Computational Intelligence Framework for Automatic Quiz Question Generation

Conference Paper · July 2018

DOI: 10.1109/FUZZ-IEEE.2018.8491624

Igor Khokhlov
Rochester Institute of Technology
26 PUBLICATIONS **148** CITATIONS
SEE PROFILE

L. Reznik
Rochester Institute of Technology
142 PUBLICATIONS **1,124** CITATIONS
SEE PROFILE

Computational Intelligence Framework for Automatic Quiz Question Generation

Akhil Killawala, Igor Khokhlov, Leon Reznik
B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, USA
ask3341@rit.edu, ixk8996@rit.edu, lr@cs.rit.edu

***Abstract*—Computational intelligence techniques are attracting more and more attention in NLP and text analysis applications. This paper is devoted to their use in automatic question generation based on text analysis with the goal to develop the computational intelligence framework that should automate or semi-automate the process of quiz and exam question generation. The framework operation is based on information retrieval and NLP algorithms. It incorporates the application of production rules, LSTM neural network models, and other intelligent techniques. It allows generating multiple choice questions, true and false questions as well as "Wh"-type (What? When? How?) questions. Automation procedures for each type question generation are developed, presented and analyzed. The typical challenges in framework development and application are considered and possible solutions are discussed. The results of the framework application and its use for quiz generation in a real college class are presented and analyzed.**

***Keywords*-Automatic question generation; Natural language processing; machine learning**

I. Introduction

Most automatic question generation (AQG) applications can be broadly classified into two major groups based on their goals: 1) to support the development of dialogue and interactive question and answer systems, and 2) to automate educational assessment. Questions are generated either directly from the expository texts [1], [2] or after the domain topic identification [3]. The first AQG approach, proposed by Weizenbaums Eliza program in 1966 [4], was based on pattern matching to certain words in the patient's conversation without any understanding of the content. For example, if the patient said, "I am depressed these days" the computer would see the words "I am" and generate "How long have you been" followed by the remainder of the patient's statement so as to produce the question, "How long have you been depressed these days?". The second AQG domain in educational assessment has been investigated for many years [5], [6] too. In fact, creating an exam paper is a very time-consuming task. AQG can significantly reduce the workload of the instructors.

The overall AQG process can be described as follows [5], [6]: (1) perform a morph syntactic, semantic and/or discourse analysis of the source sentence, (2) identify topically important keywords from the sentences for question formulation, (3) replace the topically important keyword with a blank or "Wh"-question, (4) post-process the question and ensure the grammatical correctness. Interesting examples of these steps have also been discussed in Yao and Zhang [7], which made use of Minimal Recursive Semantics. Olney et al. [8] used concept maps to generate questions from text. These approaches focus more on semantics and grammar of the question created. Curto et al. [9] developed a novel approach of using lexical syntactic patterns to form question-answer pairs. Lindberg et al. [10] use semantic labels for identifying the patterns in text in order to formulate the questions.

There already exists a large body of AQG tools developed and employed for educational purposes dating back to the Autoquest system [5], which makes use of syntactic approach to generate "Wh"-type (What? Who? When? Where? How?) questions from individual sentences. In addition to Autoquest, there are other systems for "Wh"-question generation using approaches like transformation rules [11] and generating questions based on a given templates [12], [9]. There is also work done in gap filling questions, which is mainly used for vocabulary learning, vocabulary-testing, and language learning [13], [14]. In this paper, for "Wh"-questions generation, pattern matching is used to a certain extent. This paper pays a special attention to improving the quality of generated questions through supervised learning that is achieved by involving an instructor into ranking the questions. This approach facilitates generating quality questions by removing ones which are either not directly related to the text topic or have no specific meaning.

To implement AQG we use various artificial intelligence and machine learning techniques such as rule systems and neural networks. Among neural networks, recurrent neural networks (RNN) are very popular for the natural language processing (NLP) as they allow taking into consideration the text context. For the 'Wh"-question generation, we use Long Short Term Memory (LSTM) networks, which is a special modification of RNN.

The effectiveness of LSTM networks has been proven in NLP [15], language understanding [16], and machine translation [17] applications. All RNNs implement a chain of repeating modules, which allows remembering previous text. However, in standard RNNs, these modules have a very simple structure, and they are not able to remember long-term facts. LSTM networks also have these modules, but they have more complex structure. An LSTM's chaining module, instead of

