Automatic generation of multiple choice questions for e-assessment

A. Santhanavijayan*, S.R. Balasundaram, S. Hari Narayanan, S. Vinod Kumar and V. Vignesh Prasad

Department of Computer Science and Engineering,
National Institute of Technology,
Tiruchirappalli 620 015,
Tamil Nadu, India
Email: vijayana@nitt.edu
Email: bala4@gmail.com
Email: harinarayan4@gmail.com
Email: vinodkumar2@gmail.com
Email: vigneshp@gmail.com
*Corresponding author

Abstract: It is important for students to expertise in their field of study, because there is an agile change in all the domains. Even though resources are available to learn, proper assessment helps them to improve upon their knowledge. In this paper, an automatic generation of multiple choice questions on any user-defined domain is proposed. It first extracts text relevant to the given domain from the web and summarises using fireflies-based preference learning. The sentences in the summary are transformed into stem for the MCQs. The distractors are generated using similarity metrics such as hypernyms and hyponyms. The system also generates analogy questions to test the verbal ability of the students.

Keywords: analogy; e-assessment; fireflies-based preference learning; hypernym; hyponym; MCQ; natural language processing; ontology; parts-of-speech; summariser.


Biographical notes: Santhanavijayan is an Assistant Professor of CSE in the National Institute of Technology, Tiruchirappalli. He received his M.E. degree in CSE from Anna University, Chennai in 2007. His area of interests include machine learning, image processing, data mining and data warehousing, and operating systems.

Balasundaram is an Associate Professor in the Department of Computer Applications in the National Institute of Technology, Tiruchirappalli. He received his M.C.A. degree from PSG College of Technology, Coimbatore and his M.E. degree in CSE in 1992. He received his PhD. degree in ‘E-Learning and Assessment’ from NIT, Trichy. He has more than 40 papers in reputed journals and proceedings of International Conferences. His current research areas of interest include web and mobile technologies, cognitive sciences, e-learning technologies.

Hari Narayanan received his BTech degree in Computer Science and Engineering from the National Institute of Technology, Tiruchirappalli in 2016. He attended summer internships in HCL Infosystems, Chennai and Tecknodreams software consulting Pvt. Ltd., Bangalore in 2014. He has also interned at Amazon Development Centre, Chennai in 2015. His current research areas of interests include web and mobile technologies, cognitive sciences, e-learning technologies.

Vinod Kumar received his BTech degree in Computer Science and Engineering from the National Institute of Technology, Tiruchirappalli, in 2016. Previously, he has interned at L&T Infotech, Chennai, India in 2014 and at Fidelity Investments, Bengaluru, India, in 2015. His current areas of interests include natural language processing, e-learning, e-assessment, automatic question generation, and data mining.
1 Introduction

In modern era, e-learning has become one of the major evolving environments, and it has seen as its own reward. There are a plenty of resources available for e-learning, but learning is not complete without assessment. Assessment is the process of gathering and discussing information from multiple and diverse sources to develop a deep understanding of what students know, understand, and can do with their knowledge as a result of their educational experiences; the process culminates when assessment results are used to improve subsequent learning. It is the systematic basis for making inferences about the learning and development of students. Although the notion of assessment is generally more complicated, assessment is often divided for the sake of convenience into objective and subjective. Objective assessment is a form of questioning which has a single correct answer, whereas the subjective assessment is a form of questioning which may have more than one way of expressing the correct answer. Objective assessment is well suited to the increasingly popular computerised or online assessment format.

So, in this paper, the emphasis is on generating a type of objective questions multiple choice questions. Multiple choice questions are a form of assessment in which respondents are asked to select the best possible answer out of the choices from a list. In many disciplines, instructors use MCQs as a preferred assessment tool, and it is estimated that 45–67% student assessments utilise MCQs (Ghosh, 2013). The fast developments of e-Learning technologies have in turn stimulated method for automatic generation of MCQs, and today, they have become an actively developing topic in application-oriented NLP research. MCQs can be used to assess a broad range of knowledge and yet require less administration time. In addition, they can be used to provide instant feedback to test takers. On the other hand, they are hard to construct and require considerable time for setting each question. In addition to the considerable preparation time, manual construction of MCQs does not necessarily imply that they are well constructed.

