Open-domain Concept Distractor Generation for Multiple Choice Questions

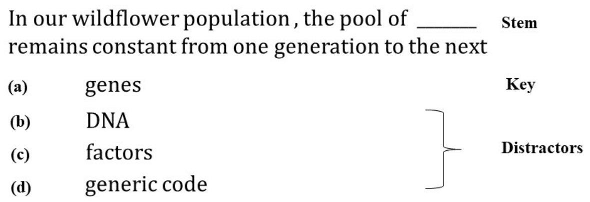
Anonymous EMNLP-IJCNLP submission

Abstract

Distractor generation aims to generate distractors for multiple choice questions (MCQ). To address the domain-specific and context-independent issues of existing methods, we propose a novel open-domain distractor generation framework that employs a candidate set generator to construct a candidate set for each question from Probase with contextual fit and a feature-based ranker to select candidates that are highly plausible and reliable. Experimental results on our newly-constructed dataset show that our framework yields distractors that are significantly more plausible and reliable than those generated by baseline methods. This dataset can also be used as a benchmark for word distractor generation.

1 Introduction

A MCQ consists of three elements: (i) stem - the question sentence; (ii) key - the correct answer; and (iii) distractors - alternative answers used to distract students from the correct answer. Figure 1 shows an example where the stem is a sentence with a blank to be filled in. Alternative (a) is the key, while alternatives (b) through (d) are distractors. Automatic multiple choice question (MCQ) generation has shown its potential in tasks such as English proficiency tests (Sumita et al., 2005), vocabulary exams (Susanti et al., 2018) and knowledge assessments (Guo et al., 2016).



In our wildflower population, the pool of ______ remains constant from one generation to the next — Stem

(a) genes — Key

(b) DNA

(c) factors

(d) generic code

— Distractors

Figure 1: Multiple-choice Item

Distractor generation (DG), which aims to generate alternatives for the given question sentence and correct answer, is a critical part of automatic MCQ generation. Alternatives of MCQs are usually concepts, so we only focus on these distractors in this paper. Generating distractors is a difficult task when creating a MCQ: the quality of a MCQ relies heavily on the quality of these distractors (Rodriguez, 2005; Haladyna et al., 2002). Good distractors ensure that the outcome of MCQ tests provides more credible and objective picture of the knowledge of the testees involved. e.g. Testees have to use expert knowledge in biology to distinguish that *DNA* in Figure 1 is incorrect despite it is similar to *genes*, and therefore *DNA* is a good distractor. On the other hand, even beginners can easily identify that *factors* is incorrectness since it is not semantically similar to *genes* at all, making it a bad distractor. *generic code* is also an correct answer, and therefore is an invalid distractor.

Literature in pedagogy generally recommends the following criteria for designing distractors: distractors should belong to the same word class and same difficult, have approximately the same length (Haladyna et al., 2002), as the target word; they should also be syntactically and semantically homogenous (Pho et al., 2014), but are not right answers. In sum, distractors must be sufficiently plausible and reliable. Specifically, by plausible, we mean distractors are semantically related to the key and consistent with the context given by stem; by reliable, we mean distractors are not correct answers to the question.

Recently, Liang et al. (2018) integrate previously proposed similarity measures and develop both feature-based ranking models and generative adversarial nets (Liang et al., 2017) that learn to select distractors. They achieve the state-of-the-art performance with feature-based ranking models. Therefore, in this paper, we focus on feature engineering to improve feature-based rankers.

Despite Liang et al. (2018)'s success, they select distractors from a rather small vocabulary containing instances from SciQ (Welbl et al., 2017) and MCQL (Liang et al., 2018) datasets, which is not generalizable. Although there exist other candidate sources including ontology (Stasaski and Hearst, 2017), manually annotated corpora (Zesch and Melamud, 2014; Sakaguchi et al., 2013), and thesaurus (Sakaguchi et al., 2013), they are all domain-specific and often limited in their coverage. To address this issue, some researchers (Jiang and Lee, 2017) propose to use words extracted from the Wiki corpus as distractor candidates. However, their method does not take the question context, an indispensable factor, into consideration. Whereas in this paper, our goal is to propose an open-domain context-dependent concept distractor generation framework.

Moreover, another issue is that many previous work only focus on how to select plausible distractors, and adopt no or weak reliability checking, which is a critical part to ensure there is only one correct answer in each item. Although Sumita et al. (2005) conduct reliable checking based on retrieval from the Web, their hypothesis that an instance is a correct answer if it appears in search results of the stem is too strict. For the example in Figure 1, *DNA* is rejected by their reliable checking filter since it often appears together with *genes* in web pages, but it is a very good distractor.

