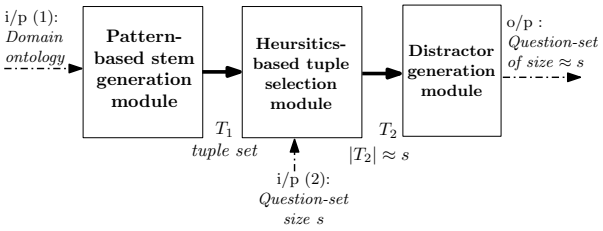

Fig. 1. The figure shows the workflow of the ATG system. The two inputs to the system are: a domain ontology and the question-set size.

We have addressed these issues by including a *heuristics based question (or tuple) selection module* (Module-2) in the ATG system. This module helps in selecting only those questions which are ideal for conducting a domain-specific assessment. A detailed summary of this module in given in Section 6.

Figure 1 shows the overview of the workflow of the ATG system, where the system takes two inputs: a domain ontology (an OWL ontology) and the size of the question-set to be generated, and it produces a question-set of size approximately equal to the required size. In this system, the count of the questions

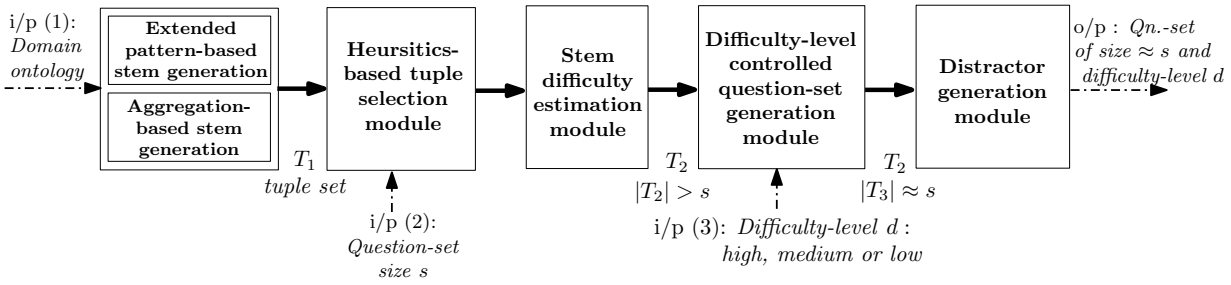

Fig. 2. The figure shows the workflow of the Extended-ATG system. The inputs to the system are: (1) a domain ontology (an OWL ontology), (2) size of the question-set to be generated and, (3) its difficulty-level.

to be generated is controlled by varying the parameters associated with the heuristics in Module-2.

This paper mainly features the new modules that are introduced in the extended version of the ATG system (called Extended-ATG or simply E-ATG system), and their significance in generating question-sets which are useful for educational purposes. The E-ATG system is an augmented version of the ATG system, with added features such as determining and controlling the difficulty values (or *difficulty-scores*) of MCQs and controlling the overall difficulty-level of question-sets. An overview of the workflow of E-ATG is given in Figure 2. In addition to the three modules of the ATG system, the E-ATG system has three additional modules.

Considering the approaches which we followed for building these three modules, the main contributions of this paper can be listed as follows:

1. A detailed study of Pattern-based MCQs, using patterns that involve more than two predicates, leading to an extended submodule for Pattern-based stem generation.
2. A generic (ontology independent) technique to generate Aggregation-based MCQs, resulting in a submodule for Aggregation-based stem generation.
3. A novel method to determine the difficulty of a generated MCQ stem, which give rise to a module for difficulty estimation of stem.
4. An algorithmic way to control the overall difficulty-level of a question-set, leading to a module for question-set generation and controlling its difficulty-level.

To set the context for explaining our work, an overview of these contributions of are given in Section 3.

In this paper, we use examples from two well-known domains — Movies and U.S geography — for illustrating our approaches. Three appendices are provided at the end of the paper. Appendix A gives a detailed explanation of the psychometric method which we have adopted in our empirical study. A sample set of system-generated MCQ stems that are used in the empirical study is listed in Appendix B. Appendix C shows the notations and abbreviations that are used in this paper.

2. Background

2.1. Multiple Choice Questions (MCQs)

An MCQ is a tool that can be used to evaluate whether (or not) a student has attained a certain learning objective. It consists of the following parts:

- ***Stem** (S).* Statement that introduces a problem to a learner.
- ***Choice set**.* Set of options corresponding to S, denoted as $A = \{ A_1, A_2, ..., A_m \}, m \geq 2$. It can be further divided into two sets:
  - ***Key***. Set of correct options, denoted as $K = \{ A_1, A_2, ..., A_i \}, 1 \leq i < m$.
  - ***Distractors***. Set of incorrect options, denoted as $D = \{ A_{i+1}, ..., A_m \}$.

$Note$: In this paper, we assume K as a singleton set. We fix the value of m, the number of options, in our experiments as 4, as it is the standard practice in MCQ tests.

2.2. Pattern-based MCQs

Pattern-based MCQs are those MCQs whose stems can be generated using simple SPARQL templates. These stems can be considered as a set of conditions which ask for an answer which is explicitly present in the ontology. Questions like *Choose a C?* or *Which of*

the following is an example of C? (where *C* is a concept symbol), are some of the examples of such stems. Example-1 is a Pattern-based MCQ, which is framed from the following assertions that are associated with the (key) *individual* `birdman`.

```
Movie(birdman)
isDirectedBy(birdman,alejandro)
hasReleaseDate(birdman,"Aug 27 2014")
```

Example 1 Choose a `Movie`, which `isDirectedBy` `alejandro` and `hasReleaseDate` "`Aug 27, 2014`".

Options	
a.	Birdman
b.	Titanic
c.	Argo
d.	The King's Speech

The possible predicate[3] combinations of size[4] one w.r.t. an individual x can be denoted as: $x\ \overrightarrow{O}\ i$, $x\ \overleftarrow{O}\ i$, $x\ \overrightarrow{D}\ v$ and $x\ \overrightarrow{a}\ C$, where i is an individual, $\overrightarrow{a}$ is `rdf:type`, $\overrightarrow{O}$ and $\overleftarrow{O}$ represent object properties of different directions, $\overrightarrow{D}$ denotes datatype property, v stands for the value of the datatype property and C is a class name. We call the individual x as the *reference-individual* of the predicate combination. The arrows ($\leftarrow$ and $\rightarrow$) represent the directions of the predicates w.r.t. the reference-individual. In this paper, we often use the terms question-template and predicate combination interchangeably, but the former term specifically denotes the predicate combination along with the position of the key.

Table 1 shows the formation of possible predicate combinations of size two and three by adding predicates to the four combinations of size one. The repetitions in the combinations are marked with the symbol "*". Note that, in those predicate patterns, we consider only the directionality and type of the predicates, but not their order. Therefore the combinations like $i_2\ \overleftarrow{O_2}\ x\ \overleftarrow{O_1}\ i_1$ and $i_1\ \overrightarrow{O_1}\ x\ \overrightarrow{O_2}\ i_2$ are considered to be the same. We refer one as duplicate of the other. After avoiding the duplicate combinations, we get 4 combinations of size one, 10 combinations of size two and 26 combinations of size three. These 40 predicate combinations can be used as the basic set of question-templates for constructing Pattern-based MCQ stems.

The distractors for these MCQs are selected from the set of individuals (or in some cases datatype values) of the ontology which satisfies the intersection classes of the domain or range of the predicates in the stem. This set is known as the *Potential-set* of the stem. Detailed explanation of distractor generation is given in Section 9.

2.2.1. Aggregation-based MCQs

Aggregation-based questions are those questions which cannot be directly obtained from a domain ontology, using patterns alone. These questions are again knowledge-level questions (or simply questions which check students' factual knowledge proficiency) of Blooms taxonomy [11]. But they require more reasoning skills to answer than Pattern-based MCQs. For example, "*Choose the state which is having the longest river.*", is an Aggregation-based question. (We assume that the there are no predicates in the ontology that explicitly contain the answer, that is `longestRiver` is not a property in the ontology.) This question can be answered only by a learner who knows about the states of a country, its rivers and the length of the rivers in it; and she should be able to reason over the known facts. We discuss more about a technique for generating Aggregation-based questions in Section 5. Our experiments based on Item Response Theory[5] (IRT), have shown that such MCQs are indeed difficult for a below-average learner to answer correctly.

3. Overview of the contributions

3.1. A detailed study of Pattern-based MCQs

An initial study on the approach for generating Pattern-based MCQs is given in [35], where questions are limited to predicate combinations of at most two predicates. In Section 4, we investigate approaches (or patterns) which generate questions that involve more than two predicates as well. Later in Section 4.1, we describe a study that we have done on a large set of real-world factual-questions — obtained from different domains — to explore the pragmatic usefulness and the scope of our approach.

[3] Includes both unary predicates (concept names) and binary predicates (role names)

[4] Signifies the number of predicates in a combination

[5] http://www.creative-wisdom.com/computer/sas/IRT.pdf

Table 1
The predicate combinations of sizes 1, 2 and 3, that are useful in generation Pattern-based MCQ stems are given below, where x denotes the reference-individual, i is a related individual, $\overrightarrow{a}$ is `rdf:type`, $\overrightarrow{O}$ and $\overleftarrow{O}$ represent object properties of different directions, $\overrightarrow{D}$ denotes datatype property, v stands for the value of the datatype property and C is a concept name. "*"-ed patterns are duplicate ones.