The structure of a multiple choice question is composed of a stem and a set of options (key + distractors). The major challenge in preparing MCQ is the need for good distractors, i.e., distractors should appear as a plausible answer to the question even to a student with good knowledge on the domain. At the same time, it should not be an alternate answer. Moreover, a well-written MCQ should contain enough information to answer the question. To reduce the effort and time required for generating MCQs, this paper presents a method to automatically generate them by using fireflies-based preference learning and ontology-based approach. Preference learning is used to identify the stem. Next, the distractors are chosen such that

1. it is relevant to the objective
2. it is not a synonym to the key.

Relevant to the objective means, it should be related to the context in the perspective of domain. If the distractor is a synonym to the key, then there will be more than one answer, which should not be the case. So, in this paper, two types of MCQs are generated, which are fill in the blank type and analogy type. The fill in the blank is a type of question or phrase with one or more words replaced within a blank, giving the reader the chance to add the missing word(s). Verbal analogy questions are of the form of comparison between two objects. ‘A car has wheel is like a book has page’ is an example of an analogy. Analogy questions will be of MCQ type having a stem and a set of options. Based on the process, the effective MCQ question assessment process is created.

Then the remaining section is organised as follows: Section 2 deals that the various authors’ discussions about the e-assessment-based MCQ analysis. Section 3 analyses the proposed system MCQ methodologies and section that generation of analogy questions is discussed. In Section 5, the efficiency of the results is evaluated. At last, it concludes in Section 6.

2 Related work

A numerous number of question generation techniques were proposed earlier (Kunichika et al., 2002; Mitkovand Ha, 2003; Heilman and Smith, 2011; Mostowand Chen, 2009). All the methods have stuck to either reading comprehension or assessing the vocabulary of the person using the tool. To extract the best sentences for generating questions, the metrics used are first sentence, count of common tokens, abbreviations and superlatives, sentence position, starting word of the sentence, length, and number of nouns and pronouns (Agarwal and Mannem, 2011). Keys are identified by having a general key list, and the best fit for a blank in the sentence is identified by the number of occurrences of key in the document, closeness with title and height of the key in the syntactic tree. The distractors are generated by
measuring the contextual similarity, dice coefficient score between the sentences, difference in term frequency of key, and distractors.

Alternatively, preference learning can be used to extract best sentences from corpus to form stem (Collins and Duffy, 2007). A model is created based on the frequency of words. Goto et al. (2010) and Heilman and Smith (2011) have calculated a score for each sentence depending on the score for each word in the sentence, and the sentence was ranked. The sentences with higher ranks are taken to form the stem. Blanks in each sentence were determined using CRF - a supervised learning algorithm. The sentences were POS and IOB2 tagged and passed to CRF algorithm (Sang and Veenstra, 1999). CRF gives the most appropriate word that can be made as blank based on the training set used. They have built the system with English literature as the domain, so the similarity among words is identified using - derivative word - shape of word and - meaning of word and is placed in the place of blank to find the number of hits in Google. The word with maximum hit is taken as key and others as distractors. Brown et al. (2005) used an approach that tests the knowledge of students by automatically generating test items for vocabulary assessment. Their system produced six different types of questions for vocabulary assessment by making use of a lexical database, WordNet. The six different types of questions include definition, synonym, antonym, hypernym, hyponym and cloze questions.

The suitable position for a blank can also be determined by the semantic relationship between the named entities which is identified by drawing the dependence tree for each sentence (Afzal and Mitkov, 2014). In a dependence tree, the main verb is taken as the root and its dependent noun phrases, and adjectives are considered to be its children. The best key is chosen by identifying the named entities that are present in the subtree with at least five levels. Also the named entity at a greater height in that subtree is more preferred to be a key than a named entity which is at a lesser height. This is because if the key is present at a greater height, then all its children will be useful in understanding the context of the sentence; thereby, it becomes easy for the student to identify the key correctly. Distractors are generated through distributional similarity measure. Distributional similarity is a useful measure and is used in many NLP applications such as language modelling, information retrieval, automatic thesaurus generation and word sense disambiguation.

Sathiyamurthy and Geetha (2012) proposed automatic question generation from documents for e-learning was focused on question generation based on Bloom’s taxonomy; here, the documents are collected over the web using Google search, and then those ranked documents are processed for noun/phrase extraction that serves as the keywords for question templates, which were designed for different level of Bloom’s taxonomy. Multiple choice analogy questions generation can be viewed as a two-phase process:

1. extraction of interesting pairs of concepts using the relatedness function, those pairs can be used as stems, keys, or distractors
2. generation of multiple-choice questions based on the similarity between pairs which can be derived from an analogy function (Alsubait et al., 2012).

