Educational and Information Technologies (2019) 24: 973–993
https://doi.org/10.1007/s10639-018-9814-3

A hybrid approach for automatic generation of named entity distractors for multiple choice questions

Rakesh Patra[1] · **Sujan Kumar Saha**[1]

Received: 11 June 2018 / Accepted: 14 September 2018 / Published online: 21 September 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract

Assessment plays an important role in learning and Multiple Choice Questions (MCQs) are quite popular in large-scale evaluations. Technology-enabled learning necessitates a smart assessment. Therefore, automatic MCQ generation became increasingly popular in the last two decades. Despite a large amount of research effort, system generated MCQs are not useful in real educational applications. This is because of the inability to produce diverse and human-alike distractors. Distractors are the wrong choices given along with the correct answer (key) to befuddle the examinee. In several domains, the MCQs deal with names or named entities. However, existing literature is not adequate in generating quality named entity distractors. In this paper, we present a method for automatic generation of named entity distractors. The technique uses a combination of statistical and semantic similarities. To compute the statistical similarity, a set of class-specific attributes are defined are their values are extracted from the web. Semantic similarity is computed using a predicate-argument extraction based method. The proposed technique is tested in cricket domain because of the availability of a large number of web resources and MCQs for dataset preparation. An evaluation strategy is proposed along with a set of metrics. A set of human evaluators performed the evaluation and they found that the average closeness value of the distractors as 2.3. This value indicates that 2.3 out of 3 system-generated distractors are as good as human-generated distractors. Two good distractors make an MCQ usable in real assessment. So, the proposed technique is capable of generating high-quality distractors.

Keywords Multiple choice questions · Educational assessment · Distractor generation · Named entity · Web information · Text mining

✉ Sujan Kumar Saha
sujan.kr.saha@gmail.com

[1] Department of Computer Science and Engineering, Birla Institute of Technology Mesra, Ranchi, 835215, India

1 Introduction

Multiple choice question (MCQ) is a very popular form of assessment in which respondents are asked to select the best possible answer out of a set of choices. Due to many applications, a substantial amount of research effort has been devoted for automatic MCQ generation (Coniam 1997; Mitkov and Ha 2003; Brown et al. 2005; Papasalouros et al. 2008; Aldabe and Maritxalar 2010; Agarwal and Mannem 2011; Bhatia et al. 2013; Majumdar and Saha 2014a; Araki et al. 2016). An MCQ is composed of three core elements: stem, key, and distractors. The *stem* (also known as *item*) is the sentence from which the question is formed, *key* (also named as *target word*) is the correct answer of the question and *distractors* are the set of wrong answers or choices.

Distractors are the wrong choices given along with the correct answer to befuddle the examinee. The distractors play an important role in cloze question generation, as the quality of an MCQ largely depends on the quality of the distractors. If the distractors are not able to sufficiently confuse the examinee, the correct answer can be chosen easily. As a result, the overall quality and usability of the MCQ degrade. Mitkov et al. (2009) stated that good quality distractors ensure that the outcome of the tests provides a more credible and objective picture of the knowledge of the examiners involved. Again, poor distractors would not contribute much to the accuracy of the assessment as obvious or too easy distractors will pose no challenge to the students. Also in absence of good distractors, the MCQs will not be able to distinguish high performing from low performing students.

Let us consider a question from the sports domain: "Who was the highest run scorer of the India side in the final match of Cricket World Cup 2011?". The answer is 'Gautam Gambhir'. If the distractors are given as, *Ricky Ponting*, *Adam Gilchrist* and *Chris Gayle* then the MCQ becomes silly. Because among these options Gautam Gambhir only is from India and becomes the obvious choice. Those who have little knowledge on world cricket, they will easily answer the question. Again, if the distractors are *Sunil Gavaskar*, *Harbhajan Singh* and *Zaheer Khan* then also the MCQ fails to meet the quality; because they are Indian but either not played in 2011 World Cup or are bowlers who are unlikely to become the highest run scorer. But in this question, if the distractors are taken as, *Sachin Tendulkar*, *Yuvraj Singh* and *Mahendra Singh Dhoni*, then the MCQ becomes practical.

A distractor is basically a concept semantically close to the keyword. In many domains (e.g., sports, entertainment, travel, history, biomedical) named entities are dominant,where the majority of the MCQs deal with the names. However, the literature does not contain sufficient works that particularly focus on named entity distractor generation. General distractor generation techniques include similarity in parts-of-speech (POS) tag (Brown et al. 2005; Liu et al. 2005), frequency count (Coniam 1997; Brown et al. 2005), distributional similarity (Karamanis et al. 2006; Afzal and Mitkov 2014), phonetic similarity (Correia et al. 2010), pattern matching (Hoshino and Nakagawa 2007; Goto et al. 2010) etc. These are not effective in name distractor generation as two similar names might not possess these similarities. For example, two similar names 'Tendulkar' and 'Dhoni' do not guarantee similar

frequency in large corpora, not phonetically similar or distributional similarity is also different. WordNet based approach is widely used in generic distractor generation (Mitkov and Ha 2003; Lin et al. 2007; Goto et al. 2010). However, WordNet is not applicable here as it does not include the names. Domain ontology-based approaches (Karamanis et al. 2006; Papasalouros et al. 2008) also fail here as it is not feasible to capture all possible names of a domain in an ontology (Afzal and Mitkov 2014). Patra and Saha (2017) presented a study on the use of web information for automatic generation of named entity distractor. First, they defined a set of attributes for each name class. Then gathered a list of names and collected their attribute values from the web. The names having similar attribute values with the keys were chosen as distractors. This approach is having limitations including, performance depends on the coverage of the list and time-consuming. We worked on these limitations to develop a more efficient and robust system.

In this paper, we present a system for generation of named entity distractors. The system takes the stem and key (which is a named entity) as input and generates three distractors as output. For distractor generation, two types of similarity computation are performed: statistical and semantic. In a certain domain, like sports, statistical data might help to distinguish an entity. The system uses the entity statistics to choose the distractors. For extraction of entity statistics, a set of attributes is defined and the attribute values for a particular entity are collected from certain trusted websites. As the second approach, a predicate-argument extraction based approach is employed. A set of sentences related to the entities are collected from the web, from which the predicate-argument are extracted. The amount of matching in predicate-arguments between the key and a candidate distractor is considered as the semantic similarity. Runtime extraction of statistics from the web and similarity computation is time-consuming. Therefore, we use class-specific repositories created during the development phase. Additionally, we propose a hierarchical clustering based representation of the entities, that embeds the similarities between the entities in a tree structure. The system finds the location of the key in the tree and picks the nearby entities as the distractors.

