Multidiszciplináris tudományok, 9. kötet. (2019) 4 sz. pp. 500-507 **https://doi.org/10.35925/j.multi.2019.4.53**

CHALLENGES IN TARGET SELECTION FOR AUTOMATIC QUESTION GENERATION

Walelign Tewabe Sewunetie
PhD student, University of Miskolc, Institute of Information Science
3515 Miskolc, Miskolc-Egyetemváros

László Kovács
professor, University of Miskolc, Institute of Information Science
3515 Miskolc, Miskolc-Egyetemváros, e-mail: kovacs@iit.uni-miskolc.hu

Abstract
Today's educational systems need an efficient tool to perform competently assessment of students on their major concepts they learnt from study material. Preparing a set of questions for assessment can be time consuming for teachers. Automatic Question Generation (AQG) is the technique for generating a right set of questions from a content. A key problem with AQG systems is the selection of the appropriate target sentences and concepts. In this paper, we investigate the difficulties in automatic target selection using unannotated document sources. As the result of the analysis, we identified the following key issues: hard to measure the topic relevance; the large segmentation of the information content and the incomplete background knowledge bases.

Keywords: *Automatic Question Generation, ontology, knowledge modeling*

1. Introduction

Intelligent Tutoring System (ITS) is a computer-based system that aims to offer direct and customized instruction or feedback to learners with personalized guidelines based on their cognitive skills, usually without requiring intervention from a human teacher [1]. According to Fletcher and Sottilare [2] , intelligent tutoring may be viewed as "an effort to capture in computer technology the capabilities and practices of a human instructor who is expert in both the subject matter and one-to-one tutoring". From the earliest days of computers, researchers have struggled to develop ITS that are as effective as human tutors [3].

Current ITS systems consist of five key modules (see Fig .1). The domain module contains the knowledge database of the learning material; the external knowledge database provides additional ontology support of general scope; the tutor module is to control and supervise the training process; the learner module provides a behavior model of the students and finally, the interface module relates to the technical aspects, containing the required hardware/software components.

Considering the tutor module, one of the key functions is to manage the assessment and test components to get feedback about the current knowledge level of the student. This feedback is very important input to the selection of the learning path.

In traditional e-learning systems, the question pool for assessment is generated manually by domain experts. This method is usually very time consuming and it has a very restricted flexibility. In intelligent tutoring systems, the questions are constructed automatically to provide a higher level of

adaptivity. The domain of automatic question generation (AQG) is an actively investigated area in the development of intelligent systems.

2. Automatic Question Generation Methods

Automatic question generation is the task of generating questions automatically from a given text. A lot of work has been done in this field with the help of various tools like Semantic Role Labeller, POS Tagger and Annotated corpora tools [5]. One key application of question generation is the area of education to generate questions for reading comprehension [6]. In addition to the above applications, question generation systems can aid the development of annotated data sets for natural language processing (NLP) research in reading comprehension and question answering.

Question generation depending upon the target complexity can be mainly categorized into two categories, deep question generation and shallow question generation. Deep QG generates deep questions that involve more logical thinking (such as why, why not, what-if, what-if-not and how questions) whereas shallow QG generates shallow questions that focus more on facts (such as who, what, when, where, which, how many/much and yes/no questions).

Recently, the research area of automatic question generation for educational purposes has attracted attention of researchers from different disciplines. According to [7], research on question generation has a long tradition and can be traced back to the application of logic to questions. While research on question generation has been conducted for a long time, deploying automatic question generation for educational purposes has raised interests in different research communities in recent years. Studies have reported that deploying questions in teaching encourages students to self-explain, which has been shown to be highly beneficial for learning [8].

Question generation has attracted the attention of the natural language generation (NLG) community in recent years, since the work of Rus et al [9]. To ask a natural question, people usually pay attention to certain parts of the input sentence, as well as associating context information from the paragraph. Recent related works try to model the conditional probability using RNN encoder-decoder architecture [10], and adopt the global attention mechanism [11] to make the model focus on certain elements of the input when generating each word during decoding. Here, we investigate two variations of our models: one that only encodes the sentence and another that encodes both sentence and paragraph level information.