The analogy-based questions were mined from ontologies.

### 3 Generation of fill in the blank type MCQs

In this section, the proposed automatic generation of the multiple choice questions in the e-assessment process is discussed. The method analyses the questions by summarising the text contents using the preference learning which is optimised with the help of the fireflies’ algorithm. Furthermore, the proposed system generates the fill in the blank type MCQs and generation of analogy questions using the five different components. Then, the structure of the proposed system is shown in Figure 1.

**Figure 1** Proposed system architecture (see online version for colours)

3.1 Finding web page with relevant information

The first step is to find web page that is relevant to the user-requested query. The user-defined query (topic/phrase on which questions are to be generated) is passed to a Google search API which returns a response containing URLs of relevant web pages. Using web as corpus provides the flexibility to generate questions in any domain.

3.2 Extraction of useful text from web pages

After finding the relevant information from the search engine, the useful text needs to be extracted from the web page. Web pages contain a mixture of multimedia (images/videos), texts, tables, and hyperlinks to other pages. However, the most useful information from which questions
can be generated is textual, which are enclosed within \(<p>\) tags. Hence, only text enclosed by \(<p>\) tags is extracted and the reserved characters in HTML are replaced with ASCII code before processing the text. It was also observed that the bottom 1/5th of most of the pages contain irrelevant information like examples or less important information. Thus, this part can be trimmed off from the text. Then, the structure of information extraction is shown in Figure 2.

**Figure 2** Extraction of text from web pages (see online version for colours)

After extracting the text from the web page, it has to be summarising for forming the effective multiple choice questions.

### 3.3 Summariser

In this paper, summarisation does not refer to the usual text summarisation. However, it denotes the collection of all sentences that can be transformed into a multiple choice question. Every potential MCQ must be

- Informative - Questions must be taken from important points.
- Complete - The sentences should be such that it contains enough information making it possible to answer the question after placing a blank in place of the key (answer).

The sentence is added to the summariser based on the following algorithm:

**Algorithm 1** Summarisation

Input: Text Document \(T_D\)

Output: Summarized Document \(S_D\)

```
Begin
For each stopwords \(S_W\) in \(T_D\)
Begin
Remove \(S_W\)
End
WordFrequency \(W_F(T_D)\)
For each word \(W\) in \(T_D\)
Begin
If WordFrequency(\(W\)) > (TotalNumberofWords/70) then
FrequentWords.add(\(W\))
End
S_D ={}
For each sentence \(S\) in \(T_D\)
Begin
For each word \(W\) in \(S\)
Begin
If (\(W\) in FrequentWords) Increment score
End
If(S does not contain(pronoun or adverb or ‘:’) and Score>Length(S)/10) then
S_D.add(\(S\))
End
For each \(S\) in \(S_D\)
Begin
Concatenate \(S\) to Summary
End
End
```

### 3.4 Choosing best key

A key for a sentence must be chosen in such a way that its absence does not make the context unclear. It should be possible to find the key using the remaining part of the sentence. The key must be a keyword and must not be easy for the students who do not have enough knowledge in the domain to predict the answer. For finding the best key, first, the stem is POS tagged (each sentence in the summary is a stem). Thus, the cardinals, nouns, adjectives, and their corresponding positions in the stem are identified. The order of preference for placing the blank is cardinals, proper nouns with describing adjective in start or end of the statement, common nouns with describing adjective in start or end of the statement, and adjective at any position in the stem. The preferences are mostly identified according to the fireflies’ algorithm that works characteristics of the attractiveness and lightness value of the extracted text information. The intensity value depends on the minimum and maximum ratings of texts present in the web page. If none of the above cases are identified, then the stem is not suitable to generate a question of good standard. It is simply discarded.