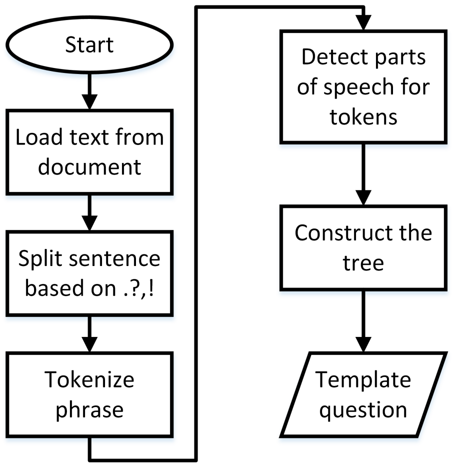
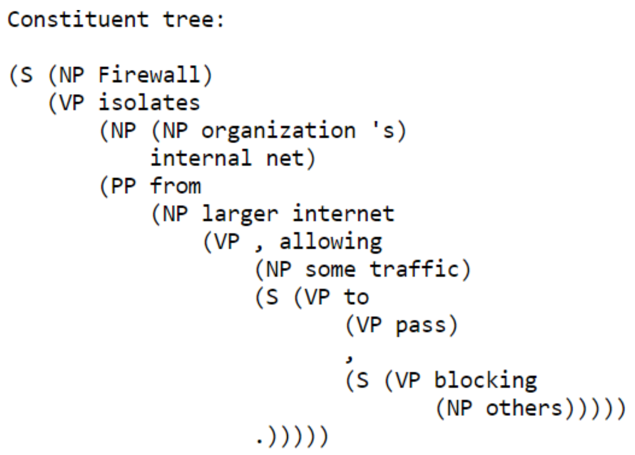


Fig. 1. Training process of question generation



```
Constituent tree:

(S (NP Firewall)
   (VP isolates
       (NP (NP organization 's)
           internal net)
       (PP from
           (NP larger internet
               (VP , allowing
                   (NP some traffic)
                   (S (VP to
                          (VP pass)
                          ,
                          (S (VP blocking
                                 (NP others)))))
               .)))))
```

Fig. 2. Constituent Tree object for the sentence

having only one neural network layer, has four of them, which interact in a special way.

Wang et al. [18] employ bidirectional LSTM RNN for the various automated tagging tasks. Ghosh et al. [19] introduced contextual LSTM network for large scale NLP tasks. This type of network demonstrates good performance in such specific NLP tasks as word prediction, next sentence selection, and sentence topic prediction. In our research, employing LSTM networks allows generating quiz questions that are relevant to the overall text topic.

II. Proposed framework implementation features

The main goal of the framework is to generate questions as well as rank the questions based on semantic correctness and difficulty level. Our framework is designed to generate four main types of quiz questions: true/false questions, multiple choice questions, fill in the blank questions, and "Wh"-questions. It implements a rule-based approach. It has been trained on a data with application of OpenNLP [20] tool. In its implementation, the framework passes two phases: the training phase (see figure 1) and the question generation phase.

The training phase could be divided into the following steps:

1) Load the text;
2) Detect topically important sentences;
3) Mark places,locations, time, and dates which can be potential gap candidates;
4) Detect and recognize different part of speech;
5) Construct the tree;
6) Eliminate unwanted phrases and punctuation marks;
7) Store the template questions and preview them.

During the training phase, the user can also rank the questions that will help direct the model to generate more meaningful questions. At this phase new words can be added to the dictionary as well in order to facilitate creating distractors discussed later in this paper. The user will have a question bank generated by the system, where the ranking can be done as per user knowledge usually the domain experts would do that.

TABLE I
An example of expression used

Purpose	Expression
1. To identify the main verb in the sentence	ROOT,<(S=clause <(VP=mainvp [<(/VB.?/=tensed !<,is | was | were | am | are | has | have | had | do | does | did) — </VB.?/=tensed !<VP,]))
2. To identify the main clause for the subject	ROOT = root <(S=clause < / (MD—VB.?) / = aux <(VP </VB.?/=verb)))

In AQG process, the text is analyzed sentence by sentence. During the analysis each sentence part is classified into various noun, pronoun, adverb categories, which are analyzed in order to select the base of the question. That is implemented as tree object constructed. For instance, if the sentence is: "Firewall isolates organization's internal net from larger Internet, allowing some traffic to pass, blocking others.", then the tree object is generated as shown in the figure 2. The Table I shows an example of various expressions which are used in order to identify the main verb or the main clause in the sentence.The tree object is created using the Standford parser library available in python.

The main verb in the sentence is identified with regular expressions (see Table I for examples) based on present verbs, their tenses and sentence clause.

In deciding whether a question should be classified as a "Wh"-question and which type ("Who", "What", "Why", etc.) two techniques are employed: Named-entity recognition (NER) parser (which is a part of the OpenNLP tool) and Super-sense tagging (SST). NER is a subtask of information extraction that tries to locate and classify named entities in text into pre-classified categories such as the names of people, organizations, locations, time and quantities. SST is a NLP task where each entity such as noun, verb, and others are annotated within the taxonomy defined by the WordNet [21]. NER parser uses words that are related to a person, location, or time as a key feature that helps to categorize "Wh"-question. Based on the feature rules that are shown in the Table II,

TABLE II
"WH"-QUESTION RULES

Common Tag	NER Tag	SST Tag	"Wh"- question
Person	Person	Noun.person	**Who**
Organization	Organization	Noun.group	**What**
Time or date	∅	Noun.time	**When**
Location	Location	Noun.location	**Where**
Other	Other	All except above	**What**

TABLE III
LSTM TRAINING SNAPSHOT

Called_VP	What did noun phrase (NP) do?
boy_ADJP	Who is the NP?
Jack_NP_PERSON	Who is NP?
said_VP	What did NP said to N1?
run_VP	Where did NP run to?

various kinds of "Wh"-questions are generated from the given sentences.