To overcome these shortcomings, we propose an open-domain, data-driven framework for generating word distractors that are both plausible and reliable with contextual fit. Based on contextual information, We construct a small-size candidate set from a large-scale general-purpose knowledge source, Probase, which is constructed from billions of web pages. Since this candidate source is generalizable and diverse enough, our method is adoptable in different domains. While generating distractors, we are actually making a trade-off between plausibility and reliability, so we can not consider than them individually. Instead of conducting candidate filtering (Jiang and Lee, 2017), we integrate new reliable checking features with other plausibility measures into distractor selector. With these methods, our framework generates *antigens, DNA, viruses* for the example in Figure 1 whose quality is comparable with *DNA, eggs, genomes* designed by human experts.

Prior methods are evaluated on different question sets collected from textbooks (Agarwal and Mannem, 2011), Wikipedia (Jiang and Lee, 2017), ESL corpora (Sakaguchi et al., 2013). To the best of our knowledge, there is no open-source benchmark to evaluate word distractors, so we construct a new dataset [1] covering science, trivia, vocabulary and common sense questions. This dataset can be used as a benchmark for further research exploration in concept distractor generation.

We devise evaluation metrics for DG to conduct quantitatively evaluation of the top generated distractors on our self-created dataset. Results show that our open-domain framework achieves significant and consistent improvement as compared to the previously proposed domain-specific word2vec-based method and thesaurus-based method. In particular, we demonstrate that distractors generated by our framework can achieve comparable plausibility and reliability to those designed by human experts.

In summary, the key contributions are four-fold:

- we create and open-source a cross-domain gap-fill MCQ dataset that span across domains, including science, vocabulary, common sense and trivia question tests, for future study on word distractor generation (Section 3.1);
- we propose a novel open-domain context-dependent candidate set generation method (Section 2.2).
- we propose new reliable checking features to select reliable distractors (Section 2.3).
- our proposed method achieves the state-of-the-art results on the dataset we created.

2 Automatic Word Distractor Generation

In this section, we introduce our model in detail. Our goal is to generate distractors that are very plausible but are not correct answers. As illustrated in Figure 2, we achieve this by combining: (1) an efficient technique to retrieve entities similar to the key under the given context from a large-scale general-purpose knowledge base. with (2) a learning-to-rank model that predict a score for each candidate distractor. The overall architecture in this paper differs from previous works in that they mainly focus on the ranking part.

[1] `https://anonymous_for_blind_review`

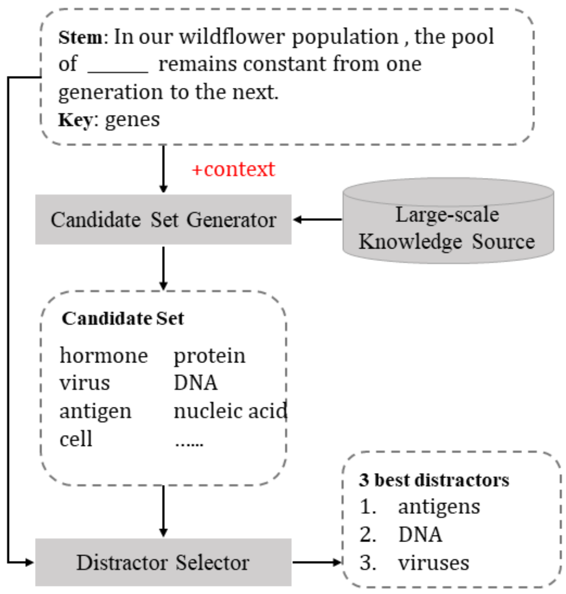


Figure 2: An overview of our framework.

2.1 Problem Statement

Formally, given a stem q and a key a, the problem is to generate n most appropriate distractors $D = \{d_1, d_2, \cdots, d_n\}$. We first retrieve m candidates from a large-scale corpus which are defined as $D = \{(d_1, p_1), (d_2, p_2), \cdots, (d_m, p_m)\}$ where d_i is the i-th retrieved candidate and p_i is the corresponding probability obtained from knowledge base. Then our ranking model measures the score of each candidate d_i given stem q and key a and outputs a n-best list $\{(d_1, s_1), (d_2, s_2), \cdots, (d_n, s_n)\}$ where s_i represents the ranking score.