**Algorithm 2** Generation of Blanks

Input: A sentence S

Output: A sentence with a blank in the appropriate position, a key

1. \(Sents=[\text{list of all sentences in the summary}]\)
2. For each sentence in \(Sents\)
3. Begin
   a. Tokenize all words and POS tag the sentence.
   b. Scan the sentence to check whether it has any cardinal with the help of the fireflies.
   c. If cardinal is present
      d. Begin
         i. Make the cardinal as key
         ii. Continue
   e. End
   f. Check for a proper noun at the beginning of the sentence
   g. If a proper noun is present
      h. Begin
         i. Concatenate all nouns that follow the proper noun till the noun phrase ends
         ii. Concatenate the describing adjective(if any) to the noun phrase formed above
         iii. Make it as the key
         iv. Continue
      i. End
   j. Repeat step ‘h’ for finding proper nouns at the end of the sentence
   k. If a proper noun is not found
      l. Begin
         i. Repeat steps ‘h’ and ‘j’ to find any common noun at the beginning or end of the sentence
         ii. Make the phrase as the key
         iii. Continue
   m. End
   n. If neither proper noun nor common noun is present in the sentence
      o. Begin
         i. Search for adjective phrase in the sentence
         ii. Make it as the key
         iii. Continue
   p. End
   q. If the key is still empty
      i. Continue

4. End

3.5 Generation of distractors

This is the most crucial step in the automated process of MCQ generation because the difficulty of each question highly relies upon its distractors. A good distractor is one that is very similar to the key but not the key itself. The goodness and toughness of an MCQ question depend on how close the distractors are to the key. The closer the distractors are to the key, the more the difficult is the question. In this paper, the closeness or the similarity measure between the distractors and the key is found through hypernyms and hyponyms of key.

**Hypernym** - Hypernym is a word that names a broad category that includes other words.

Ex: Hypernyms of India can be Common Wealth Republic and Asian Country.

**Hyponym** - A word of more specific meaning than a superordinate term applicable to it.

Ex: Hyponym of Asian countries are India, Pakistan, Bangladesh, and Afghanistan.

Here, all the possible hypernyms of the key and the corresponding hyponyms for each of the hypernyms are found. These hyponyms are considered as potential distractors. Then, the potential distractors are ranked, and the final distractors are chosen based on their rank. For example, ‘stack’ is assumed to be the key. One of the hypernyms for the key is - linear data structure of which hyponyms are queue and linked list, which are good distractors for the key. The ranks given to the potential distractors are based on whether it occurs in the text extracted from the web pages. The potential distractors that exist in the extracted text and having the same part of speech structure as the key have a higher rank than those with a different structure. This is because distractors with the same structure as the key tend to confuse the test takers more. Any three potential distractors with higher ranks are chosen at random as the final distractors. Figure 3 shows the detailed approach followed to generate the distractors.
faced. We use Wikipedia to find the hypernyms and hyponyms by traversing through the hyperlinks in the page. This method eliminates the need to construct a static ontology graph thus eliminating the storing requirements. The domain from which distractors are generated is also humongous.

**Algorithm 3 Generation of Distractors**

<table>
<thead>
<tr>
<th>Input: Key k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: A set of distractors D</td>
</tr>
<tr>
<td>PotentialDistractorList1 = {}</td>
</tr>
<tr>
<td>PotentialDistractorList2 = {}</td>
</tr>
<tr>
<td>NewDistractorList = {}</td>
</tr>
<tr>
<td>Begin</td>
</tr>
<tr>
<td>POS tag the key</td>
</tr>
<tr>
<td>If key is a cardinal</td>
</tr>
<tr>
<td>Generate distractors having the values key +/- 1 and/or key +/- 2</td>
</tr>
<tr>
<td>Shuffle the distractors and the key</td>
</tr>
<tr>
<td>Exit</td>
</tr>
<tr>
<td>Find all the possible hypernyms of the key</td>
</tr>
<tr>
<td>For each hypernym H found</td>
</tr>
<tr>
<td>Begin</td>
</tr>
<tr>
<td>Find all hyponyms of H</td>
</tr>
<tr>
<td>For each hyponym H_y</td>
</tr>
<tr>
<td>PotentialDistractorList1.add(H_y)</td>
</tr>
<tr>
<td>End</td>
</tr>
<tr>
<td>Tokenize all sentences in the summary</td>
</tr>
<tr>
<td>For each sentence S in summary</td>
</tr>
<tr>
<td>Begin</td>
</tr>
<tr>
<td>POS tag each sentence</td>
</tr>
<tr>
<td>Let W be the word/phrase which has same POS as the key</td>
</tr>
<tr>
<td>PotentialDistractorList2.add(W)</td>
</tr>
<tr>
<td>End</td>
</tr>
<tr>
<td>For each item W in PotentialDistractorList1</td>
</tr>
<tr>
<td>Begin</td>
</tr>
<tr>
<td>If W is in PotentialDistractorList2</td>
</tr>
<tr>
<td>NewDistractorList.add(W)</td>
</tr>
<tr>
<td>Else if W has any common phrase/word as that of key</td>
</tr>
<tr>
<td>NewDistractorList.add(W)</td>
</tr>
<tr>
<td>End</td>
</tr>
<tr>
<td>If Number Of Distractors is not sufficient</td>
</tr>
<tr>
<td>Begin</td>
</tr>
<tr>
<td>Choose random item W from PotentialDistractorList1</td>
</tr>
<tr>
<td>NewDistractorList.add(W)</td>
</tr>
<tr>
<td>End</td>
</tr>
<tr>
<td>Randomly choose required number of distractors from NewDistractorList for options</td>
</tr>
<tr>
<td>Append key to NewDistractorList if not already present in options</td>
</tr>
<tr>
<td>Shuffle the options</td>
</tr>
</tbody>
</table>