Predicate combinations: ↓		
Size: 1	2	3
$x\,\overrightarrow{O}\,i$	$i\,\overleftarrow{O}\,x\,\overrightarrow{O}\,i$	$i\,\overleftarrow{O}\,x\,(\overrightarrow{O}\,i)\,\overrightarrow{O}\,i$ $i\,\overleftarrow{O}\,x\,(\overrightarrow{D}\,v)\,\overrightarrow{O}\,i$ $i\,\overleftarrow{O}\,x\,(\overrightarrow{a}\,C)\,\overrightarrow{O}\,i$ $i\,\overleftarrow{O}\,x\,(\overleftarrow{O}\,i)\,\overrightarrow{O}\,i$
	$i\,\overrightarrow{O}\,x\,\overrightarrow{O}\,i$	$i\,\overrightarrow{O}\,x\,(\overrightarrow{O}\,i)\,\overrightarrow{O}\,i^*$ $i\,\overrightarrow{O}\,x\,(\overrightarrow{D}\,v)\,\overrightarrow{O}\,i$ $i\,\overrightarrow{O}\,x\,(\overrightarrow{a}\,C)\,\overrightarrow{O}\,i$ $i\,\overrightarrow{O}\,x\,(\overleftarrow{O}\,i)\,\overrightarrow{O}\,i$
	$v\,\overleftarrow{D}\,x\,\overrightarrow{O}\,i$	$v\,\overleftarrow{D}\,x\,(\overrightarrow{O}\,i)\,\overrightarrow{O}\,i^*$ $v\,\overleftarrow{D}\,x\,(\overrightarrow{D}\,v)\,\overrightarrow{O}\,i$ $v\,\overleftarrow{D}\,x\,(\overrightarrow{a}\,C)\,\overrightarrow{O}\,i$ $v\,\overleftarrow{D}\,x\,(\overleftarrow{O}\,i)\,\overrightarrow{O}\,i^*$
	$C\,\overleftarrow{a}\,x\,\overrightarrow{O}\,i$	$C\,\overleftarrow{a}\,x\,(\overrightarrow{O}\,i)\,\overrightarrow{O}\,i^*$ $C\,\overleftarrow{a}\,x\,(\overrightarrow{D}\,v)\,\overrightarrow{O}\,i^*$ $C\,\overleftarrow{a}\,x\,(\overrightarrow{a}\,C)\,\overrightarrow{O}\,i$ $C\,\overleftarrow{a}\,x\,(\overleftarrow{O}\,i)\,\overrightarrow{O}\,i^*$
$x\,\overleftarrow{O}\,i$	$i\,\overleftarrow{O}\,x\,\overleftarrow{O}\,i^*$	– – – –
	$i\,\overrightarrow{O}\,x\,\overleftarrow{O}\,i$	$i\,\overleftarrow{O}\,x\,(\overrightarrow{O}\,i)\,\overleftarrow{O}\,i^*$ $i\,\overrightarrow{O}\,x\,(\overrightarrow{D}\,v)\,\overleftarrow{O}\,i$ $i\,\overrightarrow{O}\,x\,(\overrightarrow{a}\,C)\,\overleftarrow{O}\,i$ $i\,\overrightarrow{O}\,x\,(\overleftarrow{O}\,i)\,\overleftarrow{O}\,i$
	$v\,\overleftarrow{D}\,x\,\overleftarrow{O}\,i$	$v\,\overleftarrow{D}\,x\,(\overrightarrow{O}\,i)\,\overleftarrow{O}\,i^*$ $v\,\overleftarrow{D}\,x\,(\overrightarrow{D}\,v)\,\overleftarrow{O}\,i$ $v\,\overleftarrow{D}\,x\,(\overrightarrow{a}\,C)\,\overleftarrow{O}\,i$ $v\,\overleftarrow{D}\,x\,(\overleftarrow{O}\,i)\,\overleftarrow{O}\,i$
	$C\,\overleftarrow{a}\,x\,\overleftarrow{O}\,i$	$C\,\overleftarrow{a}\,x\,(\overrightarrow{O}\,i)\,\overleftarrow{O}\,i^*$ $C\,\overleftarrow{a}\,x\,(\overrightarrow{D}\,v)\,\overleftarrow{O}\,i$ $C\,\overleftarrow{a}\,x\,(\overrightarrow{a}\,C)\,\overleftarrow{O}\,i$ $C\,\overleftarrow{a}\,x\,(\overleftarrow{O}\,i)\,\overleftarrow{O}\,i$

Predicate combinations ↓		
Size: 1	2	3
$x\,\overrightarrow{D}\,v$	$i\,\overleftarrow{O}\,x\,\overrightarrow{D}\,v^*$	– – – –
	$i\,\overrightarrow{O}\,x\,\overrightarrow{D}\,v^*$	– – – –
	$v\,\overleftarrow{D}\,x\,\overrightarrow{D}\,v$	$v\,\overleftarrow{D}\,x\,(\overrightarrow{O}\,i)\,\overrightarrow{D}\,v^*$ $v\,\overleftarrow{D}\,x\,(\overrightarrow{D}\,v)\,\overrightarrow{D}\,v$ $v\,\overleftarrow{D}\,x\,(\overrightarrow{a}\,C)\,\overrightarrow{D}\,v$ $v\,\overleftarrow{D}\,x\,(\overleftarrow{O}\,i)\,\overrightarrow{D}\,v^*$
	$C\,\overleftarrow{a}\,x\,\overrightarrow{D}\,v$	$C\,\overleftarrow{a}\,x\,(\overrightarrow{O}\,i)\,\overrightarrow{D}\,v^*$ $C\,\overleftarrow{a}\,x\,(\overrightarrow{D}\,v)\,\overrightarrow{D}\,v^*$ $C\,\overleftarrow{a}\,x\,(\overrightarrow{a}\,C)\,\overrightarrow{D}\,v$ $C\,\overleftarrow{a}\,x\,(\overleftarrow{O}\,i)\,\overrightarrow{D}\,v^*$
$x\,\overrightarrow{a}\,C$	$i\,\overleftarrow{O}\,x\,\overrightarrow{a}\,C^*$	– – – –
	$i\,\overrightarrow{O}\,x\,\overrightarrow{a}\,C^*$	– – – –
	$v\,\overleftarrow{D}\,x\,\overrightarrow{a}\,C^*$	– – – –
	$C\,\overleftarrow{a}\,x\,\overrightarrow{a}\,C$	$C\,\overleftarrow{a}\,x\,(\overrightarrow{O}\,i)\,\overrightarrow{a}\,C$ $C\,\overleftarrow{a}\,x\,(\overrightarrow{D}\,v)\,\overrightarrow{a}\,C$ $C\,\overleftarrow{a}\,x\,(\overrightarrow{a}\,C)\,\overrightarrow{a}\,C$ $C\,\overleftarrow{a}\,x\,(\overleftarrow{O}\,i)\,\overrightarrow{a}\,C$

3.2. A technique to generate Aggregation-based MCQs.

A generic technique to generate (a subset of possible) Aggregation-based MCQs is proposed in Section 5. This technique incorporates a specific non-pattern based approach which make use of aggregation-like operations, to generate questions that involve superlatives (e.g., *highest* mountain, *largest* river etc.).

3.3. A method to determine the difficulty of MCQ stems

Similarity-based theory [5] was the only effort in the literature [19,9], which helped in determining or controlling the difficulty-score of an ontology generated MCQ. The difficulty-score calculated by similarity-based theory considers only the similarity of the distracting answers with the correct answer — high similarity implies high difficulty-score and vice versa. In many a case, the stem of an MCQ is also a deciding factor for its difficulty. For instance, the predicate combination which is used to generate a stem can be chosen such that it makes the MCQ harder or easy to answer. Also, the use of *indirect addressing of instances*[6] in a stem, has a role in its difficulty. We investigate these aspects in Section 7 and, propose a novel method for calculating the difficulty-score of a system-generated stem.

An evaluation based on a complex psychometrical model is detailed in Section 10.2, to find the efficacy of the proposed difficulty-score calculation method.

In this paper, by difficulty value or difficulty-score we mean a numeric value which signifies the hardness of an MCQ, and by difficulty-levels we mean a predetermined ranges of difficulty-scores, corresponding to the standard scales: high, medium and low.

3.4. An algorithm to control the difficulty-level of a question-set

In Section 8, we propose a practically adaptable algorithmic method to control the difficulty-level of a question-set. This method controls the overall difficulty-level of a question-set by varying the count of the questions which are having (relatively) high difficulty-scores in the question-set.

[6]Instead of using the instance "Barack_Obama", one can use "44^{th} president of the U.S."

3.5. Other contributions

In addition to the above mentioned contributions, in Section 6, we discuss the existing *heuristics for question-set generation* (which we used in the ATG system), and possible modifications in these heuristics to generate question-sets which are more closer to human generated ones. We used the modified heuristics in the E-ATG system. For the completeness of the work, a section on *distractor generation* (Section 9) is also given, which illustrates the approach which we followed in the *distraction generation module* of the ATG system as well as in its extended version.

4. A detailed study of Pattern-based MCQs

As we have stated before, the stem of a Pattern-based MCQ can be framed from the tuples that are generated using the the basic set of 40 predicate combinations.

The question of whether all these 40 predicate combinations need to be used for generating the tuples, cannot be addressed at this point. Also, the issue of which among the variables of a combination should be considered for the position of a key, cannot be decided now. In the next subsection, we describe an empirical study which we have done on a large set of real-world FQs (factual-questions), to identify common FQs and their features. Based on that study, we will show how to select a subset from the basic set of predicate combinations (along with the possible position of their keys) as the *essential question-templates* for real-world FQ generation. In the *Pattern-based stem generation sub-module* of the E-ATG system, we make use of these essential question-templates for stem generation.

4.1. An empirical study of real-world FQs

Since there are no rules as such for how a FQ should look like, it was not possible to fix a scope for our study on Pattern-based questions. In order to claim that our approach could cover FQs which are useful in conducting any domain specific test and are equivalent to those questions which are chosen by experts who are having different levels of expertise, we analyzed 1748 FQs gathered from three different domains: the United States geography domain[7], the job posting domain[8]

[7] https://files.ifi.uzh.ch/ddis/oldweb/ddis/fileadmin/ont/nli/geoqueries_877.txt (last accessed 1st July 2015)

[8]https://files.ifi.uzh.ch/ddis/oldweb/ddis/fileadmin/ont/nli/jobqueries_620.txt (last accessed 1st July 2015)

and the restaurant domain[9]. The set of FQs (question-set) corresponding to these domains were gathered by Mooney's research group[10] of the University of Texas, with the help of a web-interface from real-people.

From these question-sets, we removed invalid questions and (manually) classified the rest into Pattern-based and Non-Pattern-based questions. We manually identified 570 Pattern-based questions and 729 Non-Pattern-based questions from the question-sets. We then tried to map each of these Pattern-based questions to any of the 40 predicate combinations (given in Table 1). We could map each of the 570 Pattern-based questions to at least one of the predicate combinations. This generalizes the fact that our patterns are effective in extracting almost all kinds of real-world (pattern-based) questions that could be generated from a given domain ontology.

4.2. Predicate combinations to question-templates

We have observed that most of the predicate combinations are not being mapped to by any real-world Pattern-based questions. Out of 40 combinations only 13 are found to be necessary to generate such real-world questions. We call those predicate combinations as the *essential predicate combinations*, for real-word FQ generation.

From the 13 essential predicate combinations, we have framed 19 question patterns based on our study on the features of real-world FQs. We call these question patterns as the *essential question-templates*. These question-templates are obtained by identifying the variables in the essential predicate combinations whose values can be considered as keys — we call such variables as the *key-variables* of the patterns. We list the identified essential question-templates in Column-2 of Table 2. The circled variables in the patterns denote the positions of their key (i.e., the key-variables). The square boxes represent the variables whose values can be removed while framing the stem. For example, the question: *What is the population of the state with capital Austin?* can be generated from the pattern: $\textcircled{v}\ \overleftarrow{D}\ \boxed{x}\ (\overrightarrow{a}\ C)\ \overrightarrow{O}\ i$, with v as the key-variable, D as the property `statePopulation`, O as `hasCapital`, C as the concept `State` and i as the individual `austin`. Clearly, the value of the variable x is not mandatory to frame the question statement. If the variable value is incorporated in the question, we are providing additional information to the test takers, making the question more direct and less difficult to answer.

Suitable stem-templates[11] are associated with each of the patterns (as given in Table 2), to generate corresponding (controlled English) natural language stems. Further enhancement of the readability of the stem is done by tokenizing the property names in the stem. Tokenizing includes word-segmentation[12] and processing of camel-case, underscores, spaces etc.

We discuss the details of the Potential-sets (given in Column-4 of Table 2) corresponding to each of the question patterns in the forthcoming sections.

4.3. Practicality Issue of pattern-based question generation

For the efficient retrieval of data from the knowledge base, we transform each of the essential question-templates into SPARQL queries. For example, the stem-template and query corresponding to the question pattern $C\ \overleftarrow{a}\ \textcircled{x}\ \overrightarrow{O}\ i$ are:

– *Choose a* [?C] *with* [?O] [?i].

```
select ?C ?x ?O ? where { ?x a ?C.
?x ?O ?i. ?O a owl:ObjectProperty. }
```

These queries, when used to retrieve tuples from ontologies, may generate a large result set. Last column of Table 3 lists the total count of tuples that are generated using the 19 question-templates from a selected set of domain ontologies. These tuple counts represent the possible Pattern-based questions that can be generated from the respective ontologies. From the Restaurant ontology, using the query corresponding to the question pattern number 6 alone, we could generate 288594 tuples.