2.2 Candidate Set Generator (CSG)

We propose a concept-dependent candidate set generation method that retrieves candidates from a general large-scale probabilistic knowledge base, Probase. Stems cannot be used directly to facilitate the candidate retrieval process, so the candidate set generator first extracts latent context z from the stem q and the key a. It further retrieves a candidate set D from knowledge base efficiently based on z, and measures the probability distribution $P(d_i|z)$ over all retrieved candidates.

We use Probase (Wu et al., 2012) due to its large coverage. It contains more than millions of concept-instance relationships automatically extracted using syntactic patterns from billions of Web pages, more than any other existing knowledge bases. It also specifies the probability of each instance belonging to the concept, $P(w|c)$, as well as the probabilities of the reverse direction $P(c|w)$. Its hierarchical concept structure allows us to conduct context-dependent conceptualization (CDC) (Kim et al., 2013) on the key k, and find other entities belonging to the same concept-level in this hierarchical concept graph. The topic distribution of the sentence formed by the stem and key serves as the context information for conceptualizing each instance word. With it, we estimate the concept distribution for the key by computing the probability of each concept based on the topic distribution of the sentence. The probability of concept c given instance w with its context by considering the topic distribution topics is calculated as follows:

$$p(c|w, z) \propto p(c|w)\Sigma_k \pi_{wk} \phi_{ck} \quad (1)$$

$$\phi_{ck} = \frac{C_{ck} + \beta}{\Sigma_w C_{wk} + |W|\beta} \quad (2)$$

where c is the concept, $\pi_{wk} = p(z_w = k)$ is the inferred topic distribution, $p(c|w)$ and ϕ_{ck} are defined in Probase.

After that, by following the descending chain of is-a relation and collecting hyponyms of these concepts, we obtain an entity distribution $p(d|w, z) \propto \Sigma_c p(d|c)p(c|w, z)$. Top m entities with greatest probabilities form a candidate set $A = \{(d_1, p_1), (d_2, p_2), \cdots, (d_m, p_m)\}$.

2.3 Distractor Selector (DS)

The previously generated candidate set is still relatively large and noisy. The distractor selector is used to output a n-best distractor list for each specific item and corresponding candidate set. We explore how to effectively re-rank extracted candidates using learning-to-rank models based on features covering both plausibility and reliability. Given a tuple (q; a; d) where q is a sentence with a blank (blanked sentence), a is the correct choice for the blank and d is a candidate distractor, the distractor selector first transform it into a feature vector:

- *Probase Score* $p(d|a, z)$ from candidate set generator.

- *Emb Similarity* GloVe embedding similarity (Pennington et al., 2014) and Wiki embedding similarity between q and d and the similarity between a and d. The Wiki embedding is trained using word2vec (Mikolov et al.,

2013) on Wikipedia data. We use the average word embedding as the sentence embedding. Embeddings have been demonstrated to be effective for finding semantically similar distractors (Guo et al., 2016).

- *Shape Similarity* Edit distance, token length difference and longest common suffix between a and d.

- *POS Similarity* Jaccard similarity between a and d's POS tags.

- *LM Score* Conditional Probability of d given q, $p(S) \approx \Pi_{i=1}^{k} p(w_i|w_1, \cdots, w_{i-1}, w_{i+1}, \cdots, w_k)$, predicted by language model. This is inspired by the previously proposed reliability checking approach of considering collocations involving the target word (Jiang and Lee, 2017; Pino et al., 2008; Smith et al., 2010).

- *Web-search Score* The correctness score of triples extracted from full sentence formed by q and d, obtained by comparing to web search results.

Probase Score, Emb Similarity, Shape Similarity and POS Similarity are used to make sure selected distractors are plausible enough. As far as reliability is concerned, previous works follow two extreme approaches. They either ignore whether d is a correct answer or strictly remove candidates satisfying a certain condition through dependency analysis or N-gram match (Jiang and Lee, 2017). Here we take a different approach: incorporate correctness measurements in the ranking model as a feature.

