# Knowledge-Driven Distractor Generation for Cloze-Style Multiple Choice Questions

**Siyu Ren, Kenny Q. Zhu***
Shanghai Jiao Tong University
Shanghai, China
roy0702@sjtu.edu.cn, kzhu@cs.sjtu.edu.cn

**Abstract**

In this paper, we propose a novel configurable framework to automatically generate distractive choices for open-domain cloze-style multiple-choice questions. The framework incorporates a general-purpose knowledge base to effectively create a small distractor candidate set, and a feature-rich learning-to-rank model to select distractors that are both plausible and reliable. Experimental results on a new dataset across four domains show that our framework yields distractors outperforming previous methods both by automatic and human evaluation. The dataset can also be used as a benchmark for distractor generation research in the future.

## 1 Introduction

Cloze-style multiple choice question (MCQ) is a common form of exercise used to evaluate the proficiency of language learners, frequently showing up in homework and online testings. Figure 1 shows a cloze-style MCQ, which typically consists of: a question stem with a blank to be filled in, the correct answer and multiple wrong answers used to distract testees. Despite the high demand, manual crafting of such MCQs is highly time-consuming for educators, which calls for the automatic generation of as much practice material as possible from readily available plain texts so that formally usable quizzes can be generated after light-weight human calibration.

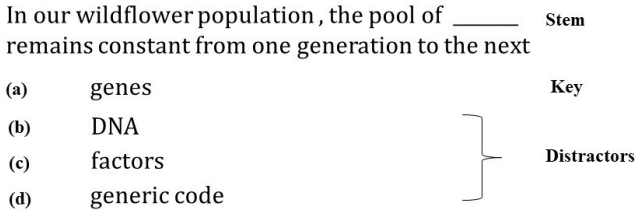


Figure 1: A cloze-style MCQ

Distractor generation, which aims to generate distractive alternatives (i.e., distractors) of the correct answer given the question stem, is a critical part of cloze-style MCQ construction. However, it is not only time-consuming but also non-trivial to produce appropriate distractors without rich experience in language education.

Literature in language pedagogy (Haladyna, Downing, and Rodriguez 2002; Pho et al. 2014) generally recommends two criteria for designing distractors: *plausibility* and *reliability*. By plausibility, it means distractors should be semantically related to the key and grammatically consistent with the context given by stem to adequately discriminate learners' proficiency. By reliability, it means the distractor, when filled into the blank of the stem, results in a logically incorrect or inconsistent statement.

Automatically generating distractors has been previously explored as part of cloze-style MCQ construction in a few studies. However, those methods generally assume prior knowledge of a specific domain (e.g., science) of the given question and then use corresponding domain-specific vocabulary as candidate distractor set, ranked by various unsupervised similarity heuristics (Sumita, Sugaya, and Yamamoto 2005; Kumar, Banchs, and D'Haro 2015; Jiang and Lee 2017; Ha and Yaneva 2018) or supervised machine learning model (Sakaguchi, Arase, and Komachi 2013; Welbl, Liu, and Gardner 2017; Liang et al. 2018). Since identifying the concrete domain of each question and preparing large-scale domain-specific vocabulary require substantial human labor, such corpus-based methods cannot be easily applied in real-world scenarios.

Another issue is that previous approaches mainly focus on selecting plausible distractors while rarely adopt reliability checks to ensure that the generated distractors are logically incorrect. Despite some attempts in early approaches (Sumita, Sugaya, and Yamamoto 2005; Jiang and Lee 2017), they both used it in the post-processing step to filter out candidate distractors rejected by diverse predefined filters such as syntactic feature (e.g., role in the dependency parse tree), which may exclude useful distractors like *DNA* in Figure 1.

In this paper, we propose a configurable distractor generation framework for English cloze-style MCQ in the open domain, whose design is motivated by the shortcomings identified above. It mainly consists of two components: (1) a

*Corresponding Author.

*context-dependent* candidate set generator, which constructs a small set of candidate distractors from a general-purpose knowledge base, based on contextual information formed by the stem and the key; (2) a learning-to-rank model that takes both reliability checking and plausibility measures into consideration. By incorporating structured, human-curated general-purpose knowledge base and conducting context-dependent conceptualization on the answer, we are able to effectively extract semantically-related candidate distractors without the need of domain-specific vocabulary. These candidate distractors are further re-ordered by a ranking model, trained with elaborately designed features to control the trade-off between plausibility and reliability.

Previous DG methods (Kumar, Banchs, and D'Haro 2015; Liang et al. 2017, 2018) are evaluated either with sole human annotation or on ad hoc datasets that are often narrow in domain. To the best of our knowledge, there is no open-source benchmark dataset for DG that is diverse enough to comprehensively evaluate the model performance. We compile a cross-domain cloze-style MCQ dataset covering science, trivia, vocabulary and common sense, which can be used as a benchmark for future research in DG. We further investigate various instantiations of the framework.