The proposed technique is applied in sports (cricket) domain. For the evaluation, we propose an evaluation strategy and three metrics namely, closeness, readability, and relevance. Closeness is the primary metric for evaluation; it indicates the similarity between the key and the generated distractors. We create a test data containing 200 cricket MCQs primarily collected from the web. The system generates three distractors for each of the MCQs and these are evaluated manually by twenty evaluators. The system achieves an average closeness value of the distractors as 2.3. This value indicates that 2.3 out of 3 system-generated distractors are as good as human-generated distractors. The readability and relevance values are also high. These values demonstrate that the proposed system generates quality named entity distractors.

2 Previous works on distractor generation

We discuss below the techniques applied in the literature for distractor generation.

2.1 POS and frequency

Parts-of-speech information has been used as a clue for distractor generation in several works. Both the key and distractors should belong to similar POS category. Additionally, frequency of the words is another hint. Brown et al. (2005) observed that the distractors are of the same POS and similar frequency to the synonym, antonym, or whatever word is the correct answer, as opposed to the target word. Similarly, Coniam (1997) also performed frequency analysis for choosing distractors. They also used POS information and certain orthographic information like capitalization to choose the distractors. Liu et al. (2005) considered POS information and word frequency based approach for distractor generation. Agarwal and Mannem (2011) used a hybrid technique to choose the best from a set of potential distractors. However, the set of potential distractors was generated based on POS information.

2.2 WordNet and ontology

WordNet is widely used for distractor generation. WordNet is a lexical database that groups the words into sets of synonyms (called synsets) and records the relations among these synonym sets or their members. Mitkov and Ha (2003) consulted WordNet to retrieve concepts semantically close to the correct answer. Hypernyms, hyponyms, and coordinates of the key was retrieved from the WordNet. If WordNet returned too many concepts, those appearing in the corpus or textbook were given preference. Mitkov et al. (2009) employed multiple strategy for computing WordNet-based semantic similarity. Lin et al. (2007) also used the WordNet and synset relationship to find the distractors. Such WordNet based distractor generation may lead to ambiguity and difficulty to choose the correct distractor. Therefore, they eliminated all synonyms and similar words belonging to the synset that the key was obtained from. Goto et al. (2010) also used WordNet for finding the synonyms and antonym words as distractors.

Domain ontology is another related resource that help to identify domain specific related terms as distractor. Papasalouros et al. (2008) used an ontology to generate distractors. They defined various strategies based on these classes and properties or relationship to find the distractors. They defined strategy to choose distracters as the related instances or classes having similar properties with the correct answer. UMLS (Unified Medical Language System) may be viewed as a comprehensive thesaurus and ontology of biomedical concepts. Karamanis et al. (2006) computed a set of potential distractors for a key term using the terms with the same semantic type in UMLS. Majumdar and Saha (2014b) used domain information extracted from the web for generation of distractors in sports domain.

2.3 Distributional similarity and collocation

The distributional hypothesis says that similar words appear in similar context. This hypothesis has been used by a group of researchers to extract the distractors. Karamanis et al. (2006) used distributional similarity measures that compute similarity between words in terms of the similarity of contexts of their occurrences in a corpus.

Mitkov et al. (2009) also used distributional similarity based approach for distractor generation. For computing distributional similarity, they used Information Radius. Information Radius measures similarity between two words as the amount of information contained in the difference between the two corresponding co-occurrence vectors. Aldabe et al. (2009) used distributional similarity and information radius based measures to find distractors. For measuring the information radius, contextual representation was extracted along with frequencies then they were compared using the distributional hypothesis. Afzal and Mitkov (2014) also used distributional similarity for generating distractors and NE distractors. They used the GENIA corpus and GENIA tagger in their task. First they applied linguistic processing; after that they built a frequency matrix which involves scanning sequential semantic classes (NEs) along with a notional word (Noun, Verb, Adverb and Adjective) in the corpus and recording their frequencies in a database. Then the semantic classes were compared using the distributional hypothesis that similar words appear in similar context. The distractors to a given correct answer were then automatically generated by measuring its similarity to entire candidate named entities. Finally, they selected the top 4 similar candidate named entities as the distractors.

Lee and Seneff (2007) used a collocation based approach for preposition distractor generation. Their method returns the preposition that appears frequently with either A or B, but not both at the same time and those were considered as the distractors. Similarity between two sentences, one contains the key and other one contains the distractor, is considered as a feature for distractor generation by Agarwal and Mannem (2011). They also used contextual similarity for the task. Contextual similarity computes the similarity between the key and the potential distractor using the surrounding words. Fattoh (2014) used a sentence similarity based approach for keyword selection. They applied semantic role labelling and named entity recognition on the selected sentences; and identified the key. Now they identified three sentences that are having highest closeness with the question sentence. Then they selected keywords using similar method from these close sentences. These keywords act as distractors.

2.4 Phonetic distractors

Phonetic similarity based distractor generation is another technique. Correia et al. (2010) aimed at exploring the most common spelling errors for the Portuguese language and generated a table containing common phonetic mistakes. For example, ‘ss’ is frequently confused with ‘ç’ (before a, o and u) and ‘c’ (before e and i). Such information is used to generate phonetic distractor from the key. Phonetic similarity based distractor generation was also experimented by Mitkov et al. (2009).

2.5 Pattern matching based

Pattern matching is another technique for distractor generation that attracted several researchers. Hoshino and Nakagawa (2007) used a pattern matching based approach for distractor generation. However, their patterns were based on the parse tree and grammar rules. They manually identified ten grammar targets and corresponding parse tree based patterns. For example, a target is past perfective and corresponding

parse tree pattern is (S ... (VP (AUX had) ...) ...). Similarly, pattern based verb distractor generation was also experimented by Aldabe et al. (2009). Their patterns targeted to match the similarity between the key and the distractor with respect to the context information. Goto et al. (2010) used pattern based method for distractor generation. For example, they defined patterns for generating derived words (e.g., worker, works, working etc. can be derived from work), shape based alternative (e.g., words having similar suffix and prefix) as distractors.

