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.

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.

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

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>genes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(b)</td>
<td>DNA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(c)</td>
<td>factors</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(d)</td>
<td>generic code</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Stem

Key

Distractors

Figure 1: A cloze-style MCQ

Distractor generation, which aims to generate distinctive 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
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 2).
- 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 3).
- 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 4).

The Framework

As illustrated in Figure 2, our framework includes two components: Candidate Set Generator (CSG) and Distractor Selector (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.

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 $s_i$.

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.

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:

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

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.
the features are defined below: transforms it into a feature vector a stem, Feature Extractor ing steps. D which endows our framework with the flexibility to be com- while for some other KBs (e.g., WordNet) it is not the case, (subsumed by the concepts in 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: 
\[ p_i = p(d_i | a, q) \propto \sum_{c \in C} p(d_i | c) p(c | a, q) \tag{2} \]

where the probability \( p(d_i | c) \) is also known as typicality (Wu et al. 2012). The prior probability \( p(c | a) \) and typicality \( p(d_i | c) \) 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 \( D_0 = \{ (d_1, p_1), (d_2, p_2), \ldots , (d_m, p_m) \} \).

**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 \( d \), \( \{t_{11}, t_{12}, \ldots , t_{1n}\} \), as well as triplets in the titles and snippets returned by the search engine, \( \{t_{21}, t_{22}, \ldots , t_{2m}\} \). 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:

\[ T(q, d) = \max_{i \in \{1, 2, \ldots , n\}} \text{EmbSim}(t_{1i}, t_{2j}), \]

where \( \text{EmbSim}(t_{1i}, t_{2j}) \) 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 distractor.

**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 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 \( d \), \( \{t_{11}, t_{12}, \ldots , t_{1n}\} \), as well as triplets in the titles and snippets returned by the search engine, \( \{t_{21}, t_{22}, \ldots , t_{2m}\} \). 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:

\[ T(q, d) = \max_{i \in \{1, 2, \ldots , n\}} \text{EmbSim}(t_{1i}, t_{2j}), \]

where \( \text{EmbSim}(t_{1i}, t_{2j}) \) 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 distractor.

**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 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 \( d \), \( \{t_{11}, t_{12}, \ldots , t_{1n}\} \), as well as triplets in the titles and snippets returned by the search engine, \( \{t_{21}, t_{22}, \ldots , t_{2m}\} \). 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:

\[ T(q, d) = \max_{i \in \{1, 2, \ldots , n\}} \text{EmbSim}(t_{1i}, t_{2j}), \]

where \( \text{EmbSim}(t_{1i}, t_{2j}) \) 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 distractor.
didates in $D_0$ extracted by CSG and output the final n-best ranked list $D = \{(d_1, s_1), (d_2, s_2), \cdots , (d_n, s_n)\}$.

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.

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 websites. 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 data. All hyperparameters are tuned on the dev set.

Table 1: Dataset Statistics.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Total</th>
<th>Science</th>
<th>Vocab.</th>
<th>Common Sense</th>
<th>Trivia</th>
</tr>
</thead>
<tbody>
<tr>
<td># MCQs</td>
<td>2880</td>
<td>758</td>
<td>956</td>
<td>706</td>
<td>460</td>
</tr>
<tr>
<td># Distractors</td>
<td>3.13</td>
<td>3.00</td>
<td>3.99</td>
<td>3.48</td>
<td>2.99</td>
</tr>
</tbody>
</table>

Figure 4: POS distribution of keys.

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.

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 et al. 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.

Baselines
We name our framework CSG+DS and compare it against:

• Thesaurus-based Method (TM) (Sumita, Sugaya, and Yamamoto 2005) ranks candidate distractors from synonyms
Results & Analysis

Combinations of CSG and DS. Table 2 shows the 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 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 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 it with WordNet CSG.