The contributions of this paper are three-folds:

- we compile and open-source a diverse and comprehensive benchmark dataset for training and evaluating distractor generation model (Section 3.1).
- we propose a configurable distractor generation framework for open-domain cloze-style MCQ, which requires no domain-specific vocabulary and jointly evaluates the plausibility and reliability of distractors (Section 2).
- we conduct comprehensive experiments to evaluate and analyze various instantiations of our framework and show that it consistently outperforms previous methods in both automatic ranking measures (about 2% F1 score) and human evaluation (Section 3.5).

## 2 The Framework

As illustrated in Figure 2, our framework includes two components: **C**andidate **S**et **G**enerator (**CSG**) and **D**istractor **S**elector (**DS**). The first component CSG extracts candidate distractors that are semantically similar to the key from a general-purpose knowledge base (KB). The second component DS, a generic feature-rich ranking model, then re-ranks those candidates according to more fine-grained assessment of grammatical consistency and reliability.

### 2.1 Task Formulation

Formally, given the stem $q$ and key $a$, the task of distractor generation is to generate $n$ most appropriate (i.e., meeting the requirement of plausibility and reliability) distractors $d_i$ in a set of distractor-score pairs $D = \{(d_i, s_i)\}$, $1 \leq i \leq n$, in descending order of $s_i$.

### 2.2 Candidate Set Generator (CSG)

The proposed CSG explicitly leverages the observation that distractors to an open-domain cloze-style MCQ are often words or short phrases living in a knowledge base (e.g., Probase (Wu et al. 2012), WordNet (Leacock and Chodorow 1998)) and stored as *nodes* in a way that they are connected with the key through a common *parent node* (which we refer to as *concept* later). Instead of enumerating all words in a huge domain-specific vocabulary in early approaches, such hierarchical structure in knowledge base allows us to extract candidate distractors by only considering a reasonably small number of concepts $C$ that are semantically related to the key, which can be efficiently identified using KB-specific interfaces.

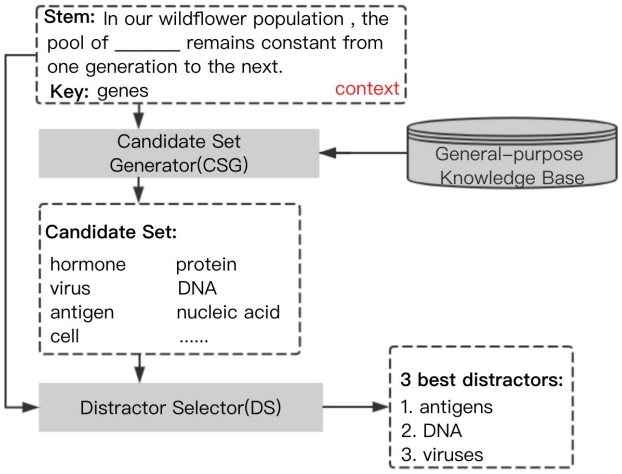


Figure 2: An overview of our framework.

Nevertheless, the specific meaning of the key varies given different stems. For example, given the sentence: "*These survivors managed to swim to the* ***bank***," where ***bank*** is the key, we would like to generate candidates like *bay* rather than the more commonly used financial-related terms.

Inspired by the idea of context-dependent conceptualization (Kim, Wang, and Oh 2013), we utilize a probabilistic topic model, LDA (Blei, Ng, and Jordan 2003), to discover the latent topic distribution of the context as well as the topic distribution of all concepts in the concept set $C$[1]. The posterior probability $p(c|a, q)$ of key $a$ belonging to concept $c$ conditioned on the stem $q$, is given by:

$$p(c|a,q) \propto p(c|a) \sum_{k=1}^{K} \pi_{a,q}^{(k)} \gamma_c^{(k)} \quad (1)$$

where $c$ is the concept, $\pi_{a,q}$ is the topic distribution of complete sentence formed by the stem and key, $\gamma_c$ denotes the topic distribution of concept $c$, $p(c|a)$ is the prior probability of $a$ belonging to $c$ corresponding to the specific choice of knowledge base, and $K$ is the total number of topics. Intuitively, concepts whose topic distribution resembles that of the complete sentence will be weighted higher than others. After obtaining the conditional probability $p(c|a, q)$ of all concepts in $C$, by following the descending chain of *is-A* relation and collecting hyponyms of these concepts as in

[1]We have also experimented using cosine similarity from BERT-based embeddings but observed longer inference time and similar performance. More details are described in Appendix A.

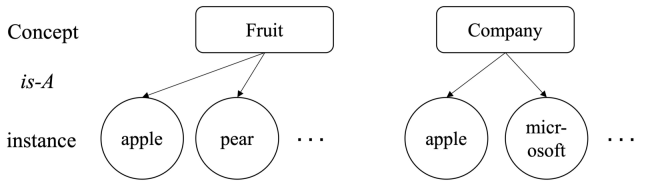


Figure 3: A snippet of Probase. Instances are connected to concepts under *is-A* relation.

Figure 3, we get a probability distribution over all entities subsumed by the concepts in $C$:

$$p_i = p(d_i|a,q) \propto \sum_{c \in C} p(d_i|c)p(c|a,q) \quad (2)$$

where the probability $p(d|c)$ is also known as typicality (Wu et al. 2012). The prior probability $p(c|a)$ and typicality $p(d|c)$ can be used off-the-shelf in some KBs (e.g., Probase) while for some other KBs (e.g., WordNet) it is not the case, which endows our framework with the flexibility to be combined with a broad class of KBs and to be customized with different ways of calculating these two probabilities.