2.6 Semantic analysis

As the distractors are the concepts that are semantically close to the keys, semantic analysis is performed by the researchers for choosing the distractors. Aldabe and Maritxalar (2010) used Latent Semantic Analysis (LSA) to generate distractors. To use LSA they first build a WordSpace using large corpora and then performed word to word semantic similarity computations. To generate distractors, Pino et al. (2008) computed semantic similarity between two words using the Patwardhan and Pedersen's method. Their method made use of a corpus to find context vector, that was basically computed by counting the co-occurrences of the word with other words. Then, the dot product of the context vectors was taken as the semantic similarity. For selection of candidate distractors, Aldabe et al. (2009) computed semantic similarity between verbs. Kumar et al. (2015) used word2vec based method for distractor generation. To run word2vec they used Wikipedia text. However, they mentioned that the quality of those distractors were not as good as human generated distractors. Araki et al. (2016) used a semantic analysis based approach for distractor generation. For generation of distractors they built event graph using event triggers and event-event relationship. A distractor is constructed by selecting the phrase comprised of a selected node (event trigger) and its surrounding entities.

3 Distractor generation: Our approach

For the generation of the distractors, we define two similarity functions that make use of two sets of information: (i) statistical and (ii) semantic. Finally, these two functions are combined. The entities having higher similarity value are chosen as the set of final distractors. Figure 1 presents a high-level overview of the approach. The details of the methodology are presented in the following sections.

4 Statistical similarity computation

This section discusses our approach for computing the statistical similarity between the key and the possible distractors. We perform the study on cricket domain and use certain domain-specific basic information (e.g., like possible name classes, set of good attributes of a particular class etc.) to develop the system. However, the approach is mostly generic and can be applied to other related domains. Here we would like to mention that the basic idea of this statistical similarity computation is

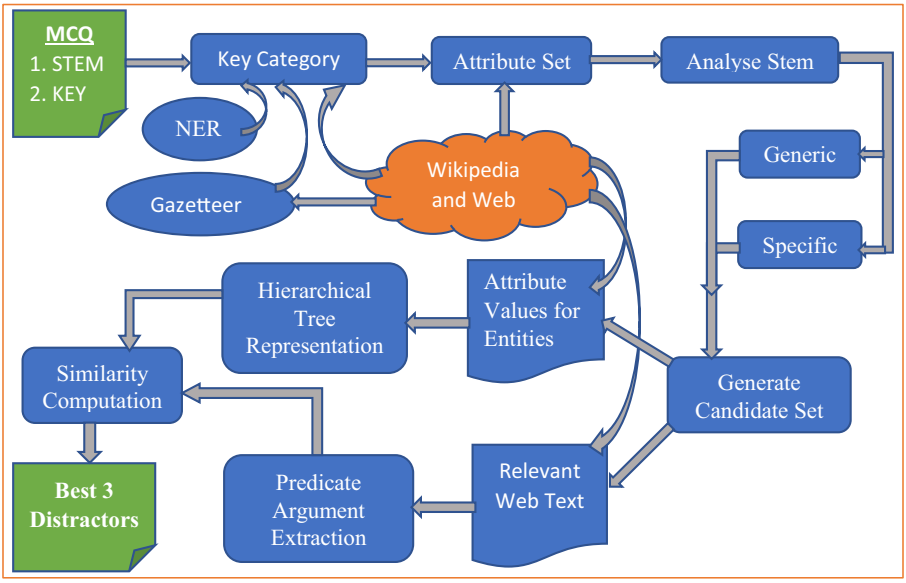

Fig. 1 A high-level overview of the distractor generation approach

motivated from the work by Patra and Saha (2017), but our system is more robust and fast as it is not relying on the name list and uses a hierarchical tree structuring for storing the similarity values.

4.1 Identifying the category of the key

For automatic generation of distractors, first, we need to identify the category of the key. Key category provides the search space for finding the possible distractors. For example, if the key is an umpire, then we obviously need a list of umpires as possible distractors. In general domain, most widely accepted name categories are person, location and organization. But this 3-class categorization is not sufficient to perform the NE distractor generation task. It requires a deeper or fine-grained categorization. Such fine-grained categorization is task specific as well as domain specific. We study the existing MCQs from the cricket domain and find various NE categories. These are, name of person (basic subcategories are: all-rounder, bowler, batsman, wicketkeeper, captain, umpire, cricket officials like board president, team owner etc)., organization name (country name, franchise name, cricket boards like ICC etc.), event name (cup, tournament, trophy, championship etc.), location name (cricket ground, city, country etc.). Use of a named entity recognition (NER) system is the primary choice to identify these categories. But unfortunately, we could not find any openly available NER system for the sports domain too, that is designed for identifying these classes. Therefore, we use a coarse level NER (Stanford NER system[1]) and then apply two more approaches on it. These are discussed below.

[1] https://nlp.stanford.edu/software/CRF-NER.shtml

Table 1 Gazetteer lists

Name category	Source	List size
Cricketer name	Wikipedia, *Yahoo*! Cricket, Espncricinfo	3978
Umpire and Referee	icc-cricket.com, Wikipedia	590
Other officials, Owners	Wikipedia, Country Boards like ecb.co.uk	536
Cricket teams	Espncricinfo, *Yahoo*! Cricket	325
Cricket organization	icc-cricket.com, Wikipedia	147
Cricket event	cricinfo archive	2324
City and Ground	Wikipedia, cricinfo grounds	742

Gazetteer List Based Our first approach is gazetteer list based identification. If the word occurs in a gazetteer list of a particular class, then it belongs to that category. But the creation of a gazetteer list that contains all the names of a particular category is very difficult, as new names are constantly included. In this study with the sports domain, we compile the gazetteer lists for the key categories with the help of a list of trusted websites: Wikipedia, EspnCricinfo, Yahoo! Cricket, CricBuzz, Cricwaves. We use these websites for the creation of the gazetteer lists. For example, to compile the list of batsman we search the Wikipedia, Yahoo! Cricket, Espncricinfo, and Cricbuzz players lists. From these sources, we compile a batsman gazetteer list containing a total of 3978 names. Similarly, we compile other domain-specific gazetteer lists. In Table 1 we have summarized the gazetteer lists we create for the cricket domain.

Using Wikipedia Our second approach is the extraction of category information from the Wikipedia. In case of named entity keys, it is expected that a Wikipedia page is available for the key from which the class related information is extracted. We study various Wikipedia pages and observe that the category related information is embed ded in the first sentence (or, first two sentences in some cases) and the Wikipedia *infobox* content. In the 'infobox' content we search for the label 'role' as the main category label. If the infobox is available and it contains the category information, then the label is extracted from it. Otherwise, the first two sentences of the page are processed to extract the required information. For extraction of category information from the sentences, two different methods are used. The first method is parse tree based. Parse tree of the sentence is explored to find the is_a predicate having the key as the subject; then the corresponding object is the category label. For predicate-argument extraction from the sentences, we use the triplet extraction methodology discussed in Rusu et al. (2007). The second one is simple keyword matching based. As we already have a list of predefined categories, we search for the category denoting word or phrase in the sentence. If obtained, then the key belongs to the corresponding category. In our experiments, we observe that the second method is very effective.