For the most part, question generation has been tackled in the past via rule-based approaches [8]. The success of these approaches hinges critically on the existence of well-designed rules for declarative-to-interrogative sentence transformation, typically based on deep linguistic knowledge. To improve quality over a purely rule-based system, [6] introduced a generate-and-rank approach that generates multiple questions from an input sentence using a rule-based approach and then ranks them using a supervised learning-based ranker [5]. Although the ranking algorithm helps to produce more acceptable questions, it relies heavily on a manually crafted feature set, and the questions generated often overlap word for word with the tokens in the input sentence, making them very easy to answer.

However, most of existing approaches to question generation have focused on generating questions from a single sentence, relying heavily on syntax only shallow semantics [12]. A problem with this approach is that the majority of questions generated from single sentences tend to be too specific and low-level to properly measure learners' understanding of the overall contents of text. In other words, what is assessed by such question generation system ends up essentially being the ability to compare sentences, just requiring learners to find a single sentence that has almost the same surface form as a

given interrogative sentence. Results of simple sentence comparisons do little to contribute towards the goal of assessing learners' reading comprehension.

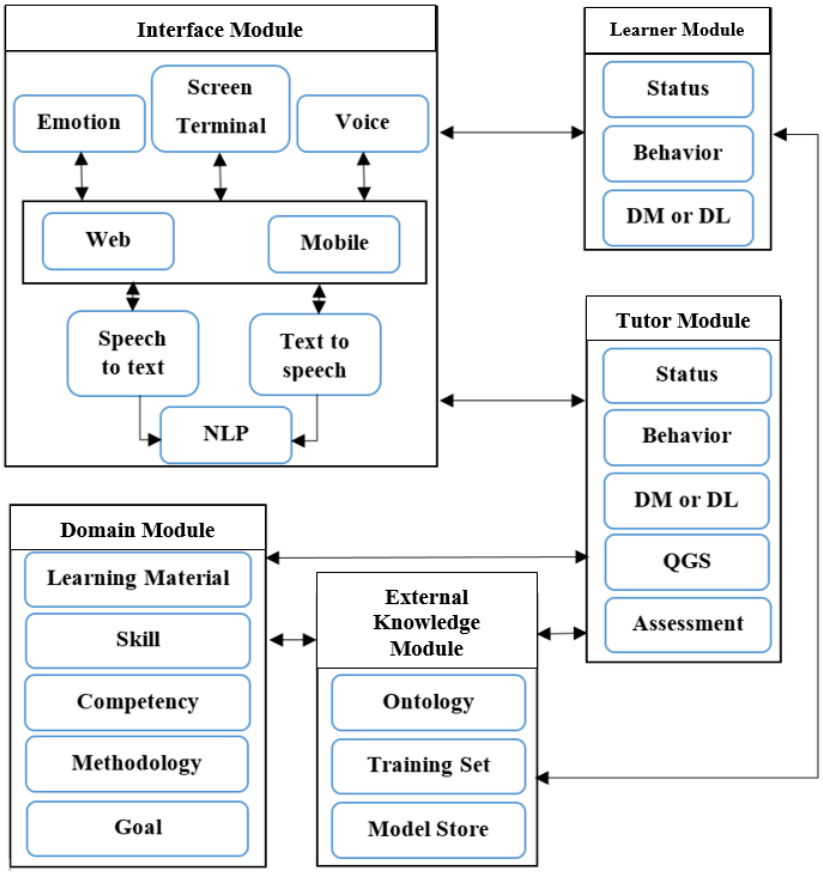


Fig 1, ITS architecture [4]

Considering the different question types, the multi-choice question is the most widely used question format in the AQG systems. The main benefit of this format is the simplicity of the evaluation and the relatively high unambiguity. In the construction of multi-choice questions, we face the following challenges:

- selection of the target sentence
- selection of the target concept / phrase
- selection of the distractors.

In our study, we focus now on these problems regarding the quality of the selected text components.

3. Target Selection Methods

Current researches are mainly focusing on sentence level. We present this approach by the methodology given in [13]. The engine selects elementary sentences without complex clauses. Using the Charniak parser [14], a syntactic tree is constructed for the selected sentence. Based on the

associated POS and NE tagged information, the subject, object, preposition and verb parts are located in the sentence. Then, the POS parts are assigned to one of the following classes: Verb, Human, Entity, Location, Time, Count. The engine uses 90 predefined sentence schemas, like "Human Verb Entity" or "Human Verb Human Time". The sentence schema infer also a question type like "Whom/Who" or "Who/Where". According to the test experiments on TREC-2007 (Question Answering Track) [15] dataset, the engine could achieve a 0.12 - 0.55 recall ratio depending on the question type.