Then we remove candidates that occur in the stem and finally the top $m$ candidates with largest probabilities form a candidate distractor set $D_0 = \{(d_1, p_1), (d_2, p_2), \cdots, (d_m, p_m)\}$.

## 2.3 Distractor Selector (DS)

Given the previously constructed candidate distractor set $D_0$, the final $n$-best distractors are generated in the following steps.

**Feature Extractor** Given a triplet $(q; a; d)$ where $q$ is the stem, $a$ is the key and $d$ is a candidate distractor, our DS first transforms it into a feature vector $f(q, a, d) \in \mathbb{R}^{33}$, in which the features are defined below:

- *Embedding Similarity:* Similarity between $q$ and $d$ and similarity between $a$ and $d$ calculated using cosine similarity between their CBOW embeddings, which is effective for finding semantically similar distractors (Guo et al. 2016). We use the average word embedding as the sentence embedding.
- *Contextual Embedding Similarity:* Cosine similarity between the ELMo (Peters et al. 2018) embedding of $a$ and $d$. This feature is complementary to *Embedding Similarity* since Word2Vec only capture static blended semantic of words, of which the significance is verified in our experiment.
- *Morphological Similarity:* Edit distance, token/character length difference, singular/plural consistency, absolute and relative length of $a$ and $d$'s longest common prefix/suffix/subsequence. These features measure the morphological similarity and are useful for cases such as abbreviation (e.g., DNA and RNA).
- *POS Similarity:* Jaccard similarity between the POS tags of $a$ and that of $d$. The intuition is that a good distractor should share similar linguistic properties as the answer.
- *Frequency:* Average unigram frequency of $a$ and $d$. Frequency has been previously utilized as a proxy for word's difficulty level (Coniam 1997). This feature aids model to select distractors with similar difficulty as $a$.
- *Compositional Similarity:* Jaccard similarity between token-level unigram set and bigram set of $a$ and $d$. This feature is motivated by the observation that distractors might share tokens with answer.
- *Web-search Score:* Details of this feature are described later in this section.

Features except *Web-search Score* are integrated to mainly evaluate the plausibility of $d$ in various aspects and granularities. *Web-search Score* is specifically introduced to assess the validity of the sentence restored by each candidate in order to further strengthen reliability. First, search results are retrieved from the web by passing the full sentence concatenated from $q$ and $d$ to the Bing search engine automatically. Then, we use ReVerb (Fader, Soderland, and Etzioni 2011) to extract *(argument1, relation phrase, argument2)* triplets involving $d$ from the sentence formed by $q$ and $d$, $\{t_{11}, t_{12}, \cdots, t_{1n}\}$, as well as triplets in the titles and snippets returned by the search engine, $\{t_{21}, t_{22}, \cdots, t_{2m}\}$. After that, we calculate embedding similarities between triplets and keep the maximal score, $T(q, d)$, that represents the correctness of triplet 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 Word2Vec-based cosine similarity between $t_{1i}$ and $t_{2j}$. If $T(q, d)$ is small, then the sentence restored with the distractor $d$ is unlikely, thus $d$ should be a reliable distracter.

**Ranker** Given the feature vector $f(q, a, d) \in \mathbb{R}^{33}$ where $q$ and $a$ are the stem and key of triplet $(q; a; D_g)$ in the dataset, we propose to utilize a feature-based learning-to-rank model, which is trained in a supervised manner and learns to assign higher scores to those $d$'s within the ground-truth distractor set $D_g$ than those in $D_0 - D_g$. Reasonable distractors outside of $D_g$ are likely to be close to ground-truth distractors in the feature space $\mathbb{R}^{33}$, which can implicitly guide the ranker to learn relative ranking of negative examples during training.

Note that we do not restrict the ranker to be any specific model. One can choose to implement it using any state-of-the-art point-wise, pair-wise or list-wise learning-to-rank models. Theoretically, training a learning-to-rank model requires a relevance score associated with each distractor, which is not available in existing cloze-style MCQ dataset. We remedy this by setting the relevance score for $d \in D_g$ as 1 and for $d \in \{D_0 - D_g\}$ as 0. For point-wise ranker, it reduces into a binary-classifier (Liang et al. 2018). The major difference between the point-wise ranking model and pair/list-wise ranking model is that the latter may learn latent feature patterns for discriminating between better or worse distractors through supervised training signal.

At test time, the ranking score $s_i$ for each candidate distractor $d_i$ predicted by the ranker is then used to sort the can-

didates in $D_0$ extracted by CSG and output the final $n$-best ranked list $D = \{(d_1, s_1), (d_2, s_2), \cdots, (d_n, s_n)\}$.

## 3 Experiments

In this section, we first present the dataset and evaluation metrics used in our experiments. Then we investigate several design choices of our framework and compare them against previous methods. Code at https://github.com/DRSY/DGen.