4.2 Analysis of the stem

Next, we analyze the stem (i.e., the question text) to extract the information embedded in it. The stem plays a big role in distractor generation. It contains certain clues that help in refining the search space. Let us consider an example to explain the role of the stem in distractor generation. The key for the stem "Who scores most fifties in World T20 2016?" is *Virat Kohli*. Good distractors for the MCQ are, Tamim Iqbal, HM Amla, JE Root etc. Also for the stem "Who scores most fifties in T20 internationals?", the key is *Virat Kohli*. Here a good distractor set should include CH Gayle and BB McCullum. As another example stem, "who is the current captain of Indian test cricket team?" - the key is again *Virat Kohli*, but here the distractors set includes MS Dhoni and Ajinkya Rahane. So, the key alone is not sufficient to generate the close distractors; the stem is also important.

The stem might contain some information that helps to redefine the search space. We need to extract this information if any. Depending on availability of special information in the stem, the stem is classified into two categories: *generic* and *specific*. If any such information is available then the stem is named as specific, otherwise generic. Type of information embedded in the stem depends on the characteristics of the domain. Therefore, the methodology to employ here is also domain-specific.

We study many existing MCQs of the cricket domain and observe that the information in the stem that modifies the search space is, specific to something (mainly, match, tournament, and team) and record related. If such information is there then the stem is specific, otherwise, the stem is labeled as generic. Availability of phrases denoting specific match, tournament, series, location, country is searched in the stem. List based approach is used here as the required lists are available. To find the record or statistics related phrases, list of class-specific records is collected from espncricinfo.com and Wikipedia. For example, the record modifiers of the class 'batsman' are: most runs, high scores, highest average, highest strike rate, most fifties, most centuries, most ducks, most sixes, most sixes in an innings, most fours, most fours in an innings, highest run in a calendar year, fastest century, maximum distance of over boundary.

4.3 Attribute set

For each of the name categories, a set of attributes is defined. Through these attributes, we try to define a category. The attribute values corresponding to a particular entity helps in finding its similarity with other entities. For example, the attribute set of the category 'cricketer' includes date-of-birth, country, team name, role, batting style, average batting position, carrier span, debut date (test-ODI-T20) (ODI: one day international, T20: twenty-twenty), last match (test-ODI-T20), total runs (test-ODI-T20), batting average (test-ODI-T20), number of innings (test-ODI-T20), strike rate (test-ODI-T20), number of century (test-ODI-T20), number of half-century (test-ODI-T20), highest score (test-ODI-T20), bowling style, number of balls (test-ODI-T20), number of wickets taken (test-ODI-T20), runs conceded (test-ODI-T20), best bowling figure (test-ODI-T20), bowling average (test-ODI-T20), 5-wickets in innings (test-ODI-T20), catches taken as wicketkeeper (test-ODI-T20), catches taken as non-wicketkeeper (test-ODI-T20).

To develop the system, we need to specify these set of attributes. Some amount of domain expertise or manual effort is required to do the task. However, this task is not entirely manual. Wikipedia and domain related websites provide clues in choosing these attributes. Here we discuss the method we adopt in cricket domain. A similar methodology can be followed in other domains.

The attributes are chosen using the fields used for summarizing the entity (e.g., carrier or the personal details in case of cricketer) in Wikipedia and EspnCricinfo like websites. Majority of the cricket related Wikipedia pages contain an information templet (or, infobox; located at the right-hand side top of the page) that contains a set of common attributes defining the class. Also, most of the Wikipedia pages for a popular named entity contain a table to summarize the entity. The fields of the table are extracted as members of the attribute set. Similarly, the entity-specific EspnCricinfo and cricbuzz pages contain semi-structured and tabular data from which these attributes are extracted. We crawl the pages of a few popular keys of a particular category and extract these attributes in a semi-automatic manner. The duplicates and synonyms are removed or renamed. Then the most common attributes are considered as the final attribute set of the class.

4.4 Candidate distractor set

Our next task is to identify the candidate distractors. If the stem contains any specific information, then it is extracted for refining the list of distractors. For example, consider the stem: "Who was the captain of India during ICC world cup 2015? " The system identifies three specific information from the stem: 'India', 'ICC world cup' and '2015'. This information gets higher weight when the candidate distractors are chosen. These values are matched with the attribute values of the corresponding key. If any match is found then, the entities having similar matching are considered as candidate distractors. For example, here 'India' matches with the 'team-name' attribute value of the key (MS Dhoni) and year '2015' belongs to the 'carrier-span' value. So, we find the entities where *team-name = India* and *carrier-span includes 2015*. Those entities form the candidate distractor set. The set includes the entities 'Virat Kohli', 'A. Rahane', 'R. Ashwin', 'R. Jadeja', 'C. Pujara' and 'KL. Rahul'.

If the stem is dealing with statistics or record of the key, then also the search space is modified accordingly. Example of such stem is: "Who is having the most century in ODI cricket?" Here a query is formed and searched in corresponding web pages. One of the entities of the page should be the key. Then we take rest of the elements as candidate distractors. If the search result returns no value, then it is processed using the method that handles the generic stems.

When the stem is generic, then the candidate distractors are chosen as those are having attribute values close to the key. A set of class-specific attributes may be chosen as 'important'. For example, for the cricketer class, we consider the attributes country, span (overlapping), batting average (difference less than ten) or bowling average (difference less than five) as important attributes that help to choose a subset. Such identification of important attribute is not a mandatory step; if such important attributes are not identified for any category then all the attributes are equally important.

4.5 Similarity computation

Now, the similarity between the key and a candidate distractor set entity is computed to decide whether to select it in the final set of distractors. In order to compute the similarity, we first create class-specific repositories. The repositories contain a list of entities with their generic attribute values. The base information of these repositories can be extracted automatically from the Wikipedia. Wikipedia provides 'list_of_*' pages in many domains that contain the list of entities of a particular category with a list of attribute values in tabular format. A focused crawler can be employed to automatically go through the pages given a list and a set of patterns, that extracts specific information from the pages. The repositories are stored as a simple file in a matrix-like format, where each row represents an entity and the columns are the attributes. The first column refers to the name of the entity. All the attribute values may not be available for some of the entities; unavailable fields are filled with 'NULL' indicator.