LSTM neural network method was chosen to train the model in order to predict which type of question it can generate. Table III demonstrates a few examples which "Wh"-questions are generated based on sentences' parts classification. The framework is trained on previous quiz questions, Wikipedia documents, and manually constructed questions. The training set consisted of around ≈ 80 questions on which the model was trained initially. Afterwards, the domain experts can rate the generated questions and this ranking is used to improve the model. All questions which had a rating between 4-7 were manually modified and those questions were added to the training set.

III. QUESTION GENERATION ALGORITHMS

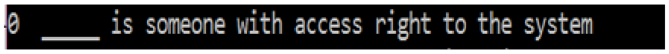
Various questions such as fill in the blank have been tested. Further discussion of each question type generation is presented in subsections below.

A. Fill in the blank question

Fill in the blank questions is one of the most common type of questions that a student encounters during the quiz or exams. For fill in the blank type of questions, the key point was to identify the correct gap for which the blank needs to be created.

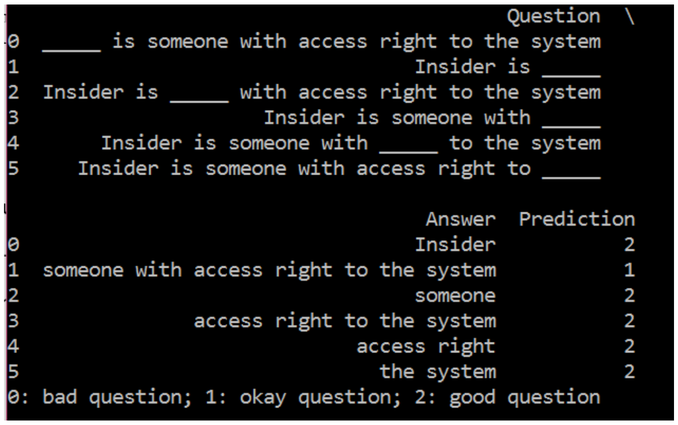
The procedure developed for identifying the gap is as follows:

1) Train the model with predefined questions and gap:
   a) Train the model on set of questions, gaps and ranks associated with it.
   b) Take the text as input and process each sentence at a time.
   c) Identify important keywords in the sentences and try to create gaps for the same.
   d) Display all possible sentences.
2) Use NLP to identify the verbs, pronoun, noun. Gap Selection from the sentence is another important aspect. To identify the gap, we employ Stanford parser to extract noun phrase (NP) and adjective phrase (ADJP) from important sentences as candidate gaps. Step wise approach is as follows:
   a) Extract nouns, np, adjp from tree object.
   b) Using the topically important sentence extract the key gaps.
   c) Use lambda function and training data set to identify potential gaps.
3) Eliminate the stop words from being used as gaps;
4) Give priority to Numbers/Places for gap creation.



```
0 _____ is someone with access right to the system
```

Fig. 3. The fill in the blank question generation



```
                                          Question  \
0  _____ is someone with access right to the system
1                                Insider is _____
2  Insider is _____ with access right to the system
3                      Insider is someone with _____
4      Insider is someone with _____ to the system
5     Insider is someone with access right to _____

                                     Answer  Prediction
0                                   Insider           2
1  someone with access right to the system           1
2                                   someone           2
3                 access right to the system           2
4                              access right           2
5                                the system           2
0: bad question; 1: okay question; 2: good question
```

Fig. 4. Showing all possible Fill in the blank question

Using the above approach, we were able to create the gap filling questions as seen in Figure 3. The original text from which the question in Figure 3 was generated is "Insider is someone with access right to the system". Despite good results with a first try, our framework creates all possible types of questions from a given sentence and then ranks them given good, bad, or OK evaluation. For ranking questions other features were used, for example, how many words the blank has, how many stop words it contains, frequency of occurring words and punctuation marks.

Figure 4 demonstrates how the system generates all possible type of questions to sentence "Insider is someone with access right to the system" and ranks them automatically.

Various issues need to be addressed during the ranking process. As seen in Figure 4, if we analyze second question generated, then it seems that a lot of words are in the blank and ranking should be ideally 0 but it as ranked as 1. So the ranking model did not always work as per the expectations.