An MCQ based exam is mainly meant to test the wider domain knowledge with a fewer number of questions. Therefore, it is required to select a small set of significant tuples from the large result set, to

[9] https://files.ifi.uzh.ch/ddis/oldweb/ddis/fileadmin/ont/nli/restaurantqueries_251.txt (last accessed 1st July 2015)

[10] https://www.cs.utexas.edu/ mooney/

[11] These are sample-templates for the readers reference. Minor variation on these templates were introduced at a later stage, to improve the question quality.

[12] Word-segmentation is done by using Python WordSegment (https://pypi.python.org/pypi/wordsegment — last accessed 11th May 2015), an Apache2 licensed module for English word segmentation

Table 2

The table shows the *essential question-templates* for real-world FQ generation, where the circled variables in the patterns denote the key-variables. The corresponding stem-templates and potential-set formulas are also listed.

No.	Question patterns	Stem-template	Potential-set w.r.t. key-variables
1	$x\ \overrightarrow{O}\ (i)$	*Choose a/the O of x.*	$Range(O)$
2	$x\ \overleftarrow{O}\ (i)$	*Choose the one with O x.*	$Domain(O)$
3	$x\ \overrightarrow{D}\ (v)$	*Choose a/the D of x.*	$Range(D)$
4	$(x)\ \overrightarrow{a}\ C$	*Choose a C.*	C
5 a.	$(v)\ \overleftarrow{D}\ x\ \overrightarrow{O}\ i$	*Choose a/the D of x with O i.*	$Range(D)$
b.	$(v)\ \overleftarrow{D}\ \boxed{x}\ \overrightarrow{O}\ i$	*Choose a/the D of the one with O i.*	$Range(D)$
6	$(v)\ \overleftarrow{D}\ \boxed{x}\ \overleftarrow{O}\ i$	*Choose the D of the one which is the O of i.*	$Range(D)$
7	$C\ \overleftarrow{a}\ (x)\ \overrightarrow{D}\ v$	*Choose a/the C with D v.*	$Domain(D)\ \sqcap\ C$
8 a.	$C\ \overleftarrow{a}\ (x)\ \overrightarrow{O}\ i$	*Choose a/the C with O i.*	$Domain(O)\ \sqcap\ C$
b.	$C\ \overleftarrow{a}\ \boxed{x}\ \overrightarrow{O}\ (i)$	*Choose a/the O of a C.*	$Range(O)$
9 a.	$C\ \overleftarrow{a}\ \boxed{x}\ \overleftarrow{O}\ (i)$	*Choose the one with a C as O.*	$Domain(O)$
b.	$C\ \overleftarrow{a}\ (x)\ \overleftarrow{O}\ i$	*Choose a/the C, which/who is O of i.*	$Range(O)\ \sqcap\ C$
10 a.	$(i_1)\ \overrightarrow{O_1}\ \boxed{x}\ (\overrightarrow{a}\ C)\ \overrightarrow{O_2}\ i_2$	*Choose the one whose O_1 is a C with O_2 i_2.*	$Domain(O_1)$
b.	$i_1\ \overrightarrow{O_1}\ \boxed{x}\ (\overrightarrow{a}\ C)\ \overrightarrow{O_2}\ (i_2)$	*Choose a/the O_2 of a C which/who is O_1 of i_1.*	$Range(O_2)$
11 a.	$v\ \overleftarrow{D}\ \boxed{x}\ (\overrightarrow{a}\ C)\ \overrightarrow{O}\ (i)$	*Choose a/the O of a C with D v.*	$Range(O)$
b.	$v\ \overleftarrow{D}\ (x)\ (\overrightarrow{a}\ C)\ \overrightarrow{O}\ i$	*Choose a/the C with D v and O i .*	$Domain(D)\ \sqcap\ C\ \sqcap\ Domain(O)$
c.	$(v)\ \overleftarrow{D}\ \boxed{x}\ (\overrightarrow{a}\ C)\ \overrightarrow{O}\ i$	*Choose a/the D of a C with O i.*	$Range(D)$
12	$(v)\ \overleftarrow{D}\ \boxed{x}\ (\overrightarrow{a}\ C)\ \overleftarrow{O}\ i$	*Choose the D of a C who/which is O of i.*	$Range(D)$
13	$i_1\ \overleftarrow{O_1}\ \boxed{x}\ (\overrightarrow{a}\ C)\ \overrightarrow{O_2}\ (i_2)$	*Choose a/the O_2 of a C whose O_1 is i_1.*	$Range(O_2)$

create a good MCQ question-set. But, the widely adopted method of random sampling can result in poor question-sets (we have verified this in our experiment section). In Section 6, we propose three heuristic based techniques to choose the most appropriate set of tuples (questions) from the large result set.

Table 3

The specifications of the test ontologies and the count of the tuples that were generated using the 19 question-templates, are given below.

Ontology	Individuals	Concepts	Object properties	Datatype properties	Total tuple count
Mahabharata	181	17	24	9	72074
Geography	713	9	174	11	449227
DSA	161	94	29	13	128213
Restaurant	9747	4	9	5	1850762
Job	4138	7	7	12	877437

5. A technique to generate Aggregation-based MCQs

The MCQ question stems like:

- Choose the Movie with the highest number of academy awards.
- Choose the Movie with the lowest number of academy awards.

which cannot be explicitly generated from the tuples that are generated using the predicate combinations, can be framed by performing operations similar to aggregation.

To generate such MCQ stems, the method which we have adopted involves three steps: grouping of the generated tuples w.r.t the properties they contain, sorting and selecting border values of the datatype properties, and including suitable adjectives (like highest or lowest) for stem enhancement. In the example MCQ stems (from the Movie ontology),

`hasNumberOfAcademyAwards` is the datatype property which is used for grouping. The border values for this property were 12 and 0, which correspond to the instances: `ben-hur` and `breathless` respectively. The adjectives which were used are *highest* and *lowest*. A detailed explanation of the method is given in the next subsection.

Since we are making use of datatype property values for generating Aggregation-based MCQ stems from OWL ontologies, it should be noted that, many of the XML Schema datatypes are supported by OWL-2 DL [21]. OWL-2 DL has datatypes defined for Real Numbers, Decimal Numbers, Integers, Floating-Point Numbers, Strings, Boolean Values, Binary Data, IRIs, Time Instants and XML Literals.

5.1. Question generation in detail

This section details the operations that are carried out in the *Aggregation-based stem generation submodule* of the E-ATG system.

To generate Aggregation-based question stems, the tuples generated using the 19 patterns (in Table 2), are grouped based on the properties they contain. We call the ordered list of properties which are useful in grouping as the *property sequence* (represented as $\mathcal{P}$) of the tuples in each group. From the grouped tuples, only those groups whose property sequence contain at least one datatype property are selected for generating Aggregation-based question stems. For instance, consider the tuples-set given in Table 4, which is generated using the property combination $C \overleftarrow{a}\ x\ \overrightarrow{D}\ v$ from the Geography ontology. GROUP BY and ORDER BY clauses were used along with the pattern's SPARQL template for grouping and sorting respectively.

In Table 4, the highlighted rows (or tuples), which bore the border values of the datatype property, can be used for framing the Aggregation-based question stems — we call these rows as the *base-tuples* of the corresponding Aggregation-based questions. The datatype property names in the 1^{st} and 3^{rd} highlighted rows are paired with the predefined-adjectives (like maximum, highest, oldest, longest, etc.) and the pair with the highest ESA relatedness score is determined (see details in the next subsection). Then the stemmed predicate (i.e., for e.g., "hasPopulation" is stemmed to "Population"), the adjective and the template associated with the question pattern are used to generate stems of the following form (where the underlined words correspond to the predicates used):

Table 4

The table shows the list of tuples from the Geography ontology that are grouped and are sorted based on their property sequences and the datatype property values respectively. The highlighted rows denote the tuples which are chosen for generating Aggregation-based questions

C	x	D	v	Property seq.
State	connecticut	hasPopulation	5020	
State	florida	hasPopulation	68664	{State,
State	colorado	hasPopulation	104000	hasPopulation}
...	...	...	...	
State	arizona	hasPopulation	114000	
River	neosho	length	740	
River	wasbash	length	764	{River,
River	pecos	length	805	length}
...	...	...	...	
River	ouachita	length	973	

- Choose the State with the *highest* population. (Key: Arizona)
- Choose the River with the *longest* length. (Key: Ouachita)

Similarly, the predicates in the 2^{nd} and 4^{th} highlighted rows are paired with the adjectives like minimum, shortest, smallest, minimum, etc., to generate stems of the form:

- Choose the State with the *lowest* population. (Key: Connecticut)
- Choose the River with the *shortest* length. (Key: Neosho)

5.2. Explicit-Semantic-Analysis based stem enhancement

Having a property in hand, to fix which quantifying adjective — highest and lowest or longest and shortest — to use, is determined by calculating pairwise relatedness score and, then, choosing the one with highest score using Explicit Semantic Analysis (ESA) [20] method. ESA method computes semantic relatedness of natural language texts with the aid of very large scale knowledge repositories (like Wikipedia). EasyESA[13] [16], an infrastructure consisting of an open source platform that can be used as a remote service or can be deployed locally, is used in our implementation, to find pairwise relatedness scores.

[13] http://easy-esa.org/ (last accessed 21st Dec 2015)

The pair — (predicate, predefined-adjective) — with highest ESA relatedness score are used for framing the question.

Table 5
The ESA scores of some sample predicate–adjective pairs

Predicate	Adjective	ESA relatedness score
has Population	highest	0.0106816739
has Population	longest	0.0000000000
has Population	lowest	0.0132820251
has Population	longest	0.0000000000

For example, as shown in Table 5, the datatype property `hasPopulation` can be used along with "highest" or "lowest", depending on the border value under consideration, as those pairs have comparatively high relatedness score.

Out of the large set of datatypes offered by OWL-2 DL, datatypes of Binary Data, IRIs and XML Literals are avoided for stem formation, as they are not useful in generating human-understandable stems.

6. Heuristics for question-set generation

The heuristics-based tuple selection module of the ATG system uses three screening heuristics which mimic the selection heuristics followed by human experts. These heuristics help in generating question-sets that are unbiased and cover the required knowledge boundaries of a domain knowledge. A summary of the three screening methods is given in the next subsection. In the E-ATG system, we adopted a new heuristic instead of the third heuristic, to achieve better results. The drawback of the third heuristic and the details of the new heuristic are explained in Sections 6.1.3 and 6.2 respectively.

Even though these heuristics were meant for Pattern-based questions, we apply the same heuristics, except the third heuristic, to Aggregation-based questions as well. This is achieved by considering the base-tuples of Aggregation-based questions.

6.1. Summary of the existing heuristics

The three heuristics introduced in [35] were:

- Property based screening
- Concept based screening
- Similarity based screening

Interested readers can refer [35], for a detailed explanation of the rationales for using these heuristics.