### 3.1 The Dataset

Our MCQ dataset covers multiple domains including science, vocabulary, common sense and trivia. It is compiled from a wide variety of open source MCQ datasets including SciQ (Welbl, Liu, and Gardner 2017), MCQL (Liang et al. 2018), AI2 Science Questions as well as trivia, and vocabulary MCQs crawled from websites. We filter out MCQs whose keys are not single tokens since this paper only focuses on extractive cloze-style DG, resulting in 2,880 items in total among which 1176 are from SciQ, 300 are from MCQL, 275 are from AI2 and the rest from website resources. Statistics of the dataset are summarized in Table 1 and Figure 4.

We convert questions to cloze form by constructing Penn Treebank style trees using Stanford Parser (Klein and Manning 2003), replacing interrogative word with blank and adjusting node order according to the identified question type. The dataset is randomly divided into train/valid/test with a ratio of 8:1:1. We use the TreebankWord tokenizer and POS tagger from NLTK (Loper and Bird 2002) to preprocess the stems and keys when constructing features.

| Domain | Total | Science | Vocab. | Common Sense | Trivia |
|---|---|---|---|---|---|
| # MCQs | 2880 | 758 | 956 | 706 | 460 |
| # Distractors | 3.13 | 3.00 | 3.99 | 3.48 | 2.99 |

Table 1: Dataset Statistics (number of MCQs in each domain and average number of distractors per question).

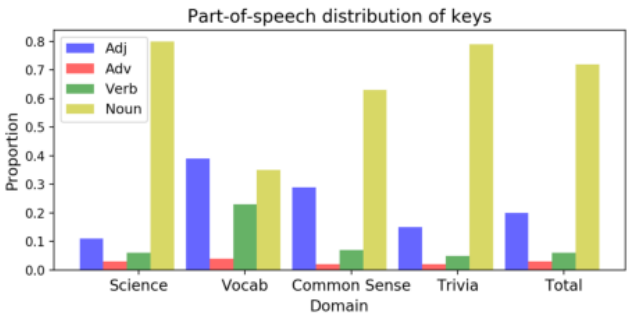


Figure 4: POS distribution of keys.

### 3.2 Evaluation Metrics

**Automatic Evaluation.** Following (Liang et al. 2018), we report F1 score(F1@3), precision(P@1, P@3) and recall (R@3) to show how well the generated distractors match the ground truth distractors, as well as the mean reciprocal rank (MRR) and normalized discounted cumulative gain (NDCG@10). Sometimes the generated distractors do not exactly match the ground truth, but are semantically very close. Word2Vec model trained on Wikipedia dump is utilized to measure the averaged cosine similarity (Semantic Similarity@3) between the top three generated distractors and ground truth distractors.

**Human Evaluation.** Following (Jiang and Lee 2017), we ask three proficient English speakers to evaluate distractors' reliability and plausibility by showing them the key. We evenly sample 50 items in all domains from test set, each item contains multiple distractors including 3 generated by each method and all ground truth distractors designed by human experts. For each distractor, the judges decided whether it is correct or incorrect given the context. For a distractor deemed to be incorrect, the reliability score is 1 and the judges further assess its plausibility on a 3-point scale: "Obviously Wrong" (0 point), "Somewhat Plausible" (1 point), or "Plausible" (2 points). We then conduct an application-centric evaluation using another 50 samples without keys from test set by extending the original sample with additionally generated distractors and asking testees to answer it. The kappa inter-annotator agreement scores are 0.65 and 0.74 respectively for plausibility and reliability.

### 3.3 Design Choices of CSG and DS

We investigate Probase and WordNet as the knowledge base in CSG and additionally extract all words and phrases from WordNet as a baseline of CSG in the following experiments. For Probase, both $p(c|a)$ and $p(d|c)$ are natively supported and can be obtained using official APIs. The size of a concept set $C$ is set to be 20. For nouns and verbs in WordNet, we treat the set of unique hypernyms (as well as their siblings) of all synsets for $a$ as concept set $C$ and compute $p(c|a)$ using the Laplace-smoothed Bayes rule on the lemma frequency provided in WordNet (count on sense tagged text). We choose all synsets and their similar/antonymic sets as concept set $C$ for adjectives and adverbs in WordNet. Topic distributions $\pi_{a,q}$ and $\gamma_c$ are obtained using LDA pre-trained on Wikipedia dump and $K$ is set to 100.

For DS, we experiment with point/pair/list-wise ranking models to find the best practice. Specifically, we employ AdaBoost (Freund and Schapire 1997) as a point-wise ranker and LambdaMART (Burges 2010) as both pair-wise and list-wise ranker. The dimensionality of feature vector $l$ is 33. Unigram frequency is calculated on Wikipedia dump. For the training of DS, negative examples are sampled using the top 100 candidates extracted by CSG excluding those that are within ground truths. At test time, DS takes as input the top 30 candidates extracted by CSG and 30 candidates sampled from WordNet's own vocabulary having the same POS tag. All hyperparameters are tuned on the dev set.