Another direction is to select also the target sentence from a larger text. A good example is the Python project wikipedia-question-generator [16]. The project uses beside Wikipedia also the WordNet ontology database to determine the synonyms of the target concept. Having a topic, the engine will generate a multi-choice question. For example, for the keyword 'Tony Bennett', we get the following question:

{ "question": "Bennett is also an accomplished _____, having created works under the name Anthony Benedetto that are on permanent public display in several institutions.",

"answer": "painter"

"similar_words": ["classic", "classicist", "constructivist", "decorator", "draftsman", "etcher", "expressionist", "illustrator"] }

The target sentence is selected on a relatively simple algorithm:

- Only the summary section is considered for sentence selection; at other parts, the sentences are partially and strongly related.
- Omit the first sentence of the summary, it is too straightforward to make interesting trivia.
- Avoid sentences starting with an adverb, as these are strongly depending on the previous sentences.
- Select the first common noun in the sentence as target concept.

According to the test experiments, these simple approaches can provide a relatively good result.

4. Case Demonstration

For our test case, we have selected a standard textbook [17] entitled "A primer on SQL". The textbook contains 64 pages. We have selected the keyword 'Insert" for the target concept. The selected topic is how to insert new records into a table. This word has about 112 occurrences in the textbook. Important issue, that from a large number of matching sentences, most have only a little relevance, like the example sentence in Fig 2.

This means that only few sentences can be used for generating of relevant questions. The evaluation of an adequate relevance level is a key challenge to be solved in the future developments.

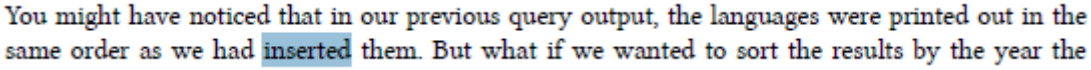
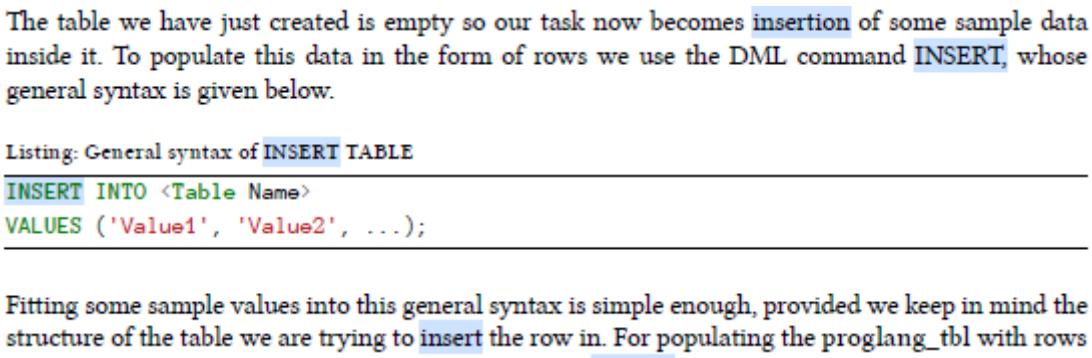


Fig 2, Sample sentence to select the keyword 'insert' for the target concept

Another key difficulty relates to the fact that the definitions are usually spread across several sentences.



The table we have just created is empty so our task now becomes insertion of some sample data inside it. To populate this data in the form of rows we use the DML command INSERT, whose general syntax is given below.