6.1.1. Property based screening

The property based screening was mainly meant to avoid those questions which are less likely to be chosen by a domain expert for conducting a good MCQ test. This is achieved by looking at the triviality score (called *Property Sequence Triviality Score*, abbreviated as PSTS) of the property sequence of the tuples.

PSTS of a property sequence $\mathcal{P}$ was defined in [35] as follows (we later use this PSTS in Section 7):

$$\text{PSTS}(P) = \frac{\#\text{ Individuals satisfying all the properties in } \mathcal{P}}{\#\text{ Individuals in the Potential-set of } \mathcal{P}}$$

Potential-set of $\mathcal{P}$ denotes the set of individuals which *may* possibly satisfy the properties (or predicates) in $\mathcal{P}$. It is characterized by the expression $Type(Q, \mathcal{P}, r)$, where Q is the question pattern used for generating the tuples, r denotes a reference position in the pattern (see the following example). $Type(Q, \mathcal{P}, r)$ is defined as the intersection of the class constraints and the domain and range of those properties in $\mathcal{P}$ which are associated with r. Consider $\mathcal{P} = \{p_1, p_2\}$ in the pattern $Q = i_2\ \overrightarrow{p_1}\ x\ \overrightarrow{p_2}\ i_1$ and r as the pivot instance x. Then, $Type(Q, \mathcal{P}, x)$ is taken as $Range(p_1) \sqcap Domain(p_2)$. Similarly for $\mathcal{P} = \{C_1, p_1\}$ and $Q = C_1\ \overleftarrow{a}\ x\ \overleftarrow{p_1}\ i_1$, $Type(Q, \mathcal{P}, x) = C_1 \sqcap Range(p_1)$. For the same q and $\mathcal{P}$, if r is i_1, $Type(Q, \mathcal{P}, i_1)$ is taken as $Domain(p_1)$.

In the property based screening, the position of the reference-individuals (introduced in Section 2.2) is taken as r for finding the potential-set. The third column of Table 2 lists the generic formula to calculate the potential-sets for the 19 patterns. But, it should be noted that, the calculation is w.r.t. the *key-variables* – we later use this in Section 9.

A suitable threshold for PSTS is fixed based on the number of tuples to be filtered at this level of screening.

6.1.2. Concept based screening

This level of screening was mainly meant to select only those tuples which are relevant to a given domain, for MCQ generation.

We achieve this by looking at the reference-individual in a given tuple. If the individual satisfies any of the key-concept[14] of the domain, the question which is framed out of the corresponding tuple, can be considered as relevant for conducting a domain related test.

[14] In the implementation, we used KCE API [29] to extract a required number of potentially relevant concepts (or simple key-concepts) from a given ontology.

Table 6
The table shows the list of tuples in two locally-similar groups, generated from the Movie ontology. The highlighted rows denote the representative tuples — selected based on their *popularities*

x (Pivot instance)	O_1	i_1	O_2	i_2	Popularity
argo	wonAward	oscar_12	isBasedOn	the_great_escape	7.20
a_beautiful_mind	wonAward	oscar_01	isBasedOn	a_beautiful_mind_novel	8.26
forest_gump	wonAward	oscar_94	isBasedOn	forest_gump_novel	15.44
h_potter_and_the_sorcerers_stone	directedBy	chris_columbus	hasNextSequel	h_potter_and_the_chamber_of_secrets	34.51
jurassic_park	directedBy	steven_spielberg	hasNextSequel	the_lost_world_jurassic_park	22.56
hobbit_an_unexpected_journey	directedBy	peter_jackson	hasNextSequel	hobbit_the_desolation_of_smaug	14.47
national_treasure	directedBy	jon_turteltaub	hasNextSequel	national_treasure_book_of_secrets	7.89

The number of tuples to be screened in this level, is controlled by varying the count of the key concepts.

6.1.3. Similarity based screening

The tuple-set S, selected using the first two levels of screening, may contain (semantically) similar tuples; they will make the final question-set biased. To avoid this, selecting only a representative set of tuples from among these similar set of tuples is necessary. In [35], this issue was addressed by considering an undirected graph $G = (V, E)$, with vertex set $V = \{t \mid t \in S\}$, and edge set $E = \{\, (t_1, t_2) \,|\, t_1, t_2 \in S \, and \, Similarity(t_1, t_2) \geq c \,\}$, where $Similarity(.)$ is a symmetric function which determines the similarity of two tuples with respect to their reference-instances and c is the minimum similarity score threshold. From the graph, a minimum dominating set (i.e., a dominating set[15] of minimum cardinality) of nodes was selected as the set of representative tuples. The similarity measure which we have adopted is as follows:

$$Similarity(t_1, t_2) = \frac{1}{2}\left(\frac{\#(\, X(P(t_1)) \cap X(P(t_2))\,)}{\#(\, X(P(t_1)) \cup X(P(t_2))\,)}\right.$$

$$\left. + \frac{\#Triples\ in\ t_1\ Semantically\ Equivalent\ to\ triples\ in\ t_2}{Max(\#Triples\ in\ t_1,\ \#Triples\ in\ t_2)}\right)$$

In the equation, $P(t)$ represents the property sequence of t, and $X(P(t))$ denotes the set of individuals (in the ontology) which satisfies the properties in $P(t)$. The equation calculates the similarity score of two tuples based on the relationship between (unary and binary) predicates in one tuple to their counterparts in the other tuple, and the number of the semantically similar triples in them.

[15] A dominating set for a graph $G = (V, E)$ is the subset U of V s.t. $\forall v \in V \backslash U$, v is adjacent to at least one member of U.

Drawback of this heuristic: In our observation, selecting representative tuples based on minimum dominating set (MDS) — using an approximation algorithm[16] — is more like a random selection of representative nodes ensuring the dominating set constraints. Therefore, to improve the quality of the result, instead of simply finding the MDS, we select representative tuples from each of the similar set of tuples based on their *popularities*; details are given in the next subsection.

6.2. Proposed change in the heuristic

The tuples that are screened after two levels of filtering are grouped based on their similarity. The similarity measure from the previous subsection is reused for this grouping. In case if a tuple shows similarity to multiple groups, we place the tuple to the group to which it shows maximum similarity. Each of these groups are addressed as *locally-similar groups*; appropriate minimum similarity score (called *minimum local-similarity threshold* (denoted as mls)) is used for creating the groups.

Within a locally-similar group, a *popularity* based score is assigned to each of the tuples. The most-popular tuple from each group is taken as the representative tuple. An illustration of the selection of representative tuples is shown in Table 6, where the highlighted tuples denote the selected ones. In the table, the third tuple and the first tuple in the group-1 and group-2 respectively have higher popularity score than the rest of the tuples in the respective groups, making them the suitable candidates for the question-set.

[16] JGraph MDS

Calculation of the popularity of a tuple: The widely used popularity measure for a concept is based on the count of the individuals of the other concepts that are connected to the individuals of that concept [33]; we follow a similar approach to find the popularity of an individual. That is, the popularity of an individual x in an ontology $\mathcal{O}$, can be said as the *connectivity* of x from the other individuals in $\mathcal{O}$ which belong to a concept which x does not belong to. On getting the popularities of the individuals in a tuple t, we calculate the popularity of t in $\mathcal{O}$ as:

$$\textit{Popularity}(t, \mathcal{O}) = (1/2)\mathcal{C}_{t,\mathcal{O}}(t.r) + \sum_{i=1}^{n} \log\left(1 + \mathcal{C}_{t,\mathcal{O}}(x_i)\right)$$

The popularity of a tuple is defined as the sum of the connectivities of its individuals, by giving more preference to the connectivity value of the reference-individual — i.e., the sum of half the connectivity value of the reference-individual and $\log$ of the connectivity values of the other individuals are taken. In the equation, $t.r$ denotes the reference-individual of the tuple t; the set, $\{ x_1, x_2, ..., x_n \}$, denotes the individuals other than the reference-individual of t; $\mathcal{C}_{t,\mathcal{O}}(j)$ represents the connectivity (formally defined below) of an individual j in a tuple t from an ontology $\mathcal{O}$.

The connectivity of an individual x in t is defined as follows, where C_x and C_y are concepts in $\mathcal{O}$.

$$\mathcal{C}_{t,\mathcal{O}}(x) = \#\{y \mid \mathcal{O} \models C_x(x) \sqcap C_y(y) \sqcap R(y, x) \wedge \mathcal{O} \not\models C_y(x) \wedge not(C_x \sqsubseteq C_y) \wedge not(C_y \sqsubset C_x)\}$$

The equation gives the count of the individuals which are related to x by a relation R and whose satisfying concepts do not satisfy x and are not hierarchically (sub-class–super-class relationship) related to any of the satisfying concepts of x.

7. A method to determine the difficulty of MCQ Stems

One possible way to decide the difficulty value (or difficulty-score) of a stem is by finding how its predicate combination is making it difficult to answer. Our study on FQs which have been generated from different domain ontologies shows that, increasing the answer-space of the predicates in a stem has an effect on its difficulty value. For example, the stem "*Choose a President who was born on Feb 12th 1809.*" is more difficult to answer than "*Choose an American President who was born on Feb 12th 1809.*" This is because, the answer-space of (some of) the conditions in the former question is broader than the answer-space of (some of) the conditions in the latter. The answer-space of the condition *Choose a President* in the first stem, is larger than the condition *Choose an American President* in the second stem. Being a more generic concept (unary predicate) than `AmericanPresident`, the concept `President`, when used in a stem, makes the question difficult to answer. Therefore, a practical approach to make a stem difficult is by incorporating a predicate p_1 which is present for large number of individuals, along with a predicate p_2 which is present only for comparatively less number of instances, so that p_1 may deviate the learner away from the correct answer and p_2 may direct her to the correct answer.

The predicate combinations of such type can be easily identified by finding those property sequences with less triviality score; this is because, all the predicate combinations with at least one specific predicate and at least one generic predicate, will have a less PSTS. But, having a less PSTS does not always guarantee that one predicate in it is generic when compared to the other roles in the property sequence; the following condition also needs to be satisfied.

$$\exists\, p_1, p_2 \in \mathcal{P} \text{ such that } \#I(p_1) >> \#I(p_2) \quad (1)$$

In the condition, $\mathcal{P}$ represents the property sequence and $\#I(p)$ denotes the number of individuals satisfying the property p.

The tuples that satisfy Condition-1, can be assigned a difficulty-score based on its triviality score as shown in Eq. 2, where $\mathcal{P}_t$ denotes the property sequence corresponding to the tuple t. The equation guarantees that a tuple with a high PSTS value will get a low difficulty-score and vice versa. We consider the difficulty-score of a question as a constant value[17], if its property sequence does not satisfy Condition-1 .

$$\textit{Difficulty}(t) = \frac{1}{e^{\text{PSTS}(\mathcal{P}_t)}} \quad (2)$$

In addition to the above method to find tuples (or questions) which are difficult to answer, the difficulty-score of a question can be further increased (or tuned)

[17] A difficulty-score of 0.3 is given, since the maximum value of PSTS is 1, and the minimum possible value from Eq. 2 is 0.368.

by indirectly addressing the individuals present in it. We have already illustrated this in Section 4.2. Patterns 5 b, 6, 8 b, 9 a, 10 a, 10 b, 11 a, 11 c, 12 and 13 — where indirect addressing of the reference-individuals can be done — in Table 2, can be used for generating questions (or tuples) which are comparatively difficult to answer than those generated using the rest of the patterns. For such tuples, we simply double their assigned difficulty-scores, to make their difficulty-scores relatively higher than the rest of the tuples.