Another major issue is the ambiguity in the answers. For instance, the question "_______ is someone with access rights to the system." may have various correct answers, which are not restricted to "Insider" only.

In order to solve this problem, we follow up the approach presented by Heilman et al. and Agarwal et al. [6], [13], who generate fill-in-the-blank questions and distractor answers from the text with the help of heuristic scoring measures. We

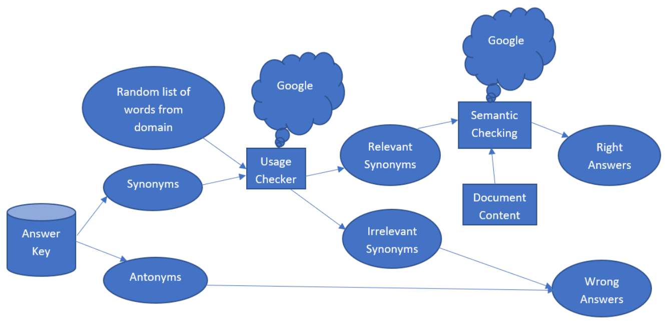


Fig. 5. Approach for generating distractors

also restrict the users answer by giving them multiple choices to select from.

The approach of distractors generation is shown in Figure 5. In Figure 5 the selected keyword which is identified as the answer key is taken into consideration. With the help of NLTK dictionary and WordNet, all possible antonyms and synonyms for the given word are produced. The antonyms generated are directly classified as the potential distractors. For the synonyms which are generated the same approach cannot be applied. The synonyms generated are passed through the checker module which internally calls the usage checker using online resources to see similar sentences, which is employed in classifying the synonyms into relevant and irrelevant. If the synonyms are irrelevant then they are considered as potential distractors. The relevant synonyms are passed through semantic checking.

For the text of technical nature, generating synonyms and antonyms could be particularly hard due to special meaning of certain words. For instance, if the candidate gap is "Firewall" then generating antonyms and synonyms is problematic as this word might have multiple special meanings. To overcome this problem, we developed the list of random words related to the particular technical domain. This random list of words is created by picking up technical jargons which are present in the entire document as well as those words which are potential blank for other sentences. The words picked from random list is also a part of distractors and incorrect answers.

B. True/False question generation

True and False type of questions present another classic example of questions commonly incorporated into exams or quizzes. The generation of True/False questions is mainly based on facts and figures. There are multiple ways to generate a True/False question. One way is to move the modal verb to the front of the sentence and add a question mark at the end. This can be achieved via transducer template solutions.

The following procedure was developed:

1) Make every statement in the list of statements False, except one.
2) If terms in the sentences have negative words like "could not / does not" convert to "could / does" and vice versa. If the sentence has no negative terms then try and add "not" in the sentence.
3) If there are numbers present in the question, try and manipulate the digits.

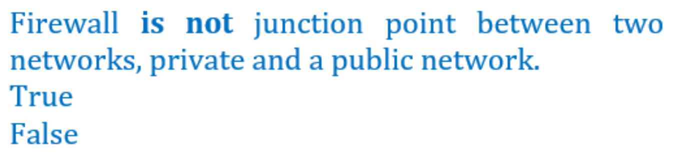

Firewall **is not** junction point between two networks, private and a public network.
True
False

Fig. 6. Classic True and False question

The procedure generates three categories of True/False questions:

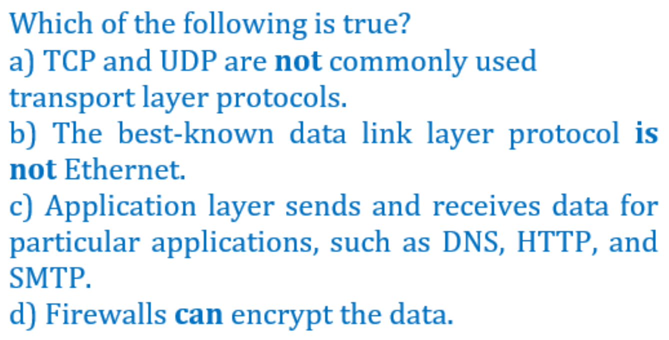
1) Classic True and False: In this scenario a statement is given and the user needs to identify whether the statement is true or false. In order to make it tricky, we use negations like can converted to cannot, as well as if numbers were present then we manipulate with it in the range. An example of this question generation can be seen in the Figure 6.

Figure 6 demonstrates that the original sentence did not have the term "not" present, but the system explicitly added the term in order to falsify the statement. The procedure employs multiple pairs of terms such as: is/not, can/cannot, does/does not, and so on, which would convert negation to a positive statement and vice versa.

As was mentioned above, number varying also can create hard to answer questions. For example, if the original sentence has "IPv4 is the fundamental network layer protocol" then the procedure will replace this number with a random one in the $\pm$ 50% range by changing it to, for instance, IPv5. In this case, it might be hypothetical but for questions pertaining in History or Biology domain these number varying techniques can be helpful to gauge students understanding.