More specifically, LM Score measures correctness statistically. In addition, we propose a web-based approach to verify the correctness of the sentence restored by each candidate. The Web includes all manners of language data in vast quantities. So we dare to assume that an (argument1, relation phrase, argument2) triple is more likely to be correct if it appears more frequently on the Web. We also assume that the correctness of triple can be used to measure the correctness of a sentence if the triple forms the main structure. We retrieve relevant information from the web by passing a sequence of words to a search engine automatically. After that, we use ReVerb (Fader et al., 2011) to extract (argument1, relation phrase, argument2) triples involving d from the sentence formed by q and d, $\{t_{11}, t_{12}, \cdots, t_{1n}\}$, as well as snippets and titles returned by a search engine, $\{t_{21}, t_{22}, \cdots, t_{2m}\}$. After that, we calculate embedding similarities between triples and keep the maximal score, $T(q, d)$, that represents the correctness of triple extracted from a sentence:

$$T(q,d) = max_{\substack{i \in \{1,2,\cdots,n\} \\ j \in \{1,2,\cdots,m\}}} EmbSim(t_{1i}, t_{2j})$$

where $EmbSim(t_{1i}, t_{2j})$ represents the embedding similarity between t_{1i} and t_{2j}. If $T(q, d)$ is small, then the sentence restored with the distractor d is unlikely, thus the d should be a good distracter.

3 Experiments

In this section, we first present dataset and hyperparameters adopted in our experiments. Then we present our evaluation of our proposed methods and several common baseline methods on this dataset.

3.1 Dataset and Evaluation Metrics

Our proposed MCQ dataset covers multiple domains including science, vocabulary, common sense and trivia questions [2]. It is compiled from many existing MCQ data sources including SciQ (Welbl et al., 2017), MCQL (Liang et al., 2018), AI2 Science Questions [3], AI2 4th Grade Science Exams [4] as well as trivia, and vocabulary MCQs crawled from websites. We filter out MCQs whose distractors are not short phrases, which results in 2,587 items in total. Table 1 summarizes the statistics of our dataset.

Domain	Total	Science	Vocab.	C.Sense	Trivia
MCQs	2587	458	956	713	460
Dis.	3.13	3.00	3.99	3.48	2.99

Table 1: Dataset Statistics (number of MCQs and average number of distractors)

We convert questions to gap-fill form by constructing Penn Treebank style trees from the parsing result with the help of the openly available Stanford Parser (Klein and Manning, 2003), and

[2]This dataset can be downloaded from http://anonymized.for.blind.review.

[3]http://data.allenai.org/ai2-science-questions/

[4]http://data.allenai.org/ai2-4th-grade-science-questions-training-set/

adjusting node order according to the identified question type. For experiments, we randomly divide the dataset into train/valid/test with an approximate ratio of 10:1:1.

We use the following metrics to evaluate distractor generation methods:

Human Annotation To examine the plausibility and reliability of generated distractors, we ask five human judges who are native English speakers to annotate the generated distractors without revealing the key. We totally use 50 items. Each item contains many alternatives that include 3 distractors generated by each method and a randomly picked distractor from the ground truth designed by human experts. For each alternative in the item, the judges decided whether it is correct or incorrect; they may identify zero, one or multiple correct answers. For an incorrect alternative, they further assess its plausibility as a distractor on a three-point scale: "Obviously Wrong" (0 point), "Somewhat Plausible" (1 point), or "Plausible" (2 points).

Ranking Measures We consider the ground truth distractors designed by human experts as good distractors. To some extent, whether the ground truth distractors come out on the top of the result list demonstrate the effectiveness of the distractor generator. Hence, we report top precision (P@1, P@3) to show how many automatically generated distractors hit the ground truth distractors, as well as mean reciprocal rank (MRR) and normalized discounted cumulative gain (NDCG) to show positions of ground truth distractors in the ranking list if they exist.

Semantic Similarity Sometimes, the generated distractors do not exactly match the ground truth, but are semantically very close and these should be deemed good distractors as well. Word2vec (Mikolov et al., 2013) model trained on Wikipedia dump is utilized to measure semantic similarity between automatically generated distractor and ground truth distractor. Word2vec typically gives a higher similarity score if the two words are often used in similar contexts.

3.2 Implementation Details

In order to use Probase in the candidate set generator, we conduct CDC on the key. We first train a probabilistic topic model, latent dirichlet allocation (LDA) that discovers the latent topic distribution, with a Wikipedia corpus containing 3 million documents. Following the best practice (Kim et al., 2013), we set the number of topics to be 100.

We define Recall@K as the percentage of ground truth distractors contained in the top K candidates generated by the CSG. Table 2 shows the Recall@K for CSG on our dataset. We empirically make the candidate set generator retrieve 500 candidates for each item to get a recall of 50%. Due to the nature of our problem, we use Adaptive Boosting algorithm (Freund and Schapire, 1997) to train the distractor ranker, and we set the maximum number of estimators to 50. We adopt cascaded-learning to train a two-layer AdaBoost ranker where LM score and Web-search score are only included in the second layer because they are computational expensive.