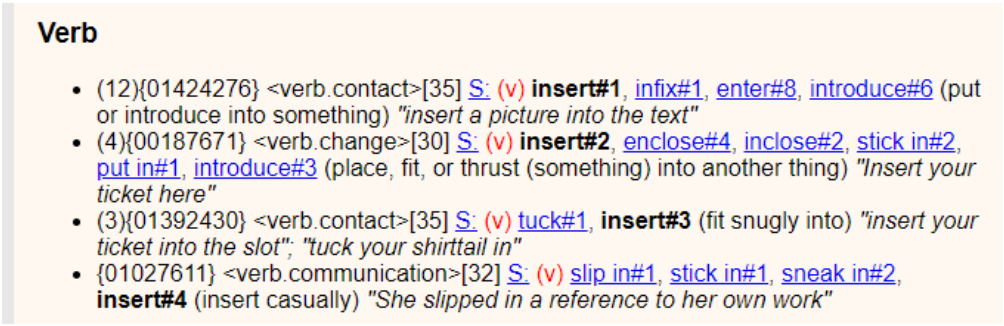
Listing: General syntax of INSERT TABLE

```
1  INSERT INTO <Table Name>
2  VALUES ('Value1', 'Value2', ...);
```

Fitting some sample values into this general syntax is simple enough, provided we keep in mind the structure of the table we are trying to insert the row in. For populating the proglang_tbl with rows

Fig 3, Section on the Insert Table Command

To connect the related adjacent sentences into a single semantic structure, the engine should incorporate an ontology level semantic engine. One approach to solve this challenge could be the generation of a dependency graph for each sentence clauses and to merge these graphs into a single semantic graph. The merge process should be supported by external ontology engines. As the example shows, the merge engine should consider also additional metadata information like heading of the paragraphs and sections; format parameters.

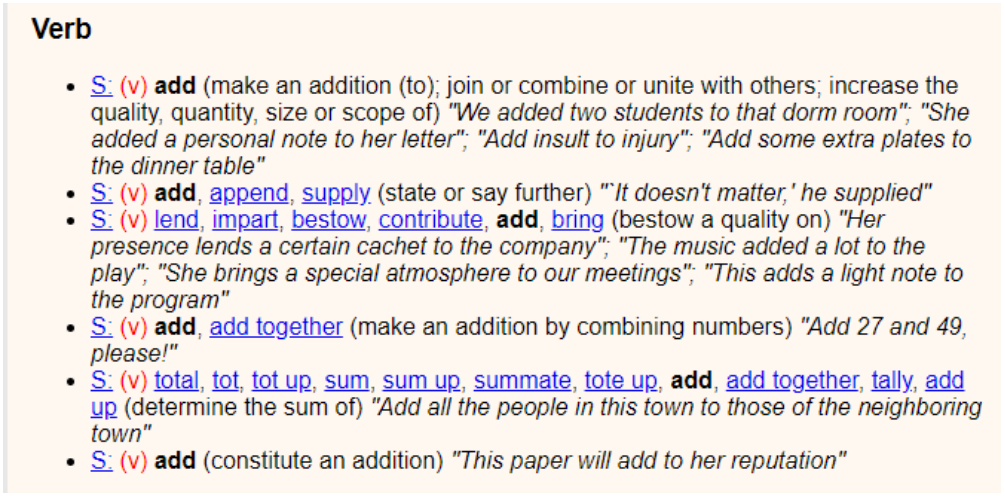


Verb

- (12){01424276} <verb.contact>[35] S: (v) **insert#1**, infix#1, enter#8, introduce#6 (put or introduce into something) *"insert a picture into the text"*
- (4){00187671} <verb.change>[30] S: (v) **insert#2**, enclose#4, inclose#2, stick in#2, put in#1, introduce#3 (place, fit, or thrust (something) into another thing) *"Insert your ticket here"*
- (3){01392430} <verb.contact>[35] S: (v) tuck#1, **insert#3** (fit snugly into) *"insert your ticket into the slot"; "tuck your shirttail in"*
- {01027611} <verb.communication>[32] S: (v) slip in#1, stick in#1, sneak in#2, **insert#4** (insert casually) *"She slipped in a reference to her own work"*

Fig 4a, WordNet Fragment on Keyword 'Insert'