As we pointed out in Section 2.2.1, the Aggregation-based questions are relatively difficult to answer than the rest of the questions. Therefore, we give them a difficulty-score of thrice the value obtained using Eq. 2, by giving their base-tuples as input.

SELECT-TUPLE-SET(G, *DiffLevel*)
// **Input:** $G(V, E)$, the graph;
DiffLevel $\in$ { *high, medium, low*}
// **Output:** S, set of suitable tuples
$S \leftarrow \{v \mid$ Degree of v in G is zero$\}$
Priority-Queue Q // Priority is based on diff. level
Vertex u
Vertex-Set A
$Q \leftarrow$ CREATE-PQUEUE(G)
While Edge(G) is not empty
  if *DiffLevel* == *high*
    $u \leftarrow$ getMax(Q)
    removeMin()
  if *DiffLevel* == *low*
    $u \leftarrow$ getMin(Q)
    removeMax()
  if *DiffLevel* == *medium*
    if odd iteration
      $u \leftarrow$ getMin(Q)
      removeMin()
    if even iteration
      $u \leftarrow$ getMax(Q)
      removeMax()
  if NO-CONFLICT(u, S, G) == *true*
    $S \leftarrow S \cup \{u\}$
    $A \leftarrow \{u\} \cup$ AdjacentVertices(u, G)
    // finding all adj. vertices of u, in G
    RemoveVertex(A, G)
    // remove $w \in A$, from G
    $Q \leftarrow$ CREATE-PQUEUE(G, Q)
return S

8. Controlling the difficulty-level of a question-set

Controlling the difficulty-level of a question-set helps in posing only those set of questions which are necessary to test a learner's skill-set. In an e-learning system's environment, for the tasks such as controlling the student-shortlisting criteria, selection of top k students etc., question-sets of varying difficulty-levels are of great use [25].

In the E-ATG system, we adopted a simple algorithm to generate question-sets of difficulty-levels: high, medium and low. This algorithm can be further extended to generate question-sets of required difficulty-levels.

CREATE-PQUEUE(G)
// **Input:** $G(V, E)$, the graph
Priority-Queue Q
for each $v \in V$
  Priority $p \leftarrow$ *Difficulty(v)*
  put(v, p, Q) // insert v to Q
return Q

NO-CONFLICT(u, S, G)
// **Input:** $G(V, E)$, the graph;
Vertex-set S; Vertex u
for each $s \in S$
  if $(s, u) \in E$
    return *false*
    // $S \cup \{u\}$ is not independent-set
return $true$ // $S \cup \{u\}$ is independent-set

8.1. Method

The set of heuristically selected tuples (denoted as $T = \{t_1, t_2, ..., t_n\}$) can be considered as the vertices of an undirected graph (similar to what we have considered in Section 6.1.3) $G = (V, E)$ with vertex-set $V = \{t \mid t \in T\}$, and edge-set $E = \{(t_1, t_2) \mid t_1, t_2 \in Similarity(t_1, t_2) \geq mgs\}$, where *Similarity*(.) is same as that of what we have defined in Section 6.1.3, and mgs denotes the minimum similarity threshold (a.k.a. *minimum global-similarity threshold*).

An edge in G can be thought of as the inter-similarity (or dependency) of tuples that are taken from two locally similar groups. Ideally, we only need to include one among those dependent vertices, for generating a question-set which is not biased to a portion of the domain-knowledge. To generate an unbiased question-set which covers the relevant knowledge boundaries, we need to include all isolated ver-

tices (tuples) and one from each of the dependent vertices. Clearly, this vertex selection process is similar to, finding the *maximal independent-set* of vertices from G. To recall, a *maximal independent-set* of a graph $G = (V, E)$ is a subset $V^{'} \subseteq V$ of the vertices such that no two vertices in V are joined by an edge in E, and such that each vertex in $V - V^{'}$ is joined by an edge to some vertex in $V^{'}$.

In our implementation, we use the procedure SELECT-TUPLE-SET to find the suitable tuples for question-set generation. SELECT-TUPLE-SET is a greedy method in which the selection of the vertices that are to be included in the final-set is carefully prioritized to generate question-sets of high, medium and low difficulty-levels. For generating question-set of high difficulty-level, we choose vertices from the top of a (double-ended priority) queue, where the elements are in the sorted (in decreasing) order of their difficulty-level. To generate a question-set of low difficult-level, vertices are selected from the bottom of the queue. For medium difficulty-level question-set, vertices are alternatively chosen from top and bottom.

9. Generation of distractors

Distractors (or distracting answers) form a main component which determines the quality and difficulty-level of an MCQ item [37]. Selection of distractors for a stem is a time consuming as well as a skillful task. In the ATG system, we utilized a simple automated method for distractor generation. We have adopted the same distractor generation module in the E-ATG system. In the E-ATG system, we considered the difficulty-score of an MCQ due to its stem features alone, in generating a question-set of a required difficulty-level. In the current work, we used the distractor generation only as the functionality of the last stage module, where the selection of distractors is done with an intention to further tune the difficulty-level calculated by the preceding stage.

9.1. Method

Distractors are generated by subtracting the *actual answers* from the *possible answers* of the question. By actual answers, we mean those instances in the ontology which satisfy the conditions (or restrictions) given in the stem. Consider A as the set of actual answers corresponding to the stem. And, the possible answers correspond to the potential-set (see Table 2 for details) of the tuple.

The set of distractors of a tuple t with k as the *key* and q as the corresponding question-template is defined as:

$$Distractor(t, k, q) = Poten.Set(t) - A \quad (3)$$

In Eq. 3, $Poten.Set(t)$ denotes the potential-set of the tuple t, and is defined as $Type(Q_t, \mathcal{P}_t, k)$ (see Section 6.1.1), where Q_t and $\mathcal{P}_t$ denote the question-template and the property sequence respectively of t. If this equation gives a null set or a lesser number of distractors when compared to the required number of options, we can always choose any instance or datatype value other than those in $Poten.Set(t)$ as a distractor. This is represented in the following equation, where U is the whole set of individuals and datatype values in the ontology. The distractors generated using the following equation — denoted as $Distractor_{appro.}$ — are considered to be farther from the key than those generated using Eq. 3.

$$Distractor_{appro.}(t, k, q) = U - Poten.Set(t) \quad (4)$$

For the Aggregation-based questions, we find the distractors in the same manner.

10. Evaluation

The proposed E-ATG system produces a required number of MCQ items that can be edited and used for conducting an assessment. In this section, (in *Experiment-1*) we first evaluate how effectively our heuristics help in generating question-sets which are closer to those prepared by domain experts; secondly (in *Experiment-2*), we correlate the predicted difficulty-levels of the stems (obtained by the method given in Section 7) with their (actual) difficulty-levels which are estimated using Item Response Theory in a classroom set-up.

Implementation: The implemented prototype of the E-ATG system has the following modules:

- Tuple generation (using 19 question-templates) module[18]
- Heuristics based tuple selection module

[18] Generates tuples (or base-tuples) for the stem generation of both Pattern-based and Aggregation-based MCQs

- Stem difficulty estimation module
- Difficulty controlled question-set generation module
- Distractor generation module

The prototype of the system was developed using Java Runtime Environment JRE v1.6, the OWL API[19] v3.1.0 and FaCT++ [34].

Equipment description: The following machine was used for the experiments mentioned in this paper: Intel Quad-core i5 3.00 GHz processor, 10 GB 1333 MHz DDR3 RAM, running Ubuntu 13.04.

10.1. Experiment-1: Evaluation with the benchmark question-sets

Our objective is to evaluate how close the question-sets generated by our approach (a.k.a. Automatically generated question-sets or AG-Sets) are to the benchmark question-sets.

Datasets: We considered the following ontologies for generating question-sets.

1. Data Structures and Algorithms (DSA) ontology: models the aspects of Data Structures and Algorithms.
2. Mahabharata (MAHA) ontology: models the characters of the epic story of Mahabharata.
3. Geography (GEO) ontology[20]: models the geographical data of the United States. Ray Mooney and his research group, from the University of Texas, have developed this ontology.

The specifications of these test ontologies are given in Table 3. The DSA ontology and MAHA ontology were developed by our research group — Ontology-based Research Group[21] — at Indian Institute of Technology Madras, and are available at our web-page[22].

Benchmark question-set preparation: As a part of our experiment, experts of the domains of interest were asked to prepare question-sets from the knowledge formalized in the respective ontologies, expressed in the English language. The experts of the domains were selected such that they were either involved in the development of the respective ontologies (as domain experts) or have a detailed understanding of the knowledge formalized in the ontology. The question-sets prepared by the domain experts are referred from now on as the *benchmark question-sets* (abbreviated as BM-Sets). A minimum of two and a maximum of three experts were involved in each of the BM-Sets' preparation. The BM-sets contain only those questions which are mutually agreed by all the experts who are considered for the specific domain.

The domain experts prepared three BM-Sets — namely, Set-A, Set-B and Set-C — each for the three domains. Set-A, Set-B and Set-C contain 25, 50 and 75 question items respectively. They took around 8 hours (across a week) each to come up with question-sets for the three domains. More details about the benchmark question selection process can be found at our project web-page[23].

For each domain, the AG-Sets corresponding to the prepared BM-Sets — Set-A, Set-B and Set-C — are generated by giving the question-set sizes 25, 50 and 75 respectively as input to the E-ATG system, along with the respective ontologies.

10.1.1. Automated question-set generation

In the screening heuristics that we discussed in Section 6, there are three parameters which help in controlling the final question count: T_p (max. triviality score threshold), I (number of important concepts) and mls (min. local-similarity score threshold for a locally-similar group). Also, the parameter mgs (min. global-similarity score threshold) discussed in Section 8 is effectively chosen to manage the question count. Our system calculates appropriate values for each of these parameters in a sequential manner. First, the T_p is fixed, to limit the number of common property patterns in the result; then, the I is determined to select only those questions which are related to the most important domain concepts. After that, the parameters mls and mgs are fixed to avoid questions which are semantically similar.

Question-sets of required sizes ($Count_{Req} = 25, 50$ and 75) are generated by finding suitable values for each of the four ontology specific parameters, using the following approximation method.

The parameters T_p and I are not only ontology specific but also specific to each of the 19 patterns. For each pattern, the system chooses a suitable value for T_p (T'_p) such that the first screening process will generate

[19] http://owlapi.sourceforge.net/

[20] https://files.ifi.uzh.ch/ddis/oldweb/ddis/research/talking-to-the-semantic-web/owl-test-data/ (last accessed 26th Jan 2016)

[21] https://sites.google.com/site/ontoworks/home

[22] https://sites.google.com/site/ontoworks/ontologies

[23] https://sites.google.com/site/ontomcqs/research

Table 7
The cardinalities of the AG-Sets and the computational time taken for generating the question-sets are given below.