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 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 can-
<table>
<thead>
<tr>
<th>Method</th>
<th>Human Evaluation</th>
<th>Automatic Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reliability</td>
<td>Plausibility</td>
</tr>
<tr>
<td>TM</td>
<td>95.57%</td>
<td>1.25±0.41</td>
</tr>
<tr>
<td>WordNet CSG</td>
<td>98.66%</td>
<td>1.25±0.34</td>
</tr>
<tr>
<td>+ ED</td>
<td>90.66%</td>
<td>1.26±0.41</td>
</tr>
<tr>
<td>+ RevUP</td>
<td>93.65%</td>
<td>1.22±0.34</td>
</tr>
<tr>
<td>+ EmbSim+CF</td>
<td>99.12%</td>
<td>1.21±0.49</td>
</tr>
<tr>
<td>+ BERT</td>
<td>89.94%</td>
<td>1.23±0.58</td>
</tr>
<tr>
<td>+ LR+LM</td>
<td>96.66%</td>
<td>1.25±0.35</td>
</tr>
<tr>
<td>+ LR+RF</td>
<td>95.56%</td>
<td>1.25±0.38</td>
</tr>
<tr>
<td>+ DS(list-wise)</td>
<td>98.66%</td>
<td>1.35±0.40</td>
</tr>
<tr>
<td>Probase CSG</td>
<td>99.23%</td>
<td>1.26±0.35</td>
</tr>
<tr>
<td>+ ED</td>
<td>94.33%</td>
<td>1.23±0.38</td>
</tr>
<tr>
<td>+ RevUP</td>
<td>94.87%</td>
<td>1.26±0.36</td>
</tr>
<tr>
<td>+ EmbSim+CF</td>
<td>96.98%</td>
<td>1.19±0.47</td>
</tr>
<tr>
<td>+ BERT</td>
<td>95.00%</td>
<td>1.27±0.58</td>
</tr>
<tr>
<td>+ LR+LM</td>
<td>98.98%</td>
<td>1.25±0.30</td>
</tr>
<tr>
<td>+ LR+RF</td>
<td>99.13%</td>
<td>1.24±0.31</td>
</tr>
<tr>
<td>+ DS(list-wise)</td>
<td>99.33%</td>
<td>1.30±0.34</td>
</tr>
<tr>
<td>w/o CSG</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>+ ED</td>
<td>93.98%</td>
<td>1.00±0.12</td>
</tr>
<tr>
<td>+ RevUP</td>
<td>92.88%</td>
<td>1.02±0.14</td>
</tr>
<tr>
<td>+ EmbSim+CF</td>
<td>94.77%</td>
<td>0.93±0.52</td>
</tr>
<tr>
<td>+ BERT</td>
<td>93.87%</td>
<td>1.02±0.24</td>
</tr>
<tr>
<td>+ LR+LM</td>
<td>96.77%</td>
<td>1.05±0.28</td>
</tr>
<tr>
<td>+ LR+RF</td>
<td>97.78%</td>
<td>1.02±0.20</td>
</tr>
<tr>
<td>+ DS(pair-wise)</td>
<td>98.43%</td>
<td>1.06±0.14</td>
</tr>
<tr>
<td>ground truth</td>
<td>100%</td>
<td>1.41±0.35</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Probase CSG</th>
<th>WordNet CSG</th>
</tr>
</thead>
<tbody>
<tr>
<td>contextual embed sim(a,d)</td>
<td>contextual embed sim(a,d)</td>
</tr>
<tr>
<td>word2vec embed sim(a,d)</td>
<td>word2vec embed sim(a,d)</td>
</tr>
<tr>
<td>word2vec embed sim(q,d)</td>
<td>word2vec embed sim(q,d)</td>
</tr>
<tr>
<td>web search score</td>
<td>web search score</td>
</tr>
<tr>
<td>relative LCS len(d)</td>
<td>relative LCS len(d)</td>
</tr>
<tr>
<td>relative LCS len(a)</td>
<td>relative LCS len(a)</td>
</tr>
<tr>
<td>character len(d)</td>
<td>character len(d)</td>
</tr>
<tr>
<td>character len difference(a,d)</td>
<td>character len difference(a,d)</td>
</tr>
<tr>
<td>edit distance(a,d)</td>
<td>edit distance(a,d)</td>
</tr>
<tr>
<td>POS similarity(a,d)</td>
<td>relative common suffix len(a)</td>
</tr>
</tbody>
</table>

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

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+RF/LM 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 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, imply-
<table>
<thead>
<tr>
<th>#</th>
<th>Probase CSG</th>
<th>RevUP</th>
<th>ED</th>
<th>EmbSim+CF</th>
<th>BERT</th>
<th>LR+RF</th>
<th>LR+LM</th>
<th>DS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>protein</td>
<td>protein</td>
<td>aldehydes</td>
<td>starch</td>
<td>glycosaminoglycans</td>
<td>hydrocarbon</td>
<td>methane</td>
<td>fat</td>
</tr>
<tr>
<td>2</td>
<td>alcohol</td>
<td>alcohol</td>
<td>carboxylic acid</td>
<td>glycerol</td>
<td>glycerol</td>
<td>methane</td>
<td>protein</td>
<td>protein</td>
</tr>
<tr>
<td>3</td>
<td>benzene</td>
<td>amino acid</td>
<td>alcohol</td>
<td>glucose</td>
<td>aldehydes</td>
<td>hormone</td>
<td>hormone</td>
<td>peptide</td>
</tr>
</tbody>
</table>

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.

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”. Underlined distractors are the ground truth, bold distractors are unreliable.

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.

Acknowledgments
Sincere gratitude goes to Xinzhu 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