### 3.4 Baselines

We name our framework **CSG+DS** and compare it against the following baselines:

| Instantiation | | F1@3 | P@1 | P@3 | R@3 | MRR | NDCG@10 | Semantic Similarity@3 |
|---|---|---|---|---|---|---|---|---|
| CSG | DS | | | | | | | |
| WordNet | - | 3.14 | 3.49 | 2.33 | 5.43 | 7.19 | 8.66 | 0.27 |
| | point-wise ranker | 7.26 | 9.30 | 5.55 | 11.95 | 14.30 | 14.63 | 0.36 |
| | pair-wise ranker | 7.11 | **10.07** | 5.30 | 12.14 | **14.40** | 14.84 | 0.35 |
| | list-wise ranker | **7.71** | 9.31 | **5.81** | **12.98** | 14.34 | **14.94** | **0.36** |
| Probase | - | 5.88 | 6.98 | 4.39 | 9.95 | 12.07 | 13.40 | 0.35 |
| | point-wise ranker | 7.91 | 8.14 | 5.94 | 12.98 | 15.09 | 17.69 | 0.41 |
| | pair-wise ranker | **9.42** | 10.08 | **7.00** | 15.88 | 17.33 | **19.70** | 0.40 |
| | list-wise ranker | 9.19 | **10.85** | 6.72 | **15.88** | **17.51** | 19.31 | **0.41** |
| w/o CSG | - | - | - | - | - | - | - | - |
| | point-wise ranker | 5.59 | 4.63 | 3.98 | 10.29 | 8.67 | 11.02 | 0.36 |
| | pair-wise ranker | 5.62 | **5.01** | 3.98 | 10.10 | **9.28** | **11.60** | **0.36** |
| | list-wise ranker | **5.94** | 4.24 | **4.24** | **10.81** | 8.81 | 11.46 | 0.35 |

Table 2: Comparison of combinations of different choices of CSG and DS. - means no ranking.

- *Thesaurus-based Method (TM)* (Sumita, Sugaya, and Yamamoto 2005) ranks candidate distractors from synonyms of the key in WordNet-based on path similarity and applies post-filtering via IR.
- *RevUP* (Kumar, Banchs, and D'Haro 2015) ranks candidate distractors based on weighted average of Word2Vec-based cosine similarity, dice coefficient and language model probability.
- *EmbSim+CF* (Jiang and Lee 2017) combines Word2Vec-based cosine similarity, tri-gram and dependency candidate filtering in ranking and filtering respectively.
- *ED* use edit distance to measure the spelling similarity between distractors and key.
- *LR+RF* (Liang et al. 2018) combines logistic regression and random forest as a two-stage cascaded ranker with features measuring the plausibility of distractors.
- *LR+LM* (Liang et al. 2018) replaces random forest in LR+RF with LambdaMART.
- *BERT* (Devlin et al. 2018) ranks candidates using cosine similarity of their BERT embeddings with that of the key.

Trigram and 5-gram Kneser Ney language model are built upon the original corpus of our dataset. 400-dimensional Word2Vec (CBOW) is pretrained on Wikipedia dump and then fine-tuned on our corpus. Dependency tree is obtained using Spacy toolkit (Honnibal and Montani 2017). We adopt the bert-base-uncased model and fine-tune it on our corpus with MLM objective.

### 3.5 Results & Analysis

**Combinations of CSG and DS.** Table 2 shows the ranking performance for different combinations of CSG and DS. Without CSG, distractor selector trained with trivial negative examples is forced to select distractors from a rather large and noisy candidate set, therefore the performance is clearly worse. We also find that combining CSG with DS yields consistent improvement by all metrics and the improvement is more significant for WordNet CSG, which is mainly because $p(c|a)$ and $p(d|c)$ in WordNet are partly biased due to the limited scale of corpus they are estimated on, hence the supervised training will lead to more performance gain. Pair/list-wise ranker achieves comparable performance mainly due to the binarized relevance score. Since named entities and common nouns mainly underpin Probase, DS with Probase CSG naturally get higher ranking scores than its counterpart with WordNet CSG.

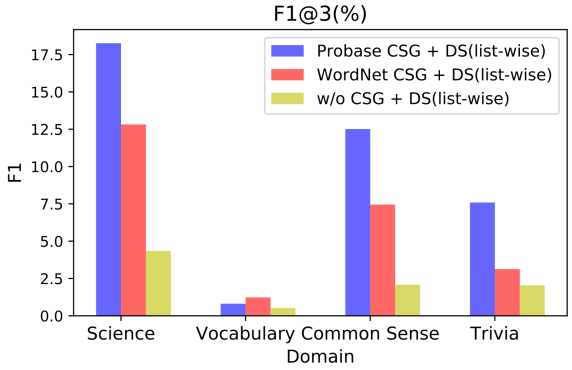


Figure 5: F1@3 score in different domains.