To obtain the similarity we use the corresponding vectors (i.e., the row) from the entity-attribute repository. The dimension of the vectors is represented by the number of attributes. The attributes fall into two categories: numeric and non-numeric. The similarity between the non-numeric attributes is considered as binary. The value is 1 if there is a match and 0 otherwise. Some of these non-numeric attributes take multiple values. In such cases, if any match is found among the values then the similarity is taken as one. Then, these values are normalized by a total number of non-numeric attributes.

To compute the distance between the numeric attributes we employ the following distance computation metric.

$$Sim1(P, Q) = 1 - \frac{1}{L} \sum_{i=1..L} \frac{(P_i \sim Q_i)}{max(P_i, Q_i)} \tag{1}$$

Where P and Q represents two vectors corresponding to the target entities. L is the total number of numeric attributes and i is the index that iteratively consider all individual attributes P_i. Finally, the distances of the numeric and non-numeric attributes are combined.

The distance computation metric returns the distance as close to zero when the attributes of two entities (one is the key and other one is a candidate distractor) are similar.

4.6 Hierarchical tree representation of the candidate distractors

To select k best distractors we need to compute the similarity between the key and all the candidate distractors and choose top k values. This task becomes time-consuming when a large number of entities occur in the candidate distractor set. To make the process faster we plan to pre-compute and store the distances. For this purpose, we adopt a hierarchical clustering based representation of the candidate distractors.

For the hierarchical clustering, we need to convert the words into vectors. We have already mentioned that we have defined a set of attributes and collected the attribute values from the web. This set of attribute values of an entity is considered here as the

vector corresponding to the entity where the size of the vector is the total number of attributes.

The hierarchical clustering requires to adopt a similarity metric and linkage criteria. The similarity metrics used here is the cosine similarity. The linkage in the tree is done using the complete linkage (farthest neighbor linking) criteria. With this linkage, a cluster from a pool of candidate clusters is chosen if the similarity value of the farthest node is higher than the farthest nodes of other clusters in the pool. The tree is a binary tree. The weights at each edge are the least similarity value found between the connecting subtree clusters. Now, the distance between two entities is the closest tree level that connects both of them. This hierarchical distance is normalized using (3).

$$d_1(x, x') = highest\ tree\ level\ connecting\ x\ and\ x' \tag{2}$$

$$Sim2(x, x') = \frac{d_1(x, x')}{\sqrt{d_1(x, x) \odot d_1(x', x')}} \tag{3}$$

Once the tree is generated, we simply find the position of the key in it and pick the entities occurring near the key.

5 Semantic similarity computation

Next, we compute the semantic similarity between the key and candidate distractors to choose the most appropriate set. Semantic closeness is computed through predicate-argument extraction from the sentences. For this, we collect the text from the corresponding Wikipedia pages and profile description text of EspnCricinfo and CricBuzz. The number of predicate-argument matches between the key and a candidate distractor indicates the similarity between the two.

5.1 Predicate-argument extraction

Predicate and the arguments like subject and object of a sentence are required to analyze to extract the fact embedded in a sentence. Predicate[2] helps to get an idea about the subject, such as what it does or what it is like. A predicate can be viewed as the relation or function over the arguments. The predicate serves either to assign a property to a single argument or to relate two or more arguments to each other. Such properties or relations can be represented through a functional representation of the form: Predicate (Subject, Object). Such relations are also referred to as 'triplet' in the literature. For example, a triplet generated from the sentence "Virat Kohli currently captains the India National Team" is: *captain (Virat Kohli, India National Team)*.

For a predicate-argument structure or triplet extraction from the relevant sentences, we primarily use the methodology discussed in Rusu et al. (2007). But to handle the application specific and domain-specific requirements, a filter is applied

[2] https://en.wikipedia.org/wiki/Predicate_(grammar)

to the original Rusu's triplet extraction methodology. The actual algorithm prefers to pick the head-noun or first noun in the NP subtree as the subject. But in this task, we are interested to collect information related to the key only. Therefore, only the triplets having the key as a subject are extracted. Additionally, we are only interested in the predicates that are common in characterizing the domain but not frequent in other domains. To find the list of domain specific predicates the tf*idf score is used, where the non-domain set contains a large number of Wikipedia pages that are not related to sports.

Wikipedia sentences are often long and one sentence contains multiple objects. For triplet extraction, first, the ARKref tool[3] is applied for pronoun resolution. Then a parse tree based sentence simplification step is used that converts the complex and compound sentences into simple sentences. Still, some sentences contain multiple predicates. Here our objective of triplet extraction is finding similarity by counting the number of matches. Extraction of all the predicates increases the amount of match. To extract multiple predicates and distinguish between the arguments, the words with 'IN' parts-of-speech label in the preposition phrases (PP) of the sentence are used. The IN word is added with the actual predicate to generate multiple variations of it. Let us consider an example sentence to make it clearer. The sentence "Player1 scored a century in World Cup 2011 against TeamX". The predicate is 'scored', the subject is 'Player1' and the predicate-argument extracted from the sentence are, *scored (Player1, century)*, *scored_in (Player1, Century, World Cup 2011)*, *scored_against (Player1, Century, TeamX)*.

5.2 WordNet for predicate resolution

Next step is the predicate resolution. Multiple predicates might be used to present a particular fact. In the cricket domain also we observe that for presenting a particular knowledge, a variety of predicates have been used. For example, "X scored a century" and "X hit a century" - both are presenting the same fact but the predicates are different. So, we need to find a relation between these predicates. For finding this relation we use the WordNet. If two predicates are the synonym to each other as per the WordNet, then we map them into one.

5.3 Similarity computation

The predicate-argument extraction module extracts all the triplets from the text corresponding to an entity. These triplets are considered as a fact-based summarization of the entity. It is observed that two closely related entities share a large number of common triplets. Again, two entities from two different categories do not share common triplet. Therefore, a number of triplets commonly occurred in a pair of entities represents the similarity between them. For triplet matching, we need to replace both the entities with a common variable first.