K	50	100	500	1000
Recall@K	21.4%	37.25%	52.94%	59.48%

Table 2: Recall@K for CSG.

3.3 Overall Results

In this part, we will show the performance of different methods on our newly-constructed dataset and provide some analysis. We compare our framework with the following baselines:

Thesaurus-based Method (TM) extracts distractor candidates from synonyms of the key in WordNet and give up those that appear in the documents retrieved from the Web search results returned by the search engine (Sumita et al., 2005).

Word2vec Prediction (WP) predicts center words using word2vec (Mikolov et al., 2013) model trained on English Wikipedia dump, and selects distractors based on the word2vec similarity between the candidate and the key. With the Continuous Bag of Words (CBOW) model, we get the probability distribution of the center word given context words, $p(d|W)$ where W represents bag of words that appear in the sentence. Top m entities with greatest similarity scores form a candidate set $D = \{(d_1, p_1), (d_2, p_2), \cdots, (d_m, p_m)\}$, where p_i measures the similarity score of each word in vocabulary with the context.

We also compare our method with its naive version in which candidates are ranked according to the Probase Score (see Section 2.3). We name our method as **CSG+DS** and its naive version **CSG**. Another baseline is to integrate Word2vec Prediction with our distractor generator, named **WP+DS**.

The human annotation results are shown in Table 3. From the results, we observe that the CSG+DS method performs best in terms of plausibility and reliability.

Reliability

Our method shows a significant improvement in reliability compared with baselines. TM, with 83.96% reliability, is more prone to yielding invalid distractors that are correct answers. This is not unexpected since it tries to find distractors from synonyms of the key, which are likely to have the exact same meaning as the key. e.g. TM gives *categorizing* as a distractor for the stem "*a scientist is using ____ when sorting rocks into two groups based on physical appearance*" and the key *classifying*. However, our method retrieves candidates belonging to the same concept-level, thus generate less invalid distractors. We also find that, with the same candidate generator, adding distractor selector helps improve the reliability rate, which verifies the validity of our reliability checking features.

Plausibility

Distractors generated by the CSG+DS method have very competitive quality, scoring on average 1.12, even higher than the average score of the ground truth designed by human experts (Human). They are also less likely to be rated "Obviously Wrong." The reason is that, different from other methods, CSG+DS method can make full use of the rich information in the stem to generate distractors belonging to the same concept-level as the key with contextual fit, which is also challenging for human experts. For example, for the stem "____ *functions primarily to defend the body against disease.*" and the key "*immune*", our CSG+DS method generates "*reproductive, respiratory, digestive*" which belong to the same concept *functions* as the key. On the other hand, the ground truth distractor *nervous* is not similar to the key enough. TM and WP achieve low plausibility since it does not consider latent context, leading to many obviously wrong distractors. Although WP does make used of context words, it treats the key and other context words similarly and may select distractors similar to a context word instead of the key.

Table 4 shows the performance of our proposed CSG+DS on different domains. It verifies our claim that adopting a large-scale knowledge base as the candidate source is more generalizable. The reason for its worse performance in vocabulary and trivia is that the ground truth distractors are mostly verbs and cannot be found in Probase. Our method gives high-quality distractors when applied on datasets whose keys are concepts, regardless of the specific domain.

3.4 Effect of Using Different Candidate Set Generators

To demonstrate the advantage of Probase, we compare it with other candidate sources such as Wiki Corpus and WordNet (Miller, 1998) by utilizing them together with our distractor selector separately.

WordNet Following Sumita et al. (2005), we extracts distractor candidates from synonyms of the key in WordNet.

Wiki Corpus We train a word2vec (Mikolov et al., 2013) model on Wiki Corpus and use treat center words predicted by it as candidates.

Table 5 shows that Probase candidate set generator effectively retrieves candidates with highest ranking measures and semantic similarity. That is to say, the Probase CSG is more likely to generate ground truth distractors on the top of the ranking list, and it matches the criterion of keeping distractor semantically homogeneous better.

3.5 Effect of Using Different Distractor Selectors

To illustrate the effectiveness of our feature-based ranker, we compare it with several baseline methods by running with the Probase candidate set generator separately:

-Reliability Checking (-RC) is a naive version of our distractor selector. It excludes LM score and Web-search score from the distractor selector.