**4 Generation of analogy questions**

The next process is generation of the analogy question that is based on comparison between two objects and their point of resemblance. Hence, underlying relationship plays a major role in analogy questions. In multiple choice analogy questions, students are given a pair of words and are asked to identify the most analogous pair among the set of alternatives. The level of difficulty is based on how close the distractors are chosen.

**Example:**

Stem: Goat : grass

- a) Tiger : lion
- b) Cat : milk
- c) Tiger : meat
- d) Cow : tail

Key: Tiger : meat

**Definition 1**

Let Q be an analogy question with relation R = (A,B), the key K = (X,Y) and the set of distractors D = \{Di = (Ei,Fi) | i = 1 to 3\}. For a good analogy question, the following conditions must be satisfied:

- Question Q, key K, and distractors D are all properly related to itself.
- Key K is more analogous to Q than other distractors.
- Distractors D’s relation is very close to each other but not same.

The stem is extracted from the given input comprehension passage. Pattern matching is used to identify patterns of the type ‘noun phrase’ followed by ‘verb phrase’ followed by ‘noun phrase’ from the passage. Here, the verb phrase gives the relationship between the two noun phrases that surround it. The stem is taken to be ‘NounPhrase1: NounPhrase2’. The key and distractors are chosen based on the verb phrase or the relation. The following similarity measure is used to determine the key and distractors:

\[
\text{Similarity} (Q,S_i) = \begin{cases} 
< 1, & \text{same, } P = PS_i \\
1, & \text{similar, } p \neq PS_i \\
> 1, & \text{not similar}
\end{cases}
\]

Where P is the priority of the relation between the noun phrases of the stem and PS_i is the priority of the relation between the noun phrases of sentence S_i taken from the
corpus. When the priorities of two relations are equal, then it implies that they can be used interchangeably. If difference in priorities of the two relations is 1, then the two relations are close or similar to each. If the difference is greater than 1, they are different. Thus, the key is chosen from a sentence with Similarity \((Q, S_i) < 1\), and distractors are chosen from sentences with Similarity \((Q, S_i) = 1\). Retrieving distractors from text require huge corpus, so a corpus of text from which distractors can be retrieved is stored. Then, the generation of the analogy question structure is shown in Figure 4.

**Figure 4** Generation of analogous questions (see online version for colours)

**Algorithm 4** Generation of Analogous Questions

\[
R \leftarrow \text{Relation classes with their priority}
\]

**Priority:**

- Similar classes – same priority
- Less similar classes – adjacent priority

\[
T \leftarrow \text{Given text.}
\]

\[
C \leftarrow \text{Corpus of text for distractors and key.}
\]

Input: Paragraph \(P\)

Output: Analogous question with choices

*For each sentence in \(T\),*

*Begin*

*Scan the passage \(P\) to find patterns of the form \(NP_1 + VP + NP_2\)*

*Relation \(R = VP\)*

*Question \(Q = NP_1:NP_2\)*

*Choose key \(K\) at random from classes such that \(\text{Priority}(\text{class}) = \text{Priority}(R)\)*

*Choose distractor1 from classes such that \(\text{Priority}(\text{class}) = \text{Priority}(R) + 1\)*

*Choose distractor2 from classes such that \(\text{Priority}(\text{class}) = \text{Priority}(R) - 1\)*

*Choose distractor3 from classes such that \(\text{Priority}(\text{class}) = \text{Priority}(R)\) and by reversing the noun phrases as \(NP_2:NP_1\)*

*Shuffle the distractors and key*

*End*

Based on the above process, the effective analogy questions are generated by analysing the similarity and preference of the information which is done with the help of the fireflies-based preference learning method. The efficiency of the system is further evaluated with the help of the experimental results and discussions.