**Domain Effect & Feature Importance.** Figure 5 shows the F1@3 of CSG+DS in different domains. The performance drops most drastically when applied in the vocabulary domain because adjectives and adverbs in Probase and WordNet are either rare or not hierarchically organized. Another possible explanation is that the ground truth distractors in the vocabulary domain are less semantically-related to the key, which makes the learning process of the ranker oscillatory. Our framework is especially better at generating distractors in science and commonsense domain, in which the keys and distractors are mostly subject-specific (e.g. physics) terminologies, real-world entities and other common nouns. Trivia domain has similar characteristics but the keys are often rarer, therefore Probase suffers less due to its larger scope. To have more insights on the proposed features, we also conduct a feature importance analysis of DS based on mean reduced impurity. It is defined as the total decrease in node impurity, weighted by the probability of reaching that

| Method | Human Evaluation | | Automatic Evaluation | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Reliability | Plausibility | F1@3 | P@1 | P@3 | R@3 | MRR | NDCG@10 | Semantic Similarity@3 |
| TM | 95.57% | 1.25±0.41 | 1.74 | 0.40 | 1.16 | 3.48 | 2.69 | 4.79 | 0.21 |
| **WordNet CSG** | 98.66% | 1.25±0.34 | 3.14 | 3.49 | 2.33 | 5.43 | 7.19 | 8.66 | 0.26 |
| + ED | 90.66% | 1.26±0.41 | 0.41 | 0.12 | 0.26 | 0.58 | 2.10 | 1.93 | 0.20 |
| + RevUP | 93.65% | 1.22±0.34 | 4.07 | 5.79 | 3.21 | 6.43 | 9.31 | 9.60 | 0.32 |
| + EmbSim+CF | **99.12%** | 1.21±0.49 | 4.62 | 6.17 | 3.60 | 7.40 | 10.32 | 10.94 | 0.36 |
| + BERT | 89.94% | 1.23±0.58 | 5.68 | 6.93 | 4.23 | 9.57 | 11.10 | 11.66 | 0.30 |
| + LR+LM | 96.66% | 1.25±0.35 | 6.48 | 9.25 | 4.89 | 10.81 | 13.42 | 13.66 | 0.29 |
| + LR+RF | 95.56% | 1.25±0.38 | 6.67 | 8.10 | 5.14 | 10.81 | 13.18 | 13.73 | 0.30 |
| + **DS(lise-wise)** | 98.66% | **1.35±0.40** | **7.71** | **9.31** | **5.81** | **12.98** | **14.34** | **14.94** | **0.36** |
| **Probase CSG** | 99.23% | 1.26±0.35 | 5.88 | 6.98 | 4.39 | 9.95 | 12.07 | 13.40 | 0.34 |
| + ED | 94.33% | 1.23±0.38 | 0.82 | 1.16 | 0.65 | 1.30 | 5.02 | 4.92 | 0.28 |
| + RevUP | 94.87% | 1.26±0.36 | 6.27 | 5.40 | 4.63 | 10.68 | 11.74 | 14.23 | 0.37 |
| + EmbSim+CF | 96.98% | 1.19±0.47 | 7.01 | 8.10 | 5.14 | 12.34 | 13.86 | 16.33 | 0.41 |
| + BERT | 95.00% | 1.27±0.58 | 7.05 | 7.72 | 5.14 | 12.23 | 13.60 | 16.21 | 0.36 |
| + LR+LM | 98.98% | 1.25±0.30 | 7.62 | 8.53 | 5.81 | 12.27 | 15.56 | 16.83 | 0.40 |
| + LR+RF | 99.13% | 1.24±0.31 | 7.48 | 8.52 | 5.42 | 13.17 | 15.87 | 19.03 | 0.40 |
| + **DS(list-wise)** | **99.33%** | **1.30±0.34** | **9.19** | **10.85** | **6.72** | **15.88** | **17.51** | **19.31** | **0.41** |
| **w/o CSG** | - | - | - | - | - | - | - | - | - |
| + ED | 93.98% | 1.00±0.12 | 0.19 | 0.38 | 0.12 | 0.38 | 0.54 | 0.53 | 0.11 |
| + RevUP | 92.88% | 1.02±0.14 | 2.01 | 2.35 | 1.35 | 4.21 | 3.95 | 5.12 | 0.38 |
| + EmbSim+CF | 94.77% | 0.93±0.52 | 2.12 | 2.70 | 1.41 | 4.24 | 4.19 | 5.24 | **0.42** |
| + BERT | 93.87% | 1.02±0.24 | 3.03 | 2.88 | 2.15 | 5.14 | 5.29 | 6.78 | 0.39 |
| + LR+LM | 96.77% | 1.05±0.28 | 4.22 | 4.34 | 2.79 | 8.69 | 7.02 | 10.16 | 0.41 |
| + LR+RF | 97.78% | 1.02±0.20 | 4.05 | 4.21 | 2.66 | 8.55 | 6.91 | 10.08 | 0.40 |
| + **DS(pair-wise)** | **98.43%** | **1.06±0.14** | **5.59** | **5.01** | **3.98** | **10.10** | **9.28** | **11.60** | 0.36 |
| ground truth | **100%** | **1.41±0.35** | - | - | - | - | - | - | - |

Table 3: End-to-end comparison on test set. - means no ranking algorithm to evaluate and "ground truth" denotes the score of ground-truth distractors associated with each item. Results are obtained by three runs with different random seeds.

| Probase CSG | WordNet CSG |
|---|---|
| contextual embed sim(a,d) | contextual embed sim(a,d) |
| word2vec embed sim(a,d) | word2vec embed sim(a,d) |
| word2vec embed sim(q,d) | word2vec embed sim(q,d) |
| web search score | web search score |
| relative LCS len(d) | relative LCS len(d) |
| relative LCS len(a) | relative LCS len(a) |
| character len(d) | character len(d) |
| character len difference(a,d) | character len difference(a,d) |
| edit distance(a,d) | edit distance(a,d) |
| POS similarity(a,d) | relative common suffix len(a) |

Table 4: Top 10 important features of list-wise DS.

node, averaged over all base classifiers. Table 4 reveals that semantic relation between $a$ and $d$ and web search score play a more critical role than features of other aspects.