[3] https://github.com/brendano/arkref

To get the semantic similarity score between two entities (one is the key and another is a candidate distractor), a number of common triplets are counted and then the value is normalized. For that (4) is used. In the equation, x and x' denote the entities - x is the key and x' is a candidate distractor. As the target is to find the most suitable distractors when the key is known, the size of the triplet set corresponding to the key is used as the normalization factor.

$$Sim2(x, x') = \frac{|(triplet_i \in x)\ \&\ (triplet_i \in x')|_i}{|\ triplet_j\ in\ x\ |_j} \tag{4}$$

Now we have two similarity values, one is computed through statistical information collected from the web and another one is computed through semantic analysis of the relevant text. As a final similarity value, a linear combination of these two is used. The entities are ranked based on this similarity value. Top three entities of the ranked list are taken as the final distractors. When we test the system using the question "Who was the highest run scorer of the India side in the final match of Cricket World Cup 2011?", the generated ranked list contains, *Sachin Tendulkar*, *Mahendra Singh Dhoni*, *Virat Kohli*, *Yuvraj Singh* and *Virendra Shewag*. These all are good distractors.

6 System evaluation and discussion

For evaluation of the system, we create a test set containing 200 cricket related MCQ-key pairs, collected from the web. An MCQ-key pair is given to the system as input and the system generates three distractors. Quality of these system generated distractors are then assessed.

Hard comparison between gold-standard distractors and system generated distractors cannot be used as the basis of the accuracy of the system. Because in many domains and applications an MCQ can have a large set of distractors; all of these cannot be accommodated in the gold standard dataset. So, there is a possibility that the system generates good distractors but due to unavailability of those in the gold standard dataset, the calculated accuracy turns out to be poor. Therefore, in the literature, we find that the evaluation of distractors is commonly done by human experts.

6.1 Evaluation strategy used in the literature

Several metrics have been proposed and used in the literature depending on the domain, application, and type of the distractors. For example, Mitkov et al. (2009) used difficulty, discriminating power and usefulness as the metrics for distractor evaluation. Difficulty and discriminating power consider the quality of the stem also, but usefulness is an independent measure that looks at distractors only. They used statistical analysis for finding the usefulness of a distractor. Their characterization is based on a consideration that a good distractor should attract more students from the lower group than the upper group. Upper group is the set of students who score well and lower group contains the poor students. Aldabe and Maritxalar (2010) also followed a similar strategy for distractor evaluation. They identified the distractors that were

never chosen by the examinees. To judge the quality of the distractors Pino et al. (2008) replaced the keyword by the distractor and measured the grammaticality and collocation criteria of the sentence from the syntactic and semantic point of view. Agarwal and Mannem (2011) used readability as a metric. They asked the evaluators to substitute the distractor in the gap and check the readability and semantic meaning of the sentence to classify the distractor as good or bad. For distractor evaluation, Bhatia et al. (2013) used the closeness value. The distractor set was considered as good if at least one of the distractors are close to the key. Araki et al. (2016) measured the distractor quality using a three-point scale. In their scaling, 1 (worst) specifies that the distractor is confusing because it overlaps the correct answer partially or completely; value 2 concludes that the distractor can be easily identified as an incorrect answer, and 3 (best) indicates that the distractor can be viable.

6.2 Proposed evaluation strategy and metrics

When we go for evaluation of the named entity distractors, we found that most of the existing metrics are not applicable or suitable. Majority of the existing techniques focus on language learner, vocabulary testing or a particular POS category based distractors. In case of named entity distractors similarity between two names plays the key role. Therefore, during assessment also this similarity needs to be evaluated. In our evaluation, we consider the closeness as the primary metric. Closeness is the similarity between the key and the distractors. However, we also consider two other metrics: readability and relevance. To verify the readability, we follow a similar approach used in the literature. The keyword is replaced by a distractor and grammaticality and collocation criteria of the sentence are measured from the syntactic and semantic point of view. Relevance refers to the affinity of the distractors with the stem.

Assessment of closeness is a tricky task. We use two different approaches to measure the closeness. For the evaluation, we employ two groups of human evaluators: the correct answers of the MCQs are provided to one group (Group1) and this information is kept hidden to the second group (Group2). The test set questions along with the key and distractors were given to the Group1 evaluators. Their task is to check whether these system generated distractors can be used in real MCQs. Group2 evaluators get the questions only, not the options. Their task is not only to answer the questions but also to suggest three distractors for each question. Here we have three possibilities: the answer of the evaluator is correct (Category1); the answer is not correct but the correct answer is one among the distractors (Category2), and the actual answer does not match with his answer or distractors (Category3). Based on these possibilities we adopt a weighting scheme. If the evaluator can guess the correct answer, either as answer or as the distractor, the rest of the options he suggests are injected in a reference distractor set for the question. So, the reference set (set 1) holds all the options suggested by all the Category1 and Category2 evaluators. A second reference set (set 2) is also compiled that comprises of the options proposed by the Category3 evaluators. Now a machine generated distractor is searched in the reference set 1. If found, then it is considered as a perfect distractor and its score is 1; if no, then it is searched in the reference set 2 and score of 0.5 is assigned if it

occurs there. It is obvious that the score obtained by the distractors largely depends on the number of evaluators engaged in this assessment. More evaluators lead the possibility of larger reference sets and a higher chance of occurrence. In our experiments, we employ 10 evaluators and all of them are asked to provide 5 options for each question. Still, we find that the certain distractors are good but do not occur in these reference sets. Actually, a larger evaluator set is required to adopt this evaluation strategy. When sufficient evaluators are not available, then to compensate the score, the Category1 evaluators may be asked to assess those unmatched distractors with three values: 1, if one distractor is perfect to be used in real MCQ; 0.5 if it is not perfect but may be used; and 0 if it cannot be used.

6.3 Evaluation results

We employ 5 evaluators to assess the readability and relevance. A total of 200 MCQs are there in the test data and for each of the MCQs, three distractors have been generated by the system. These distractors have been picked based on the computed similarity score. That means, the first distractor is having higher similarity score with the key than the second or third distractor. These distractors are assessed by the human evaluators using three-level scoring: 1 (distractor is perfect), 0.5 (may be used but better distractors are there) and 0 (not acceptable). In our experiments, the distractors get a high score in readability and relevance. Table 2 presents the readability and relevance scores of the system. When the individual values are considered, it is observed that the score of the first distractor is often better than the others. Average readability is 92.6% and relevance is 90.33%. The score in the 3-point scale is also computed. The score is 2.88 in readability and 2.71 in relevance. From these values, we can conclude that the system is highly accurate in generating the distractors.