$Count_{Req}$	Ontology	#AG-Set	Time in Minutes
25	MAHA	47	3.52
	DSA	44	2.55
	GEO	28	6.29
50	MAHA	61	4.22
	DSA	81	3.33
	GEO	61	7.01
75	MAHA	123	4.54
	DSA	118	4.02
	GEO	93	7.43

a tuple-set whose cardinality is relatively larger than the required count. For the current experiments, the system has chosen a T'_p such that it generated (nearly) thrice the required count ($Count_{Req}$). Considering a higher T'_p can increase the variety of property combinations in the final tuple-set. In the second level of screening, the system chooses an I value (I'), which reduces the tuple-set to the required size. Since the system is repeating this procedure for all 19 patterns, a total question count of approximately 19×25 (for $Count_{Req} = 25$) or 19×50 (for $Count_{Req} = 50$) or 19×75 (for $Count_{Req} = 75$) will be generated. Then, the system varies the min. similarity scores mls and mgs to generate a tuple-set of cardinality approximately equal to $Count_{Req}$. mls and mgs give fine and coarse grained control over the result set's cardinality. For Experiment-1, we have generated AG-sets of *medium* difficulty-level, from the E-ATG system. Table 7 shows the count of questions filtered using suitable parameter values from our three test ontologies.

Overall cost: To find the overall computation time, we have considered the time required for the tuple generation (using 19 patterns), the time required for heuristics based question selection process and time required for controlling the difficulty-level of the question-set. The time required for the distractor generations and further tuning of the difficulty were not considered, since, we were focusing only on the proper selection of MCQ stems. The overall computation time taken for generating the AG-sets from each of the three test ontologies are given in Table 7.

10.1.2. AG-Sets Vs. BM-Sets

We have used the evaluation metrics: *precision* and *recall* (as used in [35]), for comparing two question-sets. This comparison involves finding the semantic similarity of questions in one set to their counterpart in the other.

To make the comparison precise, we have converted the questions in the BM-Sets into their corresponding tuple representation. Since, AG-Sets were already available in the form of tuple-sets, the similarity measure which we used in Section 6.1.3 is adopted to find the similar tuples across the two sets. For each of the tuples in the AG-Sets, we found the most matching tuple in the BM-Sets, thereby establishing a mapping between the sets. We have considered a minimum similarity score of 0.5 (ensuring partial similarity) to count the tuples as matching ones.

After the mapping process, we calculated the *precision* and *recall* of the AG-Sets, to measure the effectiveness of our approach. The precision and recall were calculated in our context as follows:

$$\text{Precision} = \frac{\text{Number of mapped tuples in the AG-Set}}{\text{Total number of tuples in the AG-Set}} \quad (5)$$

$$\text{Recall} = \frac{\text{Number of mapped tuples in the BM-Set}}{\text{Total number of tuples in the BM-Set}} \quad (6)$$

It should be noted that, according to the above equations, a high precision does not always ensure a good question-set. The case where more than one question in an AG-Set matching the same benchmark candidate is such an example. Therefore, the recall corresponding to the AG-Set (which gives the percentage of the number of benchmark questions that are covered by the AG-Set) should also be high enough for a good question-set.

Results: Table 8 shows the precision and recall of the question-sets generated by the proposed approach as well as the random-selection method[24], calculated against the corresponding benchmark question-sets: Set-A, Set-B and Set-C. A comparison with the random-selection method is provided, as the conventional question generation systems normally use random selection of questions for test generation [35].

The evaluation shows that, in terms of precision values, the AG-Sets generated using our approach are significantly better than those generated using random method. The recall values are in an acceptable range ($\approx 50\%$). We avoid a comparison with those question-sets that were generated by the ATG system [35], since

[24] Selecting required number of question-items randomly from a pool of questions.

the system does not generate the Aggregation-based questions, and the Pattern-based questions involving more than two predicates.

Table 8
The precision and recall of the question-sets generated by the proposed approach and the random-selection method, calculated against the corresponding benchmark question-sets

$Count_{Req}$	Ontology	Our approach		Random selectn.	
		Prec.	*Rec.*	*Prec.*	*Rec.*
25	MAHA	0.72	0.80	0.17	0.04
	DSA	0.77	0.76	0.22	0.11
	GEO	0.82	0.52	0.14	0.10
50	MAHA	0.91	0.55	0.11	0.11
	DSA	0.82	0.81	0.11	0.09
	GEO	0.91	0.47	0.19	0.11
75	MAHA	0.92	0.62	0.24	0.04
	DSA	0.82	0.74	0.13	0.08
	GEO	0.93	0.43	0.21	0.13

10.2. Experiment-2: Evaluation of stem difficulty

One of the core functionalities of the presented E-ATG system is its ability to determine the difficulty-scores of the stems it generates. To evaluate the efficacy of this functionality, we have generated test MCQs from a handcrafted ontology, and determined their difficulty-levels (a.k.a *predicted difficulty-levels*) (of the stems), using the method proposed in Section 7 and using statistical methods. Then, we compared these predicted difficulty-levels with their *actual difficulty-levels* that were estimated using principles in the Item Response Theory.

10.2.1. Estimation of actual difficulty-level

Item Response Theory is an item oriented theory which specifies the relationship between learners' performance on test items and their ability which is measured by those items. In IRT, *item analysis* is a popular procedure which tells if an MCQ is too easy or too hard, and how well it discriminates students of different knowledge proficiency. Here, we have used item analysis to find the actual difficulty-levels of the MCQs.

Our experiment was based on the simplest IRT model (often called *Rasch model* or the *one-parameter logistic model* (1PL)). According to this model, we can predict the probability of answering a particular item correctly by a learner of certain knowledge proficiency level (a.k.a *trait level*), as specified in the following formula.

$$Probability = \frac{e^{(proficiency - difficulty)}}{1 + e^{(proficiency - difficulty)}} \quad (7)$$

A detailed theoretic background of the 1PL model is provided in Appendix A. To find the (actual) difficulty value, we can rewrite the Eq.7 as follows:

$$difficulty = proficiency - log_e(\frac{Probability}{1 - Probability}) \quad (8)$$

From now on, we use α and θ for (actual) *difficulty* and *proficiency* respectively. For experimental purpose, suitable θ values can be assigned for high, medium and low trait levels. Given the probability (of answering an MCQ correctly by learners) of a particular trait level, if the calculated α value is (approximately) equal or greater than the θ value, we can assign the trait level as its *actual difficulty-level*.

10.2.2. Experiment setup

A controlled set of question stems from the DSA ontology has been used to obtain evaluation data related to its quality. These stems were then associated with a set of distractors which is selected under the similarity based theory [5] such that all the test MCQs will have same difficulty-level w.r.t. their choice set. This is done to feature the significance of stem difficulty, rather than the overall MCQ difficulty involving the difficulty-level due to the choice set.

Test MCQs and instructions: We have employed a question-set of 24 test MCQs each to participants, with the help of a web interface. Appendix B lists the stems of the MCQs that are used in our study. These 24 stems were chosen such that, the test contains 8 MCQs each of high, medium and low (predicted) difficulty-levels. The difficulty-scores of these stems were predetermined using the method detailed in Section 7. Difficulty-levels (predicted difficulty-levels) were then assigned by statistically finding three equal intervals (corresponding to low, medium and high) from the obtained difficulty-scores of all the stems. All the test MCQs were carefully vetted by human-editors to correct grammatical and punctuation errors, and to capitalize the proper nouns in the question stems. Each MCQ contains choice set of cardinality four (with exactly one key) and two additional options: SKIP and INVALID. A sample MCQ is shown in Example-2.

Example 2 Choose an Internal Sorting Algorithm with worse case time complexity n exp 2.

Options
a. Heap Sort
b. In-order Traversal
c. Bubble Sort
d. Breadth First Search
e. SKIP
f. INVALID

The responses from (carefully chosen) 54 participants — 18 participants each with high, medium and low trait levels — were considered for generating the statistics about the item quality. The following instructions were given to the participants before starting the test.

1. The test should be finished in 40 minutes.
2. All questions are mandatory.
3. You may tick the option "SKIP" if you are not sure about the answer. Kindly avoid guess work.
4. If you find a question invalid, you may mark the option "INVALID".
5. Avoid use of the web or other resources for finding the answers.
6. In the end of the test, you are requested to enter your expert level in the subject w.r.t this test questions, in a scale of high, medium or low. Also, kindly enter your grade which you received for the ADSA course offered by the Institute.

Participant selection: Fifty four learners of the required knowledge proficiencies were selected from a large number of graduate level students (of IIT Madras), who have participated in the online MCQ test. To determine their trait levels, we have instructed them to self assess their knowledge-confidence level on a scale of high, medium or low, at the end of the test. To avoid the possible errors that may occur during the self assessment of trait levels, the participant with high and medium trait levels were selected from only those students who have successfully finished the course: CS5800: *Advanced Data Structures and Algorithms*, offered at the computer science department of IIT Madras. The participants with high trait level were selected from those students with either of the first two grade points[25] (i.e., 10 - Excellent and 9 - Very Good). The participants with medium trait level were from those students who were having any of the next two grade points (i.e., 8 - Good and 7 - Satisfactory Work).

The evaluation data collected for the item analysis is shown in Table 10 and Table 11.

10.2.3. Item analysis

The probabilities of correctly answering the test MCQs (represented as P) by the learners are listed in Table 10. In the table, the learner sets L_1, L_2 and L_3 correspond to the learners l_1 to l_{18}, l_{19} to l_{36} and l_{37} to l_{54} respectively. The learners in these learner sets have high, medium and low trait levels respectively. The probability P (a.k.a., P_{qr}) of correctly answering an MCQ q for each of the learner sets (L_r) are obtained using the following formula.

$$P_{qr} = \frac{\#\text{Learners in } L_r \text{ who have correctly answered } q}{|L_r|}$$

While calculating the P values, if a learner has chosen the option “SKIP” as the answer, the MCQ is considered as wrongly answered by her. If she has chosen “INVALID”, we do not consider her poll for calculating P.

Table 11 shows the α_i (actual difficulty) values that we have calculated using the P values given in Table 10. Eq.8 is used for finding the *difficulty* values. These α_i values were then used to assign the actual difficulty-level for the MCQs.

We are particularly interested in the highlighted rows in Table 11, where an MCQ can be assigned an actual difficulty-level as shown in Table 9. That is, for instance, if the trait level of a learner is high and α_i is approximately equal to θ_l (ideally, $\alpha_i \geq \theta_l$), then, according to the IRT model, a difficulty-level of high can be assigned. In our experiments, to calculate α_i values for high, medium and low trait levels, we used θ_l values $1.5, 0$ and -1.5 respectively.

Table 9
Thumb rules for assigning difficulty-level

Trait level	α_i	Difficulty-level
High	(> 1.5) or $(\approx 1.5 \pm .45)$	High
Medium	(> 0) or $(\approx 0 \pm .45)$	Medium
Low	(> -1.5) or $(\approx -1.5 \pm .45)$	Low

[25] https://en.wikipedia.org/wiki/Indian_Institute_of_Technology_Madras

10.2.4. Results

Figure 3 shows the statistics that can be concluded from our item analysis.

The test MCQs i_1 to i_8, except i_5, i_6 and i_7, have high difficulty-level, as predicted by our approach. The question items, i_9 to i_{16} except i_{10}, i_{12} and i_{13} have medium difficulty-level — showing 63% correlation with their predicted difficulty-levels. The MCQ items i_{16} to i_{24} have low difficulty-level, as predicted, with 88% correlation.