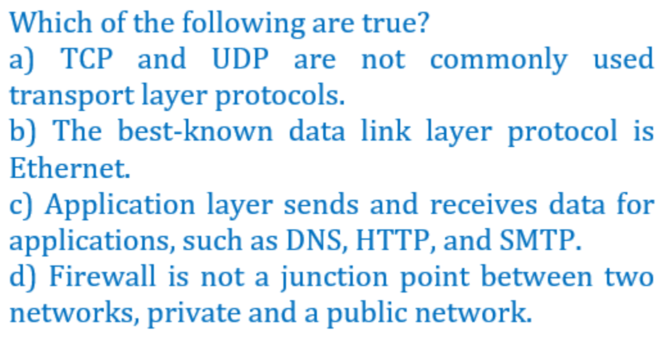
2) Multiple choice single answer True/False: In this scenario, four topically important sentences are taken into consideration, out of which three of them are negated or manipulated with number is present, and only one of these sentences is left in original form which is considered as the right answer. To select the topically important sentances, we took into account the tree parsed and created in the Figure 2 as well as the frequency of words occuring in the entire paasage and technical jargons present if any using the technical jargon reference manual for that domain. An example of this scenario is shown in Figure 7.

3) Multiple choice multi answer True/False: This approach is an extension of multiple choice single answer method. This algorithm is called up to hide three negations pattern in the previous approach. In this scenario, the numbers of true and false answers are randomly chosen each time the question is generated. The user needs to identify and mark all the true statements in order to get full credit. Figure 8 presents how this question would look like. Inclusion of this question type increases the level of a quiz difficulty.



Which of the following is true?
a) TCP and UDP are **not** commonly used transport layer protocols.
b) The best-known data link layer protocol **is not** Ethernet.
c) Application layer sends and receives data for particular applications, such as DNS, HTTP, and SMTP.
d) Firewalls **can** encrypt the data.

Fig. 7. An example of multi choice single answer True/False. Option (a), option (b) and option (d) have been negated from their original forms while option (c) is intact which is the correct answer.



Which of the following are true?
a) TCP and UDP are not commonly used transport layer protocols.
b) The best-known data link layer protocol is Ethernet.
c) Application layer sends and receives data for applications, such as DNS, HTTP, and SMTP.
d) Firewall is not a junction point between two networks, private and a public network.

Fig. 8. Multiple choice multi answer True/False. Option(a) and option (d) have been negated whereas option (b) and option (c) have been kept intact. The user has to identify all the correct options and mark them to get full credits.

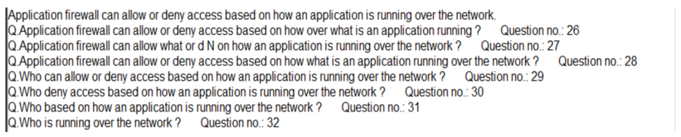
C. *"Wh"-question Generation*

In this module, the system generates "Wh"- type of questions i.e. the questions which start with "Who", "What", "Why", and "How". The following algorithm was developed to generate this type of questions:

1) Select the topically important sentences;
2) Identify the gaps using the algorithm described in subsection III-A based on various patterns associated Who, Where, What and How clauses;
3) Terminate the sentence with "?".

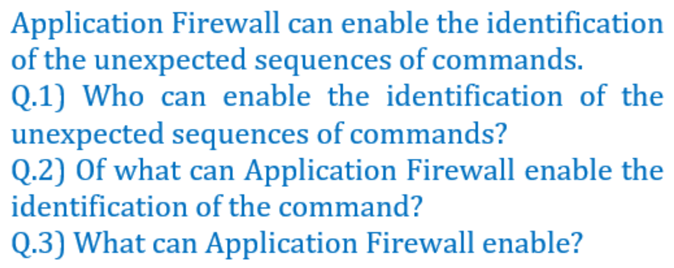
In order to detect whether a sentence should be associated with Who vs. How a list of patterns was developed by generalizing from the sentence features, which were developed at the questions generating from text phase. The question is formulated based on matching pattern, which is associated with the given set of rules. For instance, if the target keyword selected is classified as Person entity then the clause to be used more aptly is "Who" rather than "When" or "How". The process of generating this "Wh" clause questions is difficult as domain knowledge is commonly vast, and a lot of training is needed. Figure 9 illustrates multiple questions that need to be processed. In order to help the model generate better questions, we devised a ranking module user interface which would ask the user to rank these generated questions on a scale of 0-10.