Word Similarity (WS) ranks candidate distractors according to their embedding similarity scores with the target word. The scores are obtained by a word2vec model trained on Wiki corpus (Jiang and Lee, 2017).

Candidate Filtering (CF) excludes all the candidates that appear in the web search result of the stem (Sumita et al., 2005).

As shown in Table 6, our method and its naive version -RC achieves much better perfomance than the two methods. Reasonably, -RC performs better in terms of NDCG and semantic similarity as it excludes reliability checking features that, to some extent, help "filter" correct answers from the

Method	Reliability	Plausibility	Plausible(%)	P@1(%)	P@3(%)	MRR(%)	NDCG(%)	Semantic Similarity
TM	83.96%	0.55	38.43%	0.81	1.04	3.48	3.68	0.124
WP	93.73%	0.28	22.44%	0.61	0.45	0.61	1.83	0.120
WP + DS	94.69%	0.65	23.92%	3.06	1.97	5.59	6.60	0.121
CSG	90.43%	0.80	53.79%	7.49	5.89	7.50	**13.47**	0.194
CSG + DS (Our)	**96.04%**	**1.12**	**73.26%**	**7.76**	**5.91**	**12.85**	13.41	**0.214**
Human	**100%**	**1.04**	**72.65%**	-	-	-	-	-

Table 3: Plausibility and reliability evaluation results of the various distractor generation methods in human evaluation. Average plausibility scores is calculated from a 3-point scale 3.2.

Domain	P@1(%)	P@3(%)	MRR(%)	NDCG(%)	Sem Sim
Science	10.71	6.29	17.27	16.05	0.232
Common Sense	4.58	4.13	10.39	8.97	0.121
Vocabulary	3.08	1.44	4.57	5.70	0.082
Trivia	6.11	4.85	12.87	11.44	0.018
Total	7.76	5.91	12.85	13.41	0.214

Table 4: Results of CSG+DS(Our) method on different domains.

final n-bast distractor list. e.g. For the stem "*is considered a good source of calcium*" and the key *milk*, -RC generates *kefir*, a correct answer, as the best distractor. On the other hand, *kefir* is not contained in the 10-best distractor list generated by our method, but *kefir* have higher semantic similarity than other candidates.

3.6 Case Study & Discussion

Our CSG + DS framework successfully generate distractors belonging to the same concept-level as the key but are not synonyms of the key. e.g. As shown in Table 7, TM outputs "sequences", "factors" and "combinations", vague concepts that the key belongs to, while our Probase CSG + DS model outputs "antigens", "DNAs" and "viruses", all of which belongs to *biological entities*, a fine-grained concept.

One thing we notice in our experiments is that Wiki CSG may generate many candidates similar to the context words rather than the key due to its prediction nature. However, Probase CSG focus on generating words similar to the key restricted by the particular context information. So we believe that, compared with other existing general-purpose knowledge sources, Probase is a better resource for concept distractor generation.

In contrast to other distractor selectors including WS and Probase Probability (CSG alone), our framework can generate more plausible distractors such as "DNA" in this example with the help of feature-based ranking model.

4 Related Work

The process of finding good distractors involves two steps: candidate set generation which narrows down distractor candidates to a small set, and distractor selection that controls the difficulty of the items and ensures that the distractors are *not* the correct answer to the stem (reliability checking).

4.1 Candidate Generation

Sometimes the set of possible distractors is known in advance, so candidate set generation is omitted. e.g. manually annotated errors such as Lang-8 platform [5] (Sakaguchi et al., 2013), the set of English prepositions in preposition exercises (Lee and Seneff, 2007). Nevertheless, in the common scenarios, only the key and the stem is known beforehand and the set of candidates needs to be automatically generated. A solution used in previous work to construct distractor candidate sets from thesauri (Sumita et al., 2005; Smith et al., 2010) or taxonomies (Hoshino and Nakagawa, 2007; Mitkov et al., 2009; Ding and Gu, 2010). These domain-specific candidate source are still not large or general enough, however, to support open-domain distractor generation.