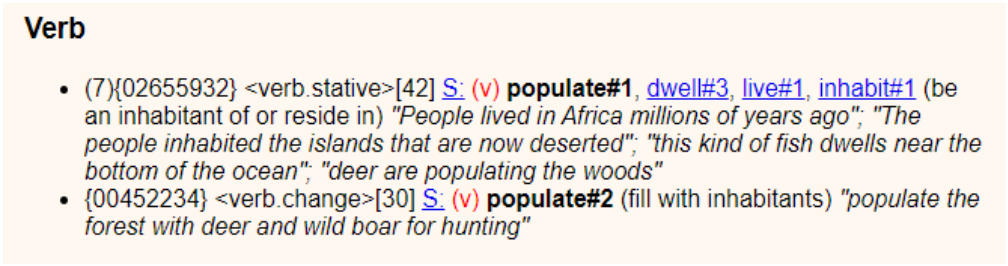
The third main challenge relates to the incomplete background ontology and knowledge databases. Currently, the prototype systems use the WordNet to get the semantic relationships (like synonyms) among the different words. On the other hand, these ontology sources are not complete yet and we cannot expect to extend these knowledge sources in the near future. In this domain, we use the verbs "insert","add" or "populate" as synonyms. For example, the query engine should use the term "add record" instead of the standard keyword 'insert". If we try to get from the word "add" to "insert" using WordNet, we cannot find the semantic link between them. As the next Figures show, the WordNet does not provide a direct link between these domain-level synonyms.



Verb

- S: (v) **add** (make an addition (to); join or combine or unite with others; increase the quality, quantity, size or scope of) *"We added two students to that dorm room"; "She added a personal note to her letter"; "Add insult to injury"; "Add some extra plates to the dinner table"*
- S: (v) **add**, append, supply (state or say further) *"`It doesn't matter,' he supplied"*
- S: (v) lend, impart, bestow, contribute, **add**, bring (bestow a quality on) *"Her presence lends a certain cachet to the company"; "The music added a lot to the play"; "She brings a special atmosphere to our meetings"; "This adds a light note to the program"*
- S: (v) **add**, add together (make an addition by combining numbers) *"Add 27 and 49, please!"*
- S: (v) total, tot, tot up, sum, sum up, summate, tote up, **add**, add together, tally, add up (determine the sum of) *"Add all the people in this town to those of the neighboring town"*
- S: (v) **add** (constitute an addition) *"This paper will add to her reputation"*

Fig 4b, WordNet Fragment on Keyword 'Add'



Verb

- (7){02655932} <verb.stative>[42] S: (v) **populate#1**, dwell#3, live#1, inhabit#1 (be an inhabitant of or reside in) *"People lived in Africa millions of years ago"; "The people inhabited the islands that are now deserted"; "this kind of fish dwells near the bottom of the ocean"; "deer are populating the woods"*
- {00452234} <verb.change>[30] S: (v) **populate#2** (fill with inhabitants) *"populate the forest with deer and wild boar for hunting"*

Fig 4c, WordNet Fragment on Keyword 'Populate'

This experience shows that the semantic parsing engine should be based mainly on local, domain specific ontology databases, especially when using non English text sources.

5. Experiments with Template Based Question Generation

Template-based method is an automatic question generation baseline which utilizes templates extracted from training set and then generates questions by filling the particular templates with certain topic entities. Template-based method can generate understandable questions for most triples and achieve competitive performance in automatic and human evaluation. We have prepared Automatic Question Generation template using Python spaCy libraries. To do this first we have first prepared the basic steps POS tagging, Dependency, Named Entity Recognition.

In the QG program, the rules are given in the form of a sequence containing

- fixed keywords (like Where, Who,..)
- POS templates (like nsubj, agent)

The POS templates tags refer to words, word sequences from the input sequence. The rules can be derfined in a straightforward way as it is shown in Fig 5.

```
#.....................................................................................
def MADV1():
    Q="What "+"does "+dePnsubj+" "+"do"+" when "+ dePadvel+" "+dePagent+ " "+dePdet+" "+dePpobj+"?"
    A=dePnsubj+" "+dePROOT+" "+dePdet+" "+dePdobj+" "+dePprep+" "+dePdet+" "+dePpobj
    print("Q1="+Q)
    print("A1="+A)
def MADV2():
    Q="What "+"does "+dePnsubj+" "+"do"+" when "+ dePadvcl+" "+dePadvmod+"?"
    A=dePnsubj+" "+dePROOT+" "+" "+dePprep+" "+dePpobj
    print("Q2="+Q)
    print("A2="+A)
def MADV3():
```

Fig 5, Template defintion commands in spaCy

The performed first experiments showed that the simple template-based question generation based only on syntactic rules, could not provide an accepatance level without using a domain dictionary. In the next step of our reserch, we integrate an ontology engine to provide the required domian knowledge dictionary.