Ranked questions would be saved in the file and later will



```
Application firewall can allow or deny access based on how an application is running over the network.
Q.Application firewall can allow or deny access based on how over what is an application running ?     Question no.: 26
Q.Application firewall can allow what or d N on how an application is running over the network ?     Question no.: 27
Q.Application firewall can allow or deny access based on how what is an application running over the network ?     Question no.: 28
Q.Who can allow or deny access based on how an application is running over the network ?     Question no.: 29
Q.Who deny access based on how an application is running over the network ?     Question no.: 30
Q.Who based on how an application is running over the network ?     Question no.: 31
Q.Who is running over the network ?     Question no.: 32
```

Fig. 9. Possible "Wh"-questions generated. For a given sentence there are many questions which are generated from the "Wh" clauses as it matches with one of more pattern rules. Out of these questions there are some questions which make no sense at all.



Application Firewall can enable the identification of the unexpected sequences of commands.
Q.1) Who can enable the identification of the unexpected sequences of commands?
Q.2) Of what can Application Firewall enable the identification of the command?
Q.3) What can Application Firewall enable?

Fig. 10. Wh- Question scenario 1. All possible questions from the sentence above are generated. Q.3 seems relevant while Q.1 and Q.2 seem irrelevant.

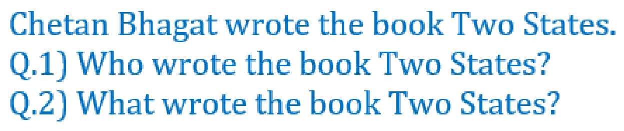
be used for training the model. For instance, all questions which would be ranked above 7 would be good to keep in a quiz, whereas questions having rank between 4-7 would need manual intervention, and questions having rank less than 4 would be irrelevant and can be ignored.

IV. Challenges

Automatic question generation creates many computational and linguistic challenges. We classify the challenges into three categories: lexical challenges, discourse related challenges, and syntactical challenges.

A. *Lexical challenges*

Lexical challenges that were encountered during this research were mainly related to questions being generated from short phrases and sentences. The semantics of the answer to a question plays an important role in determining the question that will be formed. For the fill in the blank and True/False scenario it was not a significant issue. For the "Wh"-question generation, it will determine the sub-type classification, i.e. whether the question will be classified as a who question, what question, etc. The process is illustrated in Figure 10. Another possible scenario of the misclassification is presented in Figure 11.



Chetan Bhagat wrote the book Two States.
Q.1) Who wrote the book Two States?
Q.2) What wrote the book Two States?

Fig. 11. Wh- Question scenario 2. Chetan Bhagat is an author who wrote the book called "Two states". The first question is acceptable but the second question does not seem to be relevant.

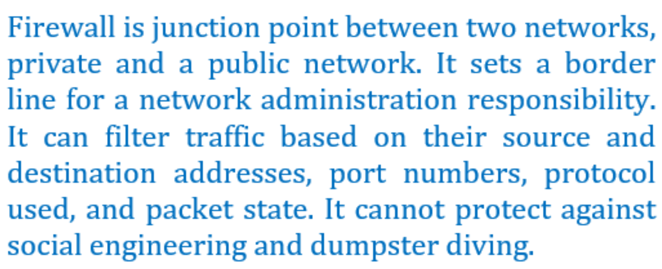

Firewall is junction point between two networks, private and a public network. It sets a border line for a network administration responsibility. It can filter traffic based on their source and destination addresses, port numbers, protocol used, and packet state. It cannot protect against social engineering and dumpster diving.

Fig. 12. Relevance of keyword "it" for question generation

B. Paraphrasing

Paraphrasing is another challenge in an automatic questions generation. It allows creating a slight variation of the original text. Paraphrasing techniques were developed by Callison-Burch [22] and Kok and Brockett [23]. For instance, in Figure 11 example, the system should also generate questions like "Who authored the book Two States?", "Who was the book Two States written by?", "Who wrote the book Two States?". All these variations are also acceptable and valid questions.

C. Generation of Distractors

The generation of distractors also becomes challenging if the topic is too technical. In this case, getting distractors with help of synonyms and antonyms is difficult as these special jargon words may not have common synonyms and antonyms. One of the way of this issue solving is to have a dedicated list created which extracts all such keywords from the text and stores them in a dictionary. All such words can be used as distractors for other questions for which these words are not a proper answer.

D. Relevance of pronouns

When the framework analyzes text, it needs to consider sentences connection. Unfortunately, our system fails to do so and an instructor needs to manually correct generated sentences. The example of this issue is presented in Figure 12. The first sentence has the main term "Firewall" and the following sentences are referring to "Firewall" by using the keyword "it". However, the framework fails to create the connection between sentences and creates questions considering sentences as independent entities.

This is a significant research problem, which search engines are currently trying to solve with varying success. For instance, if the first query is "Who is Bill Gates?" and the second query is "What is his net worth?", only some search engines can connect these two queries. If we run them one after another in Google Search then for the second query in we will get irrelevant result giving net worth of random "his" entity, where as in Microsoft Bing search or Google Assistant for the second query we will get net worth of Bill Gates.