4.2 Distractor Selection

Randomly selecting candidates is a valid strategy (Mostow and Jang, 2012), but it is only suitable for generating entry-level MCQs. Thus, more advanced approaches usually select distractors according to different metrics based on the key including embedding-based similarities (Guo

[5] `http://cl.naist.jp/nldata/lang-8/`

Source	P@1(%)	P@3(%)	MRR(%)	NDCG(%)	Semantic Similarity
Wiki Corpus	3.06	1.96	5.58	6.60	0.121
WordNet	5.63	4.41	10.25	1.83	0.204
Probase	**7.76**	**5.91**	**12.85**	**13.41**	**0.214**

Table 5: Effect of Different Candidate Set Generators

Method	P@1(%)	P@3(%)	MRR(%)	NDCG(%)	Semantic Similarity
-RC	7.49	5.88	7.49	**13.46**	**0.220**
WS	2.66	1.77	2.66	5.92	0.219
CF	0.23	0.25	0.23	1.42	0.155
Our	**7.76**	**5.91**	**12.85**	13.41	0.214

Table 6: Effect of Different Distractor Selector. -RC: Our method - reliability checking; WS: Probase CSG + Word Similarity (Jiang and Lee, 2017); IV: Probase CSG + Incorrectness Verification (Sumita et al., 2005)

TM	WP	WordNet CSG + DS	Wiki CSG + DS	Probase CSG	Probase CSG + DS	Probase CSG + WS
cistrons	hfrs	strings	hfr	proteins	**antigens**	proteins
codons	brodney	chains	pfr	cells	**DNAs**	enzymes
hierarchy	pfrs	stations	radiance	viruses	**viruses**	E. coli.
nexus	radiance	sequences	flux	general coliforms	**diseases**	nucleic acids
factors	pools	codons	pools	emzymes	**cells**	hormones

Table 7: Top 5 distractors generated by different methods for the stem "In our wildflower population, the pool of _____ remains constant from one generation to the next" and the key "genes"

et al., 2016), difficulty level (Hoshino and Nakagawa, 2007; Brown et al., 2005; Coniam, 2013; Shei, 2001), co-occurrence likelihood (Hill and Simha, 2016), WordNet-based metrics (Mitkov et al., 2003) and syntactic features (Agarwal and Mannem, 2011).

In addition to the key, some approaches also consider the semantic relatedness of distractors with the whole stem (Pino et al., 2008; Agarwal and Mannem, 2011; Mostow and Jang, 2012), i.e. they pick distractors that are from the same domain as the key. Some researchers (Liang et al., 2018; Sakaguchi et al., 2013; Liang et al., 2017) investigate how to apply learning-based ranking models to select distractors that resemble those in actual exam MCQs, and quantitatively evaluate the top generated distractors.

4.3 Reliability Checking

Selecting more challenging distractors usually means making them more similar to the key. This also increases the probability that a distractor might actually be a correct answer, a more sophisticated approach is required for checking the reliability of distractor candidates. In those cases where we have a limited list of potential target words and distractors, a supervised classifier can be trained to do this job (Lee and Seneff, 2007). However, it cannot be easily applied to open word classes like nouns or verbs, which are much larger and dynamic in nature.

Another way to perform reliability checking is by considering collocations involving the target word (Pino et al., 2008; Smith et al., 2010; Jiang and Lee, 2017). This approach is effective, but requires strong collocations statistics to discriminate between valid and invalid distractors and may not be applied to the sentence in Figure 1 which contains rare word combinations.

Also, a web search approach is applied by Sumita et al. (2005) to judge the reliability of a candidate. They discard words that can be found on the web search results of the stem with blank filled with the distractor. We note that the applicability of this approach is limited, as this may rule out many reliable candidates thus lower the plausibility of generated distractors.

5 Conclusion

In this paper, we present a novel framework for automatic concept distractor generation. The framework has two components: candidate set generator and distractor selector. It utilizes context information to generate distractors from Probase. Experiments on our newly compiled benchmark dataset show that our proposed framework outperforms the existing methods significantly on distractor plausibility and reliability.

References

Manish Agarwal and Prashanth Mannem. 2011. Automatic gap-fill question generation from text books. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 56–64. Association for Computational Linguistics.

Jonathan C Brown, Gwen A Frishkoff, and Maxine Eskenazi. 2005. Automatic question generation for vocabulary assessment. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 819–826. Association for Computational Linguistics.

David Coniam. 2013. A preliminary inquiry into using corpus word frequency data in the automatic generation of english language cloze tests. *Calico Journal*, 14(2-4):15–33.

Xiang-Min Ding and Hong-Bin Gu. 2010. Automatic generation technology of chinese multiple-choice items based on ontology. *Computer Engineering and Design*, 31(6):1397–1400.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1535–1545. Association for Computational Linguistics.

Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

Qi Guo, Chinmay Kulkarni, Aniket Kittur, Jeffrey P Bigham, and Emma Brunskill. 2016. Questimator: Generating knowledge assessments for arbitrary topics. In *IJCAI-16: Proceedings of the AAAI Twenty-Fifth International Joint Conference on Artificial Intelligence*.

Thomas M Haladyna, Steven M Downing, and Michael C Rodriguez. 2002. A review of multiple-choice item-writing guidelines for classroom assessment. *Applied measurement in education*, 15(3):309–333.

Jennifer Hill and Rahul Simha. 2016. Automatic generation of context-based fill-in-the-blank exercises using co-occurrence likelihoods and google n-grams. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 23–30.

Ayako Hoshino and Hiroshi Nakagawa. 2007. Assisting cloze test making with a web application. In *Society for Information Technology & Teacher Education International Conference*, pages 2807–2814. Association for the Advancement of Computing in Education (AACE).

Shu Jiang and John Lee. 2017. Distractor generation for chinese fill-in-the-blank items. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 143–148.

Dongwoo Kim, Haixun Wang, and Alice Oh. 2013. Context-dependent conceptualization. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

John Lee and Stephanie Seneff. 2007. Automatic generation of cloze items for prepositions. In *Eighth Annual Conference of the International Speech Communication Association*.

Chen Liang, Xiao Yang, Neisarg Dave, Drew Wham, Bart Pursel, and C Lee Giles. 2018. Distractor generation for multiple choice questions using learning to rank. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 284–290.

Chen Liang, Xiao Yang, Drew Wham, Bart Pursel, Rebecca Passonneaur, and C Lee Giles. 2017. Distractor generation with generative adversarial nets for automatically creating fill-in-the-blank questions. In *Proceedings of the Knowledge Capture Conference*, page 33. ACM.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

George Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Ruslan Mitkov, Le An Ha, Andrea Varga, and Luz Rello. 2009. Semantic similarity of distractors in multiple-choice tests: extrinsic evaluation. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 49–56. Association for Computational Linguistics.

Ruslan Mitkov et al. 2003. Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing*.

Jack Mostow and Hyeju Jang. 2012. Generating diagnostic multiple choice comprehension cloze questions. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 136–146. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Van-Minh Pho, Thibault André, Anne-Laure Ligozat, Brigitte Grau, Gabriel Illouz, Thomas François, et al. 2014. Multiple choice question corpus analysis for distractor characterization. In *LREC*, pages 4284–4291.

Juan Pino, Michael Heilman, and Maxine Eskenazi. 2008. A selection strategy to improve cloze question quality. In *Proceedings of the Workshop on Intelligent Tutoring Systems for Ill-Defined Domains. 9th International Conference on Intelligent Tutoring Systems, Montreal, Canada*, pages 22–32.

Michael C Rodriguez. 2005. Three options are optimal for multiple-choice items: A meta-analysis of 80 years of research. *Educational Measurement: Issues and Practice*, 24(2):3–13.

Keisuke Sakaguchi, Yuki Arase, and Mamoru Komachi. 2013. Discriminative approach to fill-in-the-blank quiz generation for language learners. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 238–242.

Chi-Chiang Shei. 2001. Followyou!: An automatic language lesson generation system. *Computer Assisted Language Learning*, 14(2):129–144.

Simon Smith, PVS Avinesh, and Adam Kilgarriff. 2010. Gap-fill tests for language learners: Corpus-driven item generation. In *Proceedings of ICON-2010: 8th International Conference on Natural Language Processing*, pages 1–6. Macmillan Publishers.

Katherine Stasaski and Marti A Hearst. 2017. Multiple choice question generation utilizing an ontology. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 303–312.

Eiichiro Sumita, Fumiaki Sugaya, and Seiichi Yamamoto. 2005. Measuring non-native speakers' proficiency of english by using a test with automatically-generated fill-in-the-blank questions. In *Proceedings of the second workshop on Building Educational Applications Using NLP*, pages 61–68. Association for Computational Linguistics.

Yuni Susanti, Takenobu Tokunaga, Hitoshi Nishikawa, and Hiroyuki Obari. 2018. Automatic distractor generation for multiple-choice english vocabulary questions. *Research and Practice in Technology Enhanced Learning*, 13(1):15.

Johannes Welbl, Nelson F Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*.

Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. 2012. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 481–492. ACM.

Torsten Zesch and Oren Melamud. 2014. Automatic generation of challenging distractors using context-sensitive inference rules. In *Proceedings of the Ninth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 143–148.