**End-to-End Comparison.** Table 3 shows the end-to-end results. Despite the significantly reduced number of candidates, ranking methods with our candidate set generator can achieve much higher performance than with unstructured vocabulary. TM performs poorly due to its naive path similarity ranking criterion. The results of ED are worst among all unsupervised methods while embedding based methods can even achieve comparable performance against LR+LM/RF when provided with a high-quality candidate set. BERT ranks distractors using contextualized representation thus leading to the lowest reliability according to human evaluation. LR+RF/LM achieves similar ranking performance yet obtain poorer reliability than CSG+DS since they only focus on the plausibility of selected distractors. CSG+DS, despite its relative simplicity, obtain consistent improvements over LR+RF/LM without two-stage cascaded training. We observe certain inconsistencies between plausibility and automatic metrics of baselines, part of the reason may be that methods such as LR+RF/LM focus much on shallow feature patterns of ground-truth distractors and fail to unearth potential acceptable distractors. However, distractors generated by CSG+DS yield highest-ranking measures while rated as most plausible by human annotators. Unsupervised methods work solely relying on the semantic similarity hence their reliabilities are generally lower than supervised ones, among which our DS turns out to be the most reliable. Exceptionally, EmbSim+CF gets higher reliability with WordNet, whose unreliable candidates get more chance to be eliminated by post-filtering than those in Probase.

**Application-Centric Evaluation.** The frequency of generated distractors being chosen as answer for each tested

| **Key** | RevUP | ED | EmbSim+CF | BERT | LR+RF | LR+LM | **DS** |
|---|---|---|---|---|---|---|---|
| **0.42** | 0.06 | 0.03 | 0.04 | 0.12 | 0.11 | 0.07 | **0.14** |

Table 5: Human evaluation on the frequency of being chosen as answers for each model paired with Probase CSG. DS denotes our list-wise distractor selector. Red colored number corresponds to the correct answer.

| # | Probase CSG | RevUP | ED | EmbSim+CF | BERT | LR+RF | LR+LM | **DS** |
|---|---|---|---|---|---|---|---|---|
| 1 | protein | protein | aldehydes | starch | glycosaminoglycans | hydrocarbon | methane | fat |
| 2 | alcohol | alcohol | carboxylic acid | glycerol | glycerol | methane | protein | protein |
| 3 | benzene | amino acid | alcohol | **glucose** | aldehydes | hormone | hormone | peptide |

Table 6: Top 3 distractors from different rankers running with Probase CSG given the stem "The main source of energy for your body is _____." and the key "carbohydrate". Red colored distractors are the ground truth, bold distractors are unreliable.

model is shown in Table 5. Our DS obtains the highest distracting rate compared to all baselines, indicating that distractors generated by our framework are more likely to distract testees in real-world scenarios. The Pearson correlation coefficient between the frequency and F1@3 is 0.46, implying a certain positive correlation between automatic metrics and actual distracting capacity.

### 3.6 Case Study

Table 6 compares predictions of all baselines and DS (list-wise) running with Probase CSG. We can see that Probase CSG alone and RevUP are both able to generate distractors belonging to the same conceptual level as the key and accurately match one ground truth. However, running Probase CSG with ED yields distractors that are more semantically distant from the key. Despite the use of candidate filtering, EmbSim+CF still produces candidates like "glucose", which is an eligible answer to the stem. BERT instead generates compound names that are too technical and belong to a lower concept level than ground truth. Among all the supervised rankers, DS hits another ground-truth distractor "fat" while LM+RF/LM predicts some obviously wrong distractors such as "methane" due to its coarse-grained features. A real-world mobile application using our framework can be found in Appendix B.

## 4 Related Work

Extractive distractor generation typically involves two steps: candidate set generation and distractor selection.In the common scenarios, only the key and the stem are known beforehand and the set of candidates need to be automatically generated. A prior solution is to construct a distractor candidate sets from domain-specific vocabulary, thesauri (Sumita, Sugaya, and Yamamoto 2005; Smith, Avinesh, and Kilgarriff 2010) or taxonomies (Mitkov et al. 2009). These domain-specific candidate sources are still not large or general enough to support open-domain distractor generation. In contrast, our framework is able to utilize a broad spectrum of general-purpose KBs and perform context-dependent conceptualization in an open-domain setting.