We were trying to employ the Long short term memory (LSTM) graph storage to form connections between sentences. However, for a given piece of passage it is very difficult to form the tree object in order to create questions. Unfortunately,

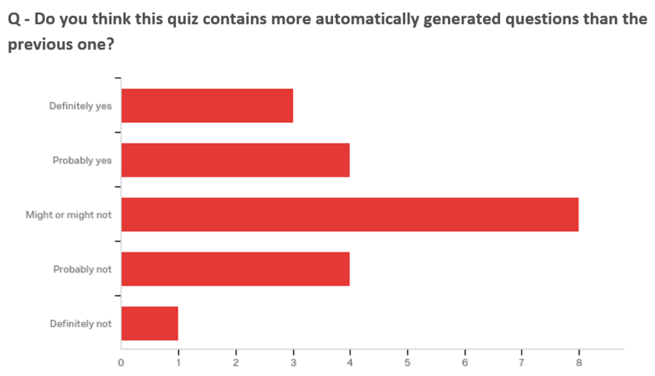

Q - Do you think this quiz contains more automatically generated questions than the previous one?

Fig. 13. Survey question 2 analysis

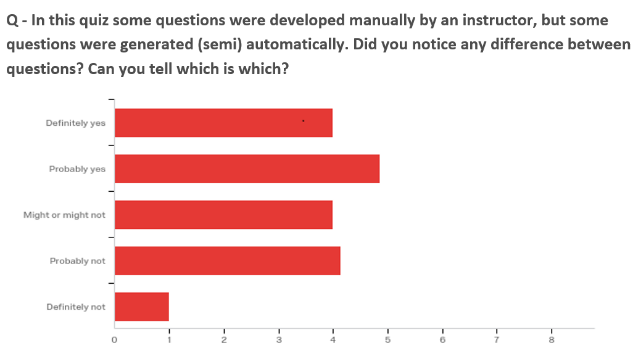

Q - In this quiz some questions were developed manually by an instructor, but some questions were generated (semi) automatically. Did you notice any difference between questions? Can you tell which is which?

Fig. 14. Survey question 3 analysis

our investigation on LSTM shows that there are various limitations for this approach.

V. Testing and Survey

In this project we were trying to verify the AQG performance by comparing it with automatic generated questions against an instructor generated questions. In this project we conducted the survey of students who were given two quizzes. One quiz included only instructor generated questions, and another contained both the instructor's and automatically generated questions.

The questions in the survey were mainly focused on understanding the user behavior and asking responders if they could notice any difference between the manually generated quiz and automatically generated questions as well as if they could identify which questions could be classified in which category.

Based on Figure 13 and Figure 14 it is easy to gauge that the majority of responders seem to have found a difference between automatic questions and manual questions, but they would have difficulty in being able to identify which question is generated automatically and which is manually constructed. The votes are more or less evenly distributed from "Definitely Yes" to "Probably not". It implies to a certain extent that the framework was able to perform well on evaluating the student's performance on the knowledge of the topic.

This survey helped us to understand that probably more fine tuning of the question generation needs to be done but at the same time the questions created by the system are good enough in comparing to manually generated questions.

VI. Conclusion

AQG questions have been employed in two major domains: automatic dialogue systems and for educational assessment. In the second area, the AQG helps unload instructors and allow them to concentrate on teaching and curriculum development. The automatic question generation framework that employs computational intelligence techniques has been developed, implemented and tested in real college class environment.

Unlike other implementations, the developed framework is able not only to generate questions but also to rank them based on the semantic correctness and difficulty level. An instructor is given an opportunity to rank the generated questions that later improves the model and facilitates more meaningful question generation. The question ranking introduces AQG system supervised learning which may significantly improve the generated question quality by eliminating irrelevant questions and ones with poor semantics.

Generated questions may belong to one of the four types: true/false question, multiple choice question, fill in the blank question, and "Wh"-type (What? Where? When? How?) questions. Computational intelligence techniques such as production rules and LSTM neural network models have been employed in sentence analysis, context preserving and sentences connecting, which are used for question generation.

During the framework development, a number of challenges have been addressed with an application of computational intelligence and NLP techniques. In generating "Wh"-type questions, short sentences might create difficulties in exact "Wh"-question type classification (i.e. "Who", "What", "When", or "How"). Paraphrasing based on NLP analysis is the technique that is employed to improve the quality of the generated questions. For technical texts, a distractor generation may become an issue due to a lack of synonyms or antonyms for technical jargon words in commonly used databases. This problem was solved by generating a dedicated list of words that can be applied as a distractor later on. Finally, despite trying to employ LSTM, the framework had a major difficulty in connecting sentences where a noun is substituted with pronouns.

