Enhancing Distractor Generation for Multiple-Choice Questions with Retrieval Augmented Pretraining and Knowledge Graph Integration

Han-Cheng Yu, Yu-An Shih, Kin-Man Law, Kai-Yu Hsieh, Yu-Chen Cheng, Hsin-Chih Ho, Zih-An Lin, Wen-Chuan Hsu, Yao-Chung Fan*
Department of Computer Science and Engineering, National Chung Hsing University, Taiwan
yfan@nchu.edu.tw

Abstract

In this paper, we tackle the task of distractor generation (DG) for multiple-choice questions. Our study introduces two key designs. First, we propose retrieval augmented pretraining, which involves refining the language model pretraining to align it more closely with the downstream task of DG. Second, we explore the integration of knowledge graphs to enhance the performance of DG. Through experiments with benchmarking datasets, we show that our models significantly outperform the state-of-the-art results. Our best-performing model advances the F1@3 score from 14.80 to 16.47 in MCQ dataset and from 15.92 to 16.50 in Sciq dataset.

1 Introduction

Multiple-Choice Questions (MCQs) are widely used to evaluate a learner’s knowledge. However, creating them manually requires a significant amount of time and effort from educators. Well-designed MCQs help in accurately assessing learners’ abilities. The most challenging part of creating MCQs is designing appropriate incorrect options, commonly called distractors. Therefore, researchers have focused on the automatic distractor generation for MCQs in recent years. In this paper, we explore two directions for improving DG methods.

Task-Specific Pretraining Current SOTA DG methods, such as utilizing BERT for DG (Chiang et al., 2022) or harnessing T5 for Text2Text DG (Wang et al., 2023), are all based on pre-trained language models (LMs). Existing LMs undergo task-agnostic pretraining. However, recently, the idea of task-specific pretraining has gained prominence. The aim is to refine the pretraining process to closely align with the downstream task. Therefore, our first endeavor in this paper is to investigate Task-Specific Pretraining for the DG task. In line with this goal, we propose Retrieval Augmented Pretraining (RAP) for task-specific pretraining. The main idea involves using MCQ answers to retrieve relevant sentences/passages from a large corpus, such as Wikipedia, to create pseudo questions, and then use the generated pseudo question for task-specific pertaining.

Knowledge Augmented Generation Recent research trends show that leveraging knowledge graphs (KGs) can enhance the performance of LMs-based text generation tasks, as discussed in (Yasunaga et al., 2022) and (Zhang et al., 2022). The perspective is that LMs and KGs should complement each other. KGs go beyond text by offering structural information, representing entities as nodes and their relationships as edges. This structured knowledge enhances the ability for multi-step reasoning. In line with this motivation, we propose knowledge-augmented generation (KAG), where we build upon the candidate augmentation strategy proposed by the state-of-the-art DG method (Wang et al., 2023). The idea is to retrieve knowledge triplets from a knowledge graph to serve as auxiliary information for the text2text DG model. The contributions of this paper are

• Our methods achieve a remarkable improvement in state-of-the-art DG results. Our best-performing model elevates the F1@3 score from 14.80 to 16.47 in MCQ and from 15.92 to 16.50 in Sciq, showcasing the effectiveness
Figure 1: Retrieved Augmented Pretraining of our approach.

- Extensive experimental evaluation with the benchmarking datasets are conducted and the insights of incorporating task-pretraining and knowledge triplet provision are discussed. Our study unveils promising directions for further development in DG by showcasing the efficacy of knowledge augmentation and task-specific pretraining.

The rest of this paper is organized as follows. Section 2 reviews the works of automatic distractor generation in the literature. In Section 3 we present the proposed methods. Section 4 reports the performance evaluation and Section 5 concludes this work and discuss the future work.

2 Related Work

2.1 Distractor Generation

The current DG research can be divided into two directions:

**Generating and Ranking (GR) framework** 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. Specifically, the GR architecture consists of two stages.

First, it generates candidate distractors, and then it ranks these candidates based on semantic rules and linguistic features to select the final distractor. In the GR framework, there are two approaches for generating distractor candidates: using a knowledge base (Ren and Zhu, 2021) or using a language model (LM) (Chiang et al., 2022). These approaches have significantly improved the quality and diversity of distractors compared to traditional rule-based methods (Liang et al., 2017, 2018), making them the state-of-the-art in DG.

**Text2Text generation architecture** The Text2Text generation architecture differs from the GR architecture by formulating distractor generation as a Text2Text task. Specifically, it concatenates the question stem with the answer and inputs it into a generative language model (e.g., T5 or GPT) to train the model to generate a set of distractors. Currently, research based on the Text2Text architecture (Wang et al., 2023) represents the state-of-the-art in distractor generation.

2.2 Task-specific Pretraining

In recent related work, researchers have explored the integration of task-specific priors into BERT language model pretraining to enhance performance in low-resource fine-tuning tasks (Wang et al., 2020). Traditional pretraining methods often prioritize generic natural language knowledge, overlooking task-specific information, which can lead to overfitting in low-resource downstream tasks. To address this issue, (Wang et al., 2020) propose integrating task-specific label embeddings into the self-attention layers during pretraining. This integration enables the model to filter out task-irrelevant information and improve task-specific knowledge during fine-tuning, resulting in reduced overfitting and improved performance on downstream tasks with limited resources.

Another related work, (Zhang et al., 2020), adopts a task-specific pretraining approach for text summarization tasks. The authors replace a portion of the original sentences in the text with mask tokens, allowing the model to learn to generate masked sentences. Comparing this approach to traditional masked language modeling (MLM) for pretraining, they find that closer alignment between the pretraining task and the downstream task leads to improved performance. Additionally, they experiment with pretraining on various types of datasets and observe that utilizing a dataset similar to the downstream task yields better results. For example, when pretraining on a news dataset and fine-tuning on another news dataset, the performance surpasses that of fine-tuning on a web article dataset.

Building upon these findings, we draw inspiration from the aforementioned studies to design our proposed approach, known as RAP (Retrieval-Augmented Pretraining), aimed at enhancing the performance of distractor generation (DG). RAP leverages task-specific priors and incorporates them into the pretraining phase to capture and uti-
lize relevant information specific to the distractor generation task. By aligning the pretraining task with the downstream DG task and considering task-specific information, we aim to improve the quality and effectiveness of distractor generation.

2.3 Knowledge Augmented Generation

We have conducted a survey of research that leverages Knowledge Graphs (KGs) to enhance performance in various Natural Language Processing (NLP) tasks, including Question Answering, Entity Typing/Relation Classification, Query Answering, Question Generation, and Distractor Generation (Yasunaga et al., 2022, 2021; Zhang et al., 2022; Feng et al., 2020; He et al., 2019; Ren et al., 2020; Fei et al., 2022; Chen et al., 2023; Ren and Zhu, 2021).

Notably, a significant amount of research has focused on using Knowledge Graphs to improve question-answering systems, particularly for tasks involving multi-hop reasoning. These studies utilize Knowledge Graph triplets for inferencing and have demonstrated the utility of structured background knowledge in strengthening NLP tasks.

For example, (Yasunaga et al., 2021) proposed an end