When we analyze the errors, we observe that the prime source of error is the failure in named entity recognition. As the generation of distractors is dependent on the name class, error in named entity recognition causes faulty distractors that affect the readability and relevance scores. We found that the named entity category

Table 2 Evaluation of Readability and Relevance [D1, D2 and D3 denotes the first, second and third distractors respectively; 3-point scale: average score between 0–3]

Evaluator	Readability			Relevance		
	D1	D2	D3	D1	D2	D3
Evaluator 1	189	188	185	186	182	177
Evaluator 2	187	182	182	181	180	176
Evaluator 3	189	186	185	189	179	179
Evaluator 4	185	180	177	180	174	170
Evaluator 5	192	185	186	188	184	185
Average value	188.4	184.2	183	184.8	179.8	177.4
All Distractors Average (%)	92.6			90.33		
Score (3-point scale)	2.88			2.71		

identification module sometimes fails to resolve the conflict between location and organization, trophy and organization, ground and city, cricketer and cricket officials, various forms of a particular name etc. It also fails to detect the nicknames of the players and a few entities. The system is unable to generate relevant distractors in such cases.

For evaluation of the closeness, we employ 15 evaluators: five as Group1 and rest as Group2 evaluator. Group1 evaluators put the closeness score of an MCQ as an integer between 0-3. The score is 3 if the evaluator senses all the distractors as good; 0 if none of those is good; 1 and 2 consequently. Average of the scores given by individual evaluators is taken as the Group1 score. The score is 2.38 (see in Table 3).

The score of the Group2 evaluators depends on the size of the reference sets. We compute the average size of the reference set 1 as 5.8. Therefore, if a system generated distractor is one of these 5.8 manually generated distractors then the score is 1. In our experiments, we observed that the average score achieved by the distractors are 1.58 when we use the reference set 1 only. This value is quite promising. Next, we consider reference set 2 for the distractors that do not occur in set 1. Then the value increases to 1.94. And finally, when the rest of the distractors (neither occurs in set 1 nor in set 2) are assessed by the category 1 evaluators, the score becomes 2.21.

The average of the Group 1 and Group 2 scores are taken as the final closeness value of the distractors. The score becomes 2.3. The value indicates, out of the three systems generated distractors, on average 2.3 of them are close. The value is greater than 2; it implies that the system is able to generate at least two close distractors. In reality, if an MCQ contains three close choices (one key and two distractors), then also it can be considered as a good MCQ. The experimental results demonstrate the effectiveness of the distractor generation technique.

6.4 Comparison with related techniques

Now we compare the current system with related techniques and systems presented in the literature. In the related work section (Section 2) we mentioned several approaches for distractor generation. Majority of those approaches are not applicable in named entity distractor generation, for example, parts-of-speech similarity, similar

Table 3 Evaluation of Closeness: Group1 contains 5 evaluators and Group2 contains 10 evaluators

	Group 1 Score	Group 2 Score		
	Average of 5 evaluators	Reference set 1 only	+ Reference set 2	+ Category 1 evaluator scoring
Number of entities in corresponding set	600	1160	823	212
Number of Matching	–	316	388	442
Score (individual)	2.38	1.58	1.94	2.21
Average Score (3-point scale)	2.3			

words using WordNet or Ontology, phonetic similarity, pattern matching etc. Some of the approaches use a specialized setting or environment. For the comparison, we have implemented a few suitable approaches in the current setting and compared the proposed approach with those. These approaches are, frequency based (inspired from Brown et al., 2005), co-occurrence vector (like Mitkov et al., 2009), collocation similarity (inspired from Lee and Seneff, 2007 and Agarwal and Mannem, 2011) and system by Patra and Saha (2017). For the implementation, cricket related texts were collected from Wikipedia and other websites. For comparison, we used only the relevance and closeness value. We do not consider readability in this comparison; because, if a named entity is replaced by an appropriate noun, pronoun or a generic placeholder word, then also the readability of the sentence retains. In Table 4 we have presented the result of the comparison.

As expected, it is found that the frequency, general domain NER, co-occurrence or collocation-based approaches are not capable of generating good named entity distractors. When the frequency is used alone for distractor generation, then it rarely generates a close distractor. When a general domain NER system is incorporated then the score increased substantially. Collocation performs best among all these generic approaches. The proposed system generates better distractors than the system developed by Patra and Saha (2017). This is primarily because of the incorporation of the semantic similarity.

When we compare the proposed system with other possible techniques, we observe that our technique is better in terms of both the closeness and relevance. Closeness is an important measure for distractor generation. The closeness value should be high to use the MCQ in a real examination and it should be at least two. Closeness value two indicates, two (out of the three) distractors are good. Three close or confusing choices make an MCQ usable in real examination. We found that the distractors generated by Patra and Saha (2017) get a closeness value of 2.07. However, our hybrid model performs better. It reduces the run-time too by a large factor, as the similarity scores are pre-computed and stored in the hierarchical structure. So, the MCQs generated by the proposed system are usable but other simple techniques (shown in Table 4) are unable to generate good and usable MCQs.

Next, we perform error analysis by comparing the relevance and readability score with the closeness score. We observe that there are certain MCQs that get high relevance score but low closeness value. We found that the NE class has been detected correctly there but still, the distractors are not correct. We analyze a few such MCQs.

Table 4 Comparison with related systems: for NE distractor generation in Cricket domain [ideal score is 3, that indicates all three distractors meet the criteria]

System or Approach	Relevance	Closeness
Frequency based	0.35	0.11
NER with Frequency based	1.62	0.8
Co-occurrence based	1.45	0.92
Collocation based	2.04	1.13
Patra and Saha (2017)	2.45	2.07
The Proposed System	2.71	2.3

For instance, "Last over of the 1992 Hero cup semi-finals was bowled by – ". The key of the MCQ is Sachin Tendulkar and who is basically a batsman. An ideal distractors set should contain the top bowlers or all-rounder of that time; including, J Srinath, Anil Kumble, Kapil Dev, and Manoj Prabhakar. However, the system generated distractors contain Azharuddin and Ajay Jadeja, who received higher statistical similarity. Again in the MCQ "Which Indian players registered first hat-trick in World Cup matches?", the key is 'Chetan Sharma'. The reference set created based on evaluators' suggestions contains the bowlers who took highest wickets in ODIs or have good records in World Cup matches (A Kumble, J Srinath, Kapil Dev, and Z Khan). However, the system generated distractor set does not contain any of those entities. When the Category 1 evaluators were asked to assess the system generated distractors, they found two of them (Manoj Prabhakar, Roger Binny) are good. A similar outcome is attained in this MCQ too: "Who was first test captain of Indian cricket team?" From these observations, we conclude that a deeper analysis of the stem can further improve the quality of the distractors.