6. Conclusion

The automated question generation module is a key component in ITS systems. A key problem with AQG systems is the selection of the appropriate target sentences and concepts. According to our experiments, we face the following challenges: the evaluation of the adequate relevance level for the documents fragments; the information units like definitions are usually spread across several sentences and we should find a way to connect the related adjacent sentences into a single semantic structure. The third main challenge relates to the incomplete background ontology and knowledge databases. The solution to these problems should have a high priority in the development of up-to-date intelligent tutoring systems.

Acknowledgements

The research reported here was carried out as part of the EFOP-3.6.1-16-2016-00011 “Younger and Renewing University – Innovative Knowledge City – Institutional development of the University of Miskolc aiming at intelligent specialization” project implemented in the framework of the Széchenyi 2020 program. The realization of this project is supported by the European Union, co-financed by the European Social Fund.

References

[1] Freedman, Reva; Ali , S. S.; Mcroy, S. What is an intelligent tutoring system. Intelligence 2000, 11(3):15-16. **https://doi.org/10.1145/350752.350756**

[2] Fletcher, J. D., Sottilare, R. A. Intelligent Tutoring of Teams. Design Recommendations for Intelligent Tutoring Systems: Volume 1-Learner Modeling 2013, 1:237

[3] Nkambou, Roger; Mizoguchi, Riichiro; Bourdeau, Jacqueline (ed.), Advances in intelligent tutoring systems, 2010, Springer Science & Business Media **https://doi.org/10.1007/978-3-642-14363-2**

[4] Sewunetie W. T., Ali Ahmed G. H., Kovács L., The Development and Analysis of Extended Architecture Model for Intelligent Tutoring Systems, GRADUS 2019, 6(4):128-138.
[5] Jaspreet Kaur A. K. B. Automatic Question Generation from Hindi Text using Hybrid Approach. International Journal of Advanced Technology in Engineering and Science 2015, 3(1):435-442
[6] Smith M. H. Good question! statistical ranking for question generation, Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Los Angeles, California, 2010, pp. 609–617.
[7] Piwek, B. K. E. Varieties of Question Generation: Introduction to this special issue, Dialogue & Discourse 2012, 3(2):1-9. **https://doi.org/10.5087/dad.2012.201**
[8] Nguyen-Thinh Le T. K. N. P. Automatic Question Generation for Educational Applications – The State of Art, Springer-Verlag Berlin , Heidelberg, 2011.
[9] Rus, V., Wyse, B., Piwek, P., Lintean, M., Stoyanchev, S., & Moldovan, C. The first question generation shared task evaluation challenge (2010).
[10] Dzmitry Bahdanau K. C. Y. B. Neural machine translation by jointly learning to align and translate., International Conference on Learning Representations Workshop (ICLR) 2015.
[11] Thang Luong C. D. M., Effective approaches to attention based neural machine translation, Proc. of the Conference on Empirical Methods in Natural Language Processing, 2015. **https://doi.org/10.18653/v1/D15-1166**
[12] Karen Mazidi Rodney Nielsen, Linguistic Considerations in Automatic Question Generation, 2nd Annual Meeting of the Association for Computational Linguistics, (ACL 2014), pp 321-326. **https://doi.org/10.3115/v1/P14-2053**
[13] Ali, H., Chali, Y., Hasan, S. A. Automation of question generation from sentences, Proceedings of QG2010: The Third Workshop on Question Generation, 2010, pp. 58-67.
[14] McClosky, D., Charniak, E., & Johnson, M. Reranking and self-training for parser adaptation, In Proc. of the 21st International Conference on Computational Linguistics, 2006, pp. 337-344. **https://doi.org/10.3115/1220175.1220218**
[15] Bailey, P., De Vries, A. P., Craswell, N., & Soboroff, I. Overview of the TREC 2007 Enterprise Track. In TREC, 2007.
[16] Atbaker, Wikipedia Question Generator, https://github.com/atbaker/wikipedia-question-generator.
[17] Batra R., A Primer on SQL, Leanpub Publisher, 2015.