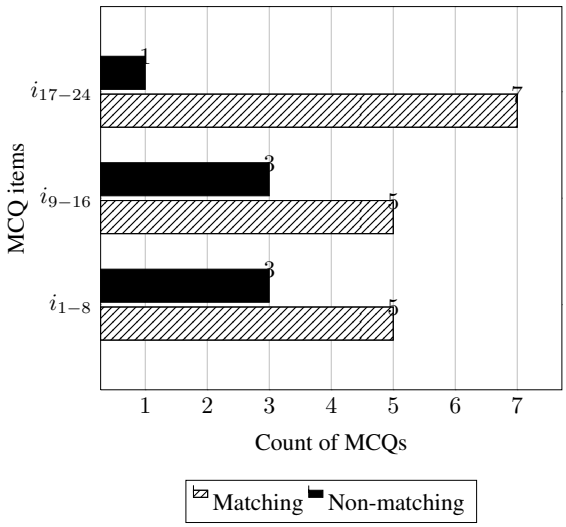

Fig. 3. The figure shows the count of MCQs whose actual difficulty-levels are matching (and not matching) with the predicted difficulty-levels.

In our observation, the repetition of similar words or part of a phrase, in an MCQ's stem and its key, is one of the main reason for the unexpected behavior of its predicted difficulty-levels. This word repetition can give a hint to the learner, enabling her to choose the correct answer. Example-3 shows the MCQ item i_7, where the repetition of the word "string" in the stem and the key, has degraded its (actual) difficulty-level.

Example 3 Choose a string matching algorithm which is faster than Robin-Karp algorithm.

Options	
a.	Selection sort
b.	Boyer Moore string search algorithm (Key)
c.	Makeset
d.	Prims algorithm
e.	SKIP
f.	INVALID

Table 10

The probabilities of correctly answering the test MCQs (P values) are shown below. Learners in L_1, L_2 and L_3 are having high medium and low domain knowledge proficiencies respectively. MCQs i_1 to i_8, i_9 to i_{16} and i_{17} to i_{24} have predicted difficulty-levels high, medium and small respectively.

MCQ item No.	Learner Set L_1	L_2	L_3
i_1	0.44	0.33	0.06
i_2	0.55	0.39	0.11
i_3	0.50	0.28	0.06
i_4	0.55	0.39	0.00
i_5	0.78	0.50	0.11
i_6	0.72	0.44	0.11
i_7	0.94	0.61	0.39
i_8	0.55	0.35	0.07
i_9	0.94	0.61	0.06
i_{10}	1.00	0.67	0.00
i_{11}	0.94	0.56	0.00
i_{12}	1.00	0.65	0.06
i_{13}	0.94	0.72	0.11
i_{14}	1.00	0.50	0.00
i_{15}	0.89	0.44	0.00
i_{16}	1.00	0.39	0.00
i_{17}	1.00	0.89	0.39
i_{18}	1.00	0.94	0.22
i_{19}	1.00	1.00	0.78
i_{20}	1.00	0.94	0.50
i_{21}	1.00	0.94	0.22
i_{22}	1.00	0.94	0.06
i_{23}	0.94	0.78	0.06
i_{24}	1.00	0.94	0.11

11. Related work

In the literature, there are several works such as [2,27,17,8,3,38,7] that are centering on the problem of *question generation from ontologies*. These works have addressed the problem w.r.t. specific applications. Some of these applications that were widely accepted by the research communities are question driven ontology authoring [30], question generation for educational purpose [19,4,35], and generation of questions for ontology validation [1]. For a detailed review of related literature, the interested readers are required to refer to [19,4].

Ontologies with potential educational values are available in different domains. However, it is still unclear how such ontologies can be fully exploited to generate useful assessment questions. Experiments in [2,35] show that question generation from asser-

Table 11
The α_i values calculated using the obtained P values.

MCQ item No.	Learner Set L_1	L_2	L_3
i_1	1.74	0.71	1.25
i_2	1.30	0.45	0.59
i_3	1.50	0.94	1.25
i_4	1.30	0.45	$+\infty$
i_5	0.23	0.00	0.59
i_6	0.56	0.24	0.59
i_7	-1.25	0.45	-0.23
i_8	1.30	0.62	1.09
i_9	-1.25	-0.45	1.25
i_{10}	$-\infty$	-0.71	$+\infty$
i_{11}	-1.25	-0.24	$+\infty$
i_{12}	$-\infty$	-0.62	1.25
i_{13}	-1.25	-0.94	0.59
i_{14}	$-\infty$	0.00	$+\infty$
i_{15}	-0.59	0.24	$+\infty$
i_{16}	$-\infty$	0.45	$+\infty$
i_{17}	$-\infty$	-2.09	-1.05
i_{18}	$-\infty$	-2.75	-0.23
i_{19}	$-\infty$	$-\infty$	-2.77
i_{20}	$-\infty$	-2.75	-1.50
i_{21}	$-\infty$	-2.75	-0.23
i_{22}	$-\infty$	-2.75	1.25
i_{23}	-1.25	-1.27	1.25
i_{24}	$-\infty$	-2.75	0.59

tional facts (ABox axioms) is useful in conducting factual-MCQ tests. These factual-MCQs, considered to be the first level of learning objectives in Bloom's taxonomy [12], are well accepted in the educational community for preliminary and concluding assessments.

When it comes to generating questions from ABox axioms, pattern-based methods are of great use. But the pattern-based approaches such as [6,1,28,39,30] use only a selected number of patterns for generating questions. A detailed study of all the possible type of templates (or patterns) is not explored by any research group. Also, since the applicability of these pattern-based approaches is limited due to the enormous number of generated questions, there was also a need for suitable mechanisms to select relevant question items or to prevent the generation of useless questions. We have addressed this issue in this paper by proposing a set of heuristics that mimic the selection process by a domain expert. Existing approaches select relevant questions for assessments by using techniques similar to text summarization. For instance, in ASSESS[15] (Automatic Self-Assessment Using Linked Data), the authors find the properties that are most frequently used in combination with an instance as the relevant properties for question framing. Sherlock[24] is another semi-automatic quiz generation system which is empowered by semantic and machine learning technologies. Educationalists are of the opinion that the question-sets that are generated using the above mentioned existing approaches, are only good for conducting quiz-type games, since they do not satisfy any pedagogical goals. In the context of setting up assessments, a pedagogical goal is to prepare a test which can distinguish learners having a particular knowledge proficiency-level, by controlling the distribution of difficult questions in the test. Our focus on these educational aspects clearly distinguishes our work from the existing approaches. Furthermore, Williams [36] presented a prototype system for generating mathematical word problems from ontologies based on predefined logical patterns, which are very specific to the domain. Our research is focused on proposing generic (domain independent) solutions for assessment generation. Another feature which distinguishes our work from the rest of the literature is our novel method for determining the stem-difficulty.

12. Conclusion and future work

In this paper, we proposed an effective method to generate MCQs for educational assessment-tests from formal ontologies. We also give the details of a prototype system (E-ATG system) that we have implemented incorporating the proposed methods. In the system, a set of heuristics were employed to select only those questions which are required for conducting a domain related test. A method to determine the difficulty-level of a question-stem and an algorithm to control the difficulty of a question-set were also incorporated in the system. Effectiveness of the suggested question selection heuristics was studied by comparing the resulting questions with those questions which were prepared by domain experts. The correlation of the difficulty-levels of the questions which were assigned by the system, to their actual difficulty-levels, was empirically verified in a classroom-setup using Item Response theory principles.

The system generated MCQs have undergone an editing phase, before they were used in the empirical study. The editing works that have been taken care by

human-editors include: correcting grammatical errors in the stem, removing those stems with words that are difficult to understand, correcting the punctuation in the stem and starting proper nouns with capital letters. As part of future work, it would be interesting to add a module to the E-ATG system that can carry out these editing tasks.

Grammatical inconsistencies and word repetitions between stem, key and distractors are some issues that are not addressed in this work. For example, if the distractors of an MCQ are in singular number, and if the key and stem are in plural number; no matter what the difficulty-level of the MCQ, a learner can always give the correct answer. In an assessment test, if these grammatical issues are not addressed properly, the MCQs may deviate from their intended behavior and can even confuse the learners.

Currently, we described only a method to find the Aggregation-based questions (a sub-category of Non-Pattern-based questions) which can be generated from an ontology; it is an open question as to how to automatically extract all possible Non-Pattern-based questions.

In this paper, we focus more on the educational relevance of the assessment tests which are generated by the proposed system, rather than the scalability and performance of the system. In future, we intend to enhance the current implementation of the system, to include several in-memory and hard drive-driven caching solutions, so that the system can scale to large knowledge bases.

13. Acknowledgements

We express our gratitude to IIT Madras and the Ministry of Human Resource, Government of India, for the funding to support this research. A part of this work was published at the 28th International FLAIRS Conference (FLAIRS-28). We are very grateful for the comments given by the reviewers. We would like to thank the AIDB Lab (Computer Science department, IIT Madras) members and students of IIT Madras who have participated in the empirical evaluation. In particular, we would like to thank Kevin Alex Mathew, Rajeev Irny, Subhashree Balachandran and Athira S, for their constant help and involvement in various phases of the project.

References

[1] Asma Ben Abacha, Marcos Da Silveira, and Cédric Pruski. Medical ontology validation through question answering. In *AIME*, pages 196–205, 2013.

[2] Maha Al-Yahya. Ontology-based multiple choice question generation. *The Scientific World Journal*, Vol 2014, page 9, ID: 10.1155/2014/274949, 2014.

[3] Maha M. Al-Yahya. Ontoque: A question generation engine for educational assesment based on domain ontologies. In *ICALT*, pages 393–395. IEEE Computer Society, 2011.

[4] T. Alsubait. *Ontology-based multiple-choice question generation.* PhD thesis, School of Computer Science, The University of Manchester.

[5] T. Alsubait, B. Parsia, and U. Sattler. A similarity-based theory of controlling mcq difficulty. In *e-Learning and e-Technologies in Education (ICEEE), 2013 Second International Conference on*, pages 283–288, Sept 2013.

[6] Tahani Alsubait, Bijan Parsia, and Uli Sattler. Generating multiple choice questions from ontologies: Lessons learnt. In *Proceedings of the 11th International Workshop on OWL: Experiences and Directions (OWLED 2014) co-located with 13th International Semantic Web Conference on (ISWC 2014), Riva del Garda, Italy, October 17-18, 2014.*, pages 73–84, 2014.

[7] Tahani Alsubait, Bijan Parsia, and Ulrike Sattler. Next generation of e-assessment: automatic generation of questions. *International Journal of Technology Enhanced Learning*, 4:156–171.

[8] Tahani Alsubait, Bijan Parsia, and Ulrike Sattler. Mining ontologies for analogy questions: A similarity-based approach. volume 849 of *CEUR Workshop Proceedings*. OWL Experiences and Directions, 2012.

[9] Tahani Alsubait, Bijan Parsia, and Ulrike Sattler. Generating multiple choice questions from ontologies: Lessons learnt. volume 1265 of *CEUR Workshop Proceedings*. OWL Experiences and Directions, 2014.

[10] Davis Barbara Gross. *Tools for Teaching*. Jossey-Bass Inc., San Francisco, California, 1993.

[11] Benjamin Bloom and David R. Krathwohl, editors. *Taxonomy of educational objectives: The classi- fication of educational goals by a committee of college and university examiners. I: Handbook I: Cognitive domain*. Addison-Wesley, New York, 1956.