An empirical study was conducted with the goal to verify the performance of the developed AQG framework. In order to compare the quality of the automatic generated questions against an instructor generated questions, the automatic generated questions were included in a real life quiz. Another quiz, which was considered as a control group comparison base, contained an instructor generated questions only. After the quiz, a student's survey was conducted that is focused on understanding students behavior and to verify if they could distinguish automatically generated questions from the manually created. Based on the respondent's responses, our study concluded that responders were more likely not able to differentiate automatically and manually generated questions that indicates a good quality of automatically generated questions.

Acknowledgment

This material is partially based upon work supported by the National Science Foundation under Award # ACl-1547301 and National Security Agency under Award # H98230-I7-l-0200.

References

[1] H. Prendinger, P. Piwek, and M. Ishizuka, "Automatic generation of multi-modal dialogue from text based on discourse structure analysis," in *Semantic Computing, 2007. ICSC 2007. International Conference on*. IEEE, 2007, pp. 27–36.

[2] P. Piwek and S. Stoyanchev, "Generating expository dialogue from monologue: motivation, corpus and preliminary rules," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 333–336.

[3] S. Harabagiu, A. Hickl, J. Lehmann, and D. Moldovan, "Experiments with interactive question-answering," in *Proceedings of the 43rd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2005, pp. 205–214.

[4] J. Weizenbaum, "ELIZAa computer program for the study of natural language communication between man and machine," *Communications of the ACM*, vol. 9, no. 1, pp. 36–45, 1966.

[5] J. H. Wolfe, "Automatic question generation from text-an aid to independent study," *ACM SIGCSE Bulletin*, vol. 8, no. 1, pp. 104–112, 1976.

[6] M. Heilman and N. A. Smith, "Question generation via overgenerating transformations and ranking," CARNEGIE-MELLON UNIV PITTSBURGH PA LANGUAGE TECHNOLOGIES INST, Tech. Rep., 2009.

[7] X. Yao and Y. Zhang, "Question generation with minimal recursion semantics," in *Proceedings of QG2010: The Third Workshop on Question Generation*, 2010, pp. 68–75.

[8] A. M. Olney, A. C. Graesser, and N. K. Person, "Question generation from concept maps," *Dialogue & Discourse*, vol. 3, no. 2, pp. 75–99, 2012.

[9] S. Curto, A. C. Mendes, and L. Coheur, "Question generation based on lexico-syntactic patterns learned from the web," *Dialogue & Discourse*, vol. 3, no. 2, pp. 147–175, 2012.

[10] D. Lindberg, F. Popowich, J. Nesbit, and P. Winne, "Generating natural language questions to support learning on-line," 2013.

[11] N. Karamanis, L. A. Ha, and R. Mitkov, "Generating multiple-choice test items from medical text: A pilot study," in *Proceedings of the Fourth International Natural Language Generation Conference*. Association for Computational Linguistics, 2006, pp. 111–113.

[12] W. Chen, "Aist, g., mostow, j.: Generating questions automatically from informational text," in *Proceedings of the 2nd Workshop on Question Generation (AIED 2009)*, 2009, pp. 17–24.

[13] M. Agarwal, R. Shah, and P. Mannem, "Automatic question generation using discourse cues," in *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, 2011, pp. 1–9.

[14] R. Agerri, J. Bermudez, and G. Rigau, "Ixa pipeline: Efficient and ready to use multilingual nlp tools." in *LREC*, vol. 2014, 2014, pp. 3823–3828.

[15] M. Sundermeyer, H. Ney, and R. Schlüter, "From feedforward to recurrent lstm neural networks for language modeling," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 23, no. 3, pp. 517–529, 2015.

[16] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu, "Recurrent neural networks for language understanding." in *Interspeech*, 2013, pp. 2524–2528.

[17] M. Sundermeyer, T. Alkhouli, J. Wuebker, and H. Ney, "Translation modeling with bidirectional recurrent neural networks." in *EMNLP*, 2014, pp. 14–25.

[18] P. Wang, Y. Qian, F. K. Soong, L. He, and H. Zhao, "A unified tagging solution: Bidirectional lstm recurrent neural network with word embedding," *arXiv preprint arXiv:1511.00215*, 2015.

[19] S. Ghosh, O. Vinyals, B. Strope, S. Roy, T. Dean, and L. Heck, "Contextual lstm (clstm) models for large scale nlp tasks," *arXiv preprint arXiv:1602.06291*, 2016.

[20] (2017) Apache OpenNLP. Accessed on 01.01.2018. [Online]. Available: https://opennlp.apache.org/

[21] (2010) Princeton University "About WordNet". Accessed on 01.01.2018. [Online]. Available: http://wordnet.princeton.edu

[22] C. Callison-Burch, C. Fordyce, P. Koehn, C. Monz, and J. Schroeder, "(meta-) evaluation of machine translation," in *Proceedings of the Second Workshop on Statistical Machine Translation*. Association for Computational Linguistics, 2007, pp. 136–158.

[23] S. Kok and C. Brockett, "Hitting the right paraphrases in good time," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 145–153.