### 5 Results and analysis

In this paper, web corpus is used to make it feasible to generate questions from any user-specified domain. Using web to retrieve information introduces the following challenges. Web pages are not as structured as databases. They are semistructured, i.e. they contain both structured components such as tables as well as unstructured components such as free text, multimedia, hyperlinks, and HTML tags. So, it is very much important to extract substantial amount of meaningful texts from a web page to generate MCQs from those texts. On the other hand, the information which is extracted from the web pages must be relevant to the query given by the user. We may not retrieve all the web pages relevant to the query, and hence, we may not be able to extract all the relevant information. Because our project uses google API, FScore of our text extractor module is same as that of the Google API. So, our model has a considerably high FScore, and hence, the information which is extracted is highly relevant to the query. The following measures are used to determine the overall performance of the automatic fill in the blank type question generator:

\[
\text{Accuracy of summariser} = \frac{\text{No. of usable questions generated}}{\text{Total no. of questions generated}}
\]

\[
\text{Accuracy of blank generator} = P\left(Q_s | \text{Questions is usable}\right)
\]

\[
\text{Accuracy of distractor} = P\left(D_s | Q_s\right)
\]

where usable questions are the questions satisfying the following properties:

- Stem is relevant to the query.
- Stem is informative enough to make question of high standard.

\(Q_s = \text{No. of questions with blank in suitable position.}\)

\(D_s = \text{No. of questions having distractors very close to key.}\)

A set of 93 questions were generated from 10 wide-ranged domains to test the accuracy of the automatic question generator. Out of the 93 questions, all were found to be relevant to the query. 67 out of the 93 relevant questions were found to be usable. Thus, the accuracy of the summariser can be calculated as

\[
\text{Accuracy of summariser} = \frac{67}{93} = 0.7204 \approx 72\%
\]
Out of these questions, 52 had the blank at the suitable position. The distractors were found to be very close to key in 41 of these questions. Thus, the accuracy for blank generation is 77.6%, and the accuracy for distractor generation is 78.8% (calculated using the above formulae). Figure 5 shows comparison of our model with existing models of the accuracy of the summariser and accuracy of blank generator.

From Figure 5, it clearly shows that the proposed system consumes higher efficiency while analysing the summariser also generating the blanks when compared with the other existing methods. The increased efficiency value leads to increase the efficiency of the distractors which is shown in Figure 6.

The accuracy of analogous question generator is found to be very high (almost all questions are of good standard) because distractors are chosen from predefined dataset. However, the set of distractors is limited. Also fewer questions are generated because it is difficult to find stem suitable for analogous type questions in a general comprehension passage. Thus, the proposed system successfully generates the automatic multiple choice question in the e-assessment system.

6 Conclusion and future work

Multiple choice questions of fill in the blank type and analogous type were successfully generated. Questions of good standard were produced with reasonably high accuracy (higher compared with the existing models). The accuracy of generating distractors given that the stem and blank are well chosen is extremely high. Thus, this paper has various applications that include generating technical question by giving a query to test students in a specific concept, generate MCQs by giving a text book or passage as input to test the students on their understanding of concepts in the input text, generate analogous MCQs on a comprehension passage to test the ability of a student to identify the underlying relation between different objects (tests the verbal reasoning ability).

In future, the accuracy of the proposed model can be improved by using n-gram POS tagger instead of unigram model used here. However, the n-gram model takes much longer time to POS tag the sentence, and hence, an optimised algorithm has to be carefully designed keeping this in mind. The accuracy of the summariser can be improved by using coreference resolution. In the model presented, we do not include any sentence that contains a pronoun into the summary because pronoun makes the sentence incomplete, which does not have enough information for question to be answered. However, there may be certain important and informative sentences with a pronoun in it. Replacing the pronouns by corresponding nouns and then summarising the text will give a better summary, including most of the informative sentences. Coreference resolution, the task of finding all expressions that refer to the same entity in text, can be used to identify the noun corresponding to the pronoun. Furthermore, this project can be extended to evaluate students through e-assessment. The generated questions can be given as an online test, and a module to evaluate the answers submitted can be developed. This can be done with ease as all the questions are objective type and the key is known in prior (the answer cannot vary from student to student as in case of subjective questions).

References