[12] Benjamin S. Bloom, Max D. Engelhart, Edward J. Furst, Walker H. Hill, and David R. Krathwohl. *Taxonomy Of Educational Objectives: Handbook 1, The Cognitive Domain*. Allyn & Bacon, Boston, 1956.

[13] Peter Brusilovsky and Philip Miller. Course delivery systems for the virtual university, 2001.

[14] Peter Brusilovsky and Philip L. Miller. Web-based testing for distance education. In *WebNet*, pages 149–155, 1999.

[15] Lorenz Bühmann, Ricardo Usbeck, and Axel-Cyrille Ngonga Ngomo. ASSESS — Automatic Self-Assessment Using Linked Data. In *International Semantic Web Conference (ISWC)*, 2015.

[16] Danilo Carvalho, Cagatay Calli, André Freitas, and Edward Curry. EasyESA: A Low-effort Infrastructure for Explicit Semantic Analysis (Demo). In *13th International Semantic Web Conference (ISWC 2014)*, Rival del Garda, 2014. Springer.

[17] Marija Cubric and Milorad Tosic. Towards automatic generation of e-assessment using semantic web technologies. In *Pro-*

ceedings of the 2010 International Computer Assisted Assessment Conference, 2010.

[18] Vladan Devedzic. Education and the Semantic Web. *International Journal of Artificial Intelligence in Education*, (14):165–191, 2004.

[19] Vinu E.V. and Sreenivasa Kumar P. A novel approach to generate MCQs from domain ontology: Considering DL semantics and open-world assumption. *Web Semantics: Science, Services and Agents on the World Wide Web*, pages –, 2015.

[20] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 1606–1611, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[21] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009.

[22] Ian Horrocks. OWL: A description logic based ontology language. In *Logic Programming, 21st International Conference, ICLP 2005, Sitges, Spain, October 2-5, 2005, Proceedings*, pages 1–4, 2005.

[23] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: the making of a web ontology language. *J. Web Sem.*, 1(1):7–26, 2003.

[24] Dong Liu and Chenghua Lin. Sherlock: a semi-automatic quiz generation system using linked data. volume 1272 of *CEUR Workshop Proceedings*, pages 9–12. CEUR-WS.org, 2014.

[25] Joseph Lowman. *Mastering the Techniques of Teaching*. Jossey-Bass, May 2000.

[26] R. Michael Furr and Verne R. Bacharach. *Psychometrics, An Introduction. Second Edition*. SAGE Publications, Inc, 2014.

[27] M.Tosic and M.Cubric. SeMCQ- Protege Plugin for Automatic Ontology- Driven Multiple Choice Question Tests Generation. In *Proceedings of the 11th International Protege Conference*, 2009.

[28] Shiyan Ou, Constantin Orasan, Dalila Mekhaldi, and Laura Hasler. Automatic question pattern generation for ontology-based question answering. In David Wilson and H. Chad Lane, editors, *FLAIRS Conference*, pages 183–188. AAAI Press, 2008.

[29] Silvio Peroni, Enrico Motta, and Mathieu dâĂŹAquin. Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures. In John Domingue and Chutiporn Anutariya, editors, *The Semantic Web*, volume 5367 of *Lecture Notes in Computer Science*, pages 242–256. Springer Berlin Heidelberg, 2008.

[30] Yuan Ren, Artemis Parvizi, Chris Mellish, Jeff Z. Pan, Kees van Deemter, and Robert Stevens. Towards Competency Question-driven Ontology Authoring. In *Proc. of 11th Conference on Extended Semantic Web Conference (ESWC 2014).*, 2014.

[31] John T. Sidick, Gerald V. Barrett, and Dennis Doverspike. Three-alternative multiple-choice tests: An attractive option. *Personnel Psychology*, Vol 47, Issue 4, pages 829âĂŞ835, 1994.

[32] Marielle Simon, Kadriye Ercikan, and Michel Rousseau. *Improving Large Scale Education Assessment: Theory, Issues, and Practice*. Routledge, New York, 2013.

[33] Samir Tartir, I. Budak Arpinar, Michael Moore, Amit P. Sheth, and Boanerges Aleman-meza. Ontoqa: Metric-based ontology quality analysis. In *IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources*, 2005.

[34] Dmitry Tsarkov and Ian Horrocks. FaCT++ description logic reasoner: System description. In *Proceedings of the International Joint Confererence on Automated Reasoning (IJCAR˜2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 292–297. Springer, 2006.

[35] Vinu E. V and P. Sreenivasa Kumar. Improving large-scale assessment tests by ontology based approach. In *Proceedings of the Twenty-Eighth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2015, Hollywood, Florida. May 18-20, 2015.*, page 457, 2015.

[36] Sandra Williams. Generating mathematical word problems. In *Question Generation, Papers from the 2011 AAAI Fall Symposium, Arlington, Virginia, USA, November 4-6, 2011*, 2011.

[37] Karyn Woodford and Peter Bancroft. Multiple choice questions not considered harmful. In *Proceedings of the 7th Australasian Conference on Computing Education - Volume 42*, ACE '05, pages 109–116, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[38] Konstantinos Zoumpatianos, Andreas Papasalouros, and Konstantinos Kotis. Automated transformation of swrl rules into multiple-choice questions. In R. Charles Murray and Philip M. McCarthy, editors, *FLAIRS Conference*. AAAI Press, 2011.

[39] Branko Å¡itko, Slavomir Stankov, Marko RosiÄĞ, and Ani GrubiÅąiÄĞ. Dynamic test generation over ontology-based knowledge representation in authoring shell. *Expert Systems with Applications*, 36(4):8185 – 8196, 2009.

Appendix

A. IRT model and Difficulty calculation

Item Response Theory (IRT) was first proposed in the field of psychometrics for the purpose of ability assessment. It is widely used in pedagogy to calibrate and evaluate questions items in tests, questionnaires, and other instruments, to score subjects based on the test takers abilities, attitudes, or other trait levels.

The experiment described in Section 10.2 is based on the simplest IRT model (often called *Rasch model* or the *one-parameter logistic model* (1PL)). According to this model, a learner's response to a question item[26] is determined by her knowledge proficiency level (a.k.a. *trait level*) and the difficulty of the item. 1PL is expressed in terms of the probability that a learner with a particular trait level will correctly an-

[26] 1PL considers binary item (i.e., true/false); since we are not evaluating the quality of distractors here, the MCQs can be considered as binary items which are either correctly answered or wrongly answered by a learner.

swer an MCQ that has a particular difficulty-level; this is represented in [26] as:

$$P(R_{li} = 1|\theta_l, \alpha_i) = \frac{e^{(\theta_l - \alpha_i)}}{1 + e^{(\theta_l - \alpha_i)}} \quad (9)$$

In the equation, R_{li} refers to response (R) made by learner l to MCQ item i (where $R_{li} = 1$ refers to a correct response), θ_l denotes the trait level of learner l, α_i represents the difficulty of item i. θ_l and α_i are scaled on a standardized metric, so that their means are 0 and the standard deviations are 1. $P(R_{li} = 1|\theta_l, \alpha_i)$ denotes the conditional probability that a learner l will respond to item i correctly. For example, the probability that a a below-average trait level (say, $\theta_l = -1.4$) learner will correctly answer an MCQ that has a relatively high hardness (say, $\alpha = 1.3$) is:

$$P = \frac{e^{(-1.4-1.3)}}{1 + e^{(-1.4-1.3)}} = \frac{e^{(-2.7)}}{1 + e^{(-2.7)}} = 0.063$$

In our experiment, we intended to find the α_i of the MCQ items with the help of learners, whose trait levels have been pre-determined as: high, medium or low. The corresponding P values are obtained by finding the ratio of the number of learners (in the trait level under consideration) who have correctly answered the item, to the total number of learners under that trait level. On getting the values for θ_l and P, the value for α_i was calculated using the Equation-10.

$$\alpha_i = \theta_l - log_e(\frac{P}{1 - P}) \quad (10)$$

In the equation, $\alpha_i = \theta_l$, when P is 0.50. That is, an MCQ's difficulty is defined as the trait level required for a learner to have 50 percent probability of answering the MCQ item correctly. Therefore, for a trait level of $\theta_l = 1.5$, if $\alpha_i \approx 1.5$, we can consider that the MCQ has a high difficulty-level. Similarly, for a trait level of $\theta_l = 0$, if $\alpha_i \approx 0$, the MCQ has medium difficulty-level. In the same sense, for a trait level of $\theta_l = -1.5$, if $\alpha_i \approx -1.5$, then MCQ has a low difficulty-level.

B. Sample MCQ stems

Table 12 shows a list of sample MCQ stems that are generated from the DSA ontology. In the table, the stems 1 to 8, 9 to 16 and 17 to 24 correspond to high, medium and low (predicted) difficulty-levels respectively. These stems along with their choice sets (please refer to our project website) were employed in the experiment mentioned in Section 10.2.

Table 12
Sample MCQ stems that are generated from the DSA ontology. Stems 1 to 8 have *high* predicted difficulty-levels, 9 to 16 have *medium* and 17 to 24 have *low* difficulty-levels.

Item No.	Stems of MCQs
1	Choose a polynomial time problem with application in computing canonical form of the difference between bound matrices.
2.	Choose an NP-complete problem with application in pattern matching and is related to frequent subtree mining problem.
3.	Choose a polynomial time problem which is also known as maximum capacity path problem.
4.	Choose an application of an NP-complete problem which is also known as Rucksack problem.
5.	Choose the one which operates on output restricted dequeue and operates on input restricted dequeue.
6.	Choose a queue operation which operates on double ended queue and operates on a circular queue.
7.	Choose a string matching algorithm which is faster than Robin-Karp algorithm.
8.	Choose the one whose worst time complexity is n exp 2 and with Avg time complexity n exp 2.
9.	Choose an NP-hard problem with application in logistics.
10.	Choose an all pair shortest path algorithm which is faster than Floyd-Warshall Algorithm.
11.	Choose the operation of a queue which operates on a priority queue.
12.	Choose the ADT which has handling process "LIFO".
13.	Choose an internal sorting algorithm with worse time complexity m plus n.
14.	Choose a minimum spanning tree algorithm with design technique greedy method.
15.	Choose an internal sorting algorithm with time complexity n log n.
16.	Choose an Internal Sorting Algorithm with worse time complexity n exp 2.
17.	Choose the operation of a file.
18.	Choose a heap operation.
19.	Choose a tree search algorithm.
20.	Choose a queue with operation dequeue.
21.	Choose a stack operation.
22.	Choose a single shortest path algorithm.
23.	Choose a matrix multiplication algorithm.
24.	Choose an external sorting algorithm.

C. Abbreviations and notations used

Table 13 lists the abbreviations and notations that are used in this paper.

Table 13
Abbreviations and notations used in this paper are listed below

Abbreviations and notations	Expansion
ATG	Automatic Test Generation
MCQ	Multiple Choice Question
FQ	Factual Question
PSTS	Property Sequence Similarity Score
IRT	Item Response Theory
MAHA ontology	Mahabharata ontology
DSA ontology	Data Structures and Algorithms ontology
GEO ontology	Geography ontology
ESA	Explicit Semantic Analysis
AG-Set	Automatically Generated questions set
BM-Set	Benchmark question set
MDS	Minimum Dominating Set
mgs	minimum global similarity threshold
mls	minimum local similarity threshold