Previous approaches usually select distractors according to different metrics based on the key, including embedding-based similarities (Guo et al. 2016), difficulty level (Brown, Frishkoff, and Eskenazi 2005; Coniam 2013), WordNet-based metrics (Mitkov et al. 2003) and syntactic features (Agarwal and Mannem 2011). Some approaches also consider the semantic relatedness of distractors with the whole stem (Pino, Heilman, and Eskenazi 2008; Mostow and Jang 2012) with domain restriction. Other researchers (Liang et al. 2017, 2018) investigate applying learning-based ranking models to select distractors that resemble those in actual exam MCQs, and quantitatively evaluate the top generated distractors. The DS in our framework incorporates a wide range of similarity measures to account for the plausibility in various aspects.

To generate reliable distractors, a supervised classifier (Lee and Seneff 2007) is adopted where they have a limited list of potential target words and distractors. Another way to perform reliability checking is by considering collocations involving the target word (Smith, Avinesh, and Kilgarriff 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. A web search approach is applied by Sumita et al. (2005) to discard words that can be found on the web search results of the stem with blank filled by the distractor. We instead propose a novel web-based reliability checking feature and integrate it into DS for more accurate selection.

## 5 Conclusion

In this paper, we presented a configurable distractor generation framework for cloze-style open-domain MCQs. Using the proposed framework, we experimentally observe substantial performance gain in terms of distractor reliability and plausibility with less computational footprint. Depending on the characteristics (e.g. capacity, POS distribution) of different general-purpose knowledge bases, the generated distractors may vary. Importantly, as knowledge bases with larger coverage and more advanced ranker inevitably emerge, they can be expediently integrated into our framework for further performance gain.

## 6 Acknowledgement

Sincere gratitude goes to Xinzhu (Elena) Cai for her initial contribution and subsequent assistance in preparing this paper. This research was supported by the SJTU-CMBCC Joint Research Scheme, SJTU Medicine-Engineering Cross-disciplinary Research Scheme, and NSFC grant 91646205.

## References

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

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3: 993–1022. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944937.

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

Burges, C. J. 2010. From RankNet to LambdaRank to LambdaMART: An Overview. Technical Report MSR-TR-2010-82. URL https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/.

Coniam, D. 1997. A Preliminary Inquiry Into Using Corpus Word Frequency Data in the Automatic Generation of English Language Cloze Tests. *CALICO Journal* 14. doi:10.1558/cj.v14i2-4.15-33.

Coniam, D. 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.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

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

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

Guo, Q.; Kulkarni, C.; Kittur, A.; Bigham, J. P.; and Brunskill, E. 2016. Questimator: Generating knowledge assessments for arbitrary topics. In *IJCAI-16: Proceedings of the AAAI Twenty-Fifth International Joint Conference on Artificial Intelligence*.

Ha, L.; and Yaneva, V. 2018. Automatic Distractor Suggestion for Multiple-Choice Tests Using Concept Embeddings and Information Retrieval. 389–398. doi:10.18653/v1/W18-0548.

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

Honnibal, M.; and Montani, I. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Jiang, S.; and Lee, J. 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*, 143–148.

Kim, D.; Wang, H.; and Oh, A. 2013. Context-dependent conceptualization. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

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

Kumar, G.; Banchs, R.; and D'Haro, L. F. 2015. RevUP: Automatic Gap-Fill Question Generation from Educational Texts. In *Tenth Workshop on Innovative Use of Nlp for Building Educational Applications*.

Leacock, C.; and Chodorow, M. 1998. Combining local context and WordNet similarity for word sense identification. *WordNet: An electronic lexical database* 49(2): 265–283.

Lee, J.; and Seneff, S. 2007. Automatic generation of cloze items for prepositions. In *Eighth Annual Conference of the International Speech Communication Association*.

Liang, C.; Yang, X.; Dave, N.; Wham, D.; Pursel, B.; and Giles, C. L. 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*, 284–290.

Liang, C.; Yang, X.; Wham, D.; Pursel, B.; Passonneaur, R.; and Giles, C. L. 2017. Distractor generation with generative adversarial nets for automatically creating fill-in-the-blank questions. In *Proceedings of the Knowledge Capture Conference*, 33. ACM.

Loper, E.; and Bird, S. 2002. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, 63–70. Stroudsburg, PA, USA: Association for Computational Linguistics. doi:10.3115/1118108.1118117. URL https://doi.org/10.3115/1118108.1118117.

Mitkov, R.; Ha, L. A.; Varga, A.; and Rello, L. 2009. Semantic similarity of distractors in multiple-choice tests: extrinsic evaluation. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, 49–56. Association for Computational Linguistics.

Mitkov, R.; 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*.

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

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations.

Pho, V.-M.; André, T.; Ligozat, A.-L.; Grau, B.; Illouz, G.; François, T.; et al. 2014. Multiple Choice Question Corpus Analysis for Distractor Characterization. In *LREC*, 4284–4291.

Pino, J.; Heilman, M.; and Eskenazi, M. 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*, 22–32.

Sakaguchi, K.; Arase, Y.; and Komachi, M. 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, 238–242.

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

Sumita, E.; Sugaya, F.; and Yamamoto, S. 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*, 61–68. Association for Computational Linguistics.

Welbl, J.; Liu, N. F.; and Gardner, M. 2017. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209* .

Wu, W.; Li, H.; Wang, H.; and Zhu, K. Q. 2012. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, 481–492. ACM.