7 Conclusion

In this paper, we have presented a system for automatic named entity distractor generation. Named entity distractors are common in various domain and automatic generation of close NE distractors is a tricky task. To capture the closeness between the key and the possible distractors, we use web information. The closeness is captured through statistical and semantic information extracted using the Wikipedia and other trusted web sources. The proposed technique is applied in the sports domain. The system takes the question sentence and the correct answer as input and generates three distractors. The quality of the system generated distractors is evaluated using a set of human evaluators. During evaluation also we aim to assess the closeness and applicability of the system generated distractors in real MCQs. For that, we propose an evaluation strategy. The experimental results show that the system generates distractors are accurate.

There are a number of directions for future work. We feel that the proposed technique is generic and can be applied to other related domains like entertainment, history etc. But it requires experimental verification on those domains. Also, the accuracy of the named entity recognition module can be improved to make the system more accurate. The current setup of the system uses certain manual information during development; one can explore the possibilities to reduce the manual effort. One can extend our system by incorporating more sophisticated semantic score where the question sentence will also be analyzed and proper weight will be assigned to the relations embedded in the stem.

Funding Information This work is partially supported by

Compliance with Ethical Standards The authors declare that:

- This article does not contain any studies with human participants or animals performed by any of the authors.
- For this type of study formal consent is not required.

Conflict of interests The authors declare that they have no conflict of interest.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

Afzal, N., & Mitkov, R. (2014). Automatic generation of multiple choice questions using dependency-based semantic relations. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, *18*(7), 1269–1281.

Agarwal, M., & Mannem, P. (2011). Automatic gap-fill question generation from text books. In *Proceedings of the sixth workshop on innovative use of NLP for building educational applications* (pp. 56–64). Portland.

Araki, J., Rajagopal, D., Sankaranarayanan, S., Holm, S., Yamakawa, Y., Mitamura, T. (2016). Generating questions and multiple-choice answers using semantic analysis of texts. In *Proceedings of COLING 2016: Technical Papers* (pp. 1125–1136). Osaka.

Aldabe, I., & Maritxalar, M. (2010). Automatic distractor generation for domain specific texts. In *Proceedings of IceTAL 2010, LNAI 6233* (pp. 27–38).

Aldabe, I., Maritxalar, M., Mitkov, R. (2009). A Study on the automatic selection of candidate sentences and distractors. In *Proceedings of the 14th international conference on artificial intelligence in education (AIED2009)*. Brighton.

Bhatia, A.S., Kirti, M., Saha, S.K. (2013). Automatic generation of multiple choice questions using wikipedia. In *Proceedings of pattern recognition and machine intelligence (PReMI -13), LNCS 8251* (pp. 733–738).

Brown, J.C., Frishkoff, G.A., Eskenazi, M. (2005). Automatic question generation for vocabulary assessment. In *Proceedings of HLT/EMNLP* (pp. 819–826). Vancouver.

Coniam, D. (1997). A preliminary inquiry into using corpus word frequency data in the automatic generation of english language cloze tests. *CALICO Journal*, *14*(2), 15–33.

Correia, R., Baptista, J., Mamede, N., Trancoso, I., Eskenazi M. (2010). Automatic generation of cloze question distractors. In *Second language studies: acquisition, learning, education and technology*.

Fattoh, I.E. (2014). Automatic multiple choice question generation system for semantic attributes using string similarity measures. *Computer Engineering and Intelligent Systems*, *5*(8), 66–73.

Goto, T., Kojiri, T., Watanabe, T., Iwata, T., Yamada, T. (2010). Automatic generation system of multiple-choice cloze questions and its evaluation. *Knowledge Management & E-Learning: An International Journal*, *2*(3), 210–224.

Hoshino, A., & Nakagawa, H. (2007). Assisting cloze test making with a web application. In *Proceedings of society for information technology and teacher education international conference* (pp. 2807–2814). Chesapeake.

Karamanis, N., Ha, L.A., Mitkov, R. (2006). Generating multiple-choice test items from medical text: A pilot study. In *Proceedings of the fourth international natural language generation conference* (pp. 111–113). Sydney.

Kumar, G., Banchs, R.E., D?Haro L.D (2015). Automatic fill-the-blank question generator for student self-assessment. In *Frontiers in education conference (FIE)*, IEEE.

Lee, J., & Seneff, S. (2007). Automatic generation of cloze items for prepositions. In *Proceedings INTERSPEECH* (pp. 2173–2176). Antwerp.

Lin, Y.C., Sung, L.C., Chen, M.C. (2007). An automatic multiple-choice question generation scheme for english adjective understanding. In *Workshop on modeling, management and generation of problems/questions in elearning, ICCE 2007* (pp. 137–142). Hiroshima.

Liu, C.L., Wang, C.H., Gao, Z.M., Huang, SM. (2005). Applications of lexical information for algorithmically composing multiple-choice cloze items. In *Proceedings of the second workshop on building educational applications using NLP* (pp. 1–8). Ann Arbor.

Majumdar, M., & Saha, S.K. (2014a). Automatic selection of informative sentences: The sentences that can generate multiple choice questions. *Knowledge Management & E-Learning: An International Journal*, *6*(4), 377–391.

Majumdar, M., & Saha, S.K. (2014b). A system for generating multiple choice questions: With a novel approach for sentence selection. In *Proceedings of the 2nd ACL workshop on natural language processing techniques for educational applications (NLP-TEA)* (pp. 64–72). Beijing.

Mitkov, R., & Ha, L.A. (2003). Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT/NAACL 2003 workshop on building educational applications using NLP* (pp. 17–22). Edmonton.

Mitkov, R., Ha, L.A., Varga, A., Rello, L. (2009). Semantic similarity of distractors in multiple-choice tests: extrinsic evaluation. In *Proceedings of the EACL 2009 workshop on GEMS: GEometical models of natural language semantics* (pp. 49–56).

Papasalouros, A., Kanaris, K., Kotis, K. (2008). Automatic generation of multiple-choice questions from domain ontologies. IADIS e-Learning.

Patra, R., & Saha, S.K. (2017). Automatic generation of named entity distractors of multiple choice questions using web information. In *International conference on computing analytics and networking (ICCAN 2017)* (pp. 511–518).

Pino, J., Heilman, M., Eskenazi, M. (2008). A selection strategy to improve cloze question quality. In *Proceedings of the workshop on intelligent tutoring systems for ill-defined domains, in 9th international conference on intelligent tutoring systems* (pp. 22–32).

Rusu, D., Dali, L., Fortuna, B., Grobelnik, M., Mladeni, D. (2007). Triplet extraction from sentences. In *Proceedings of the 10th international multiconference information society 2007* (vol. A, pp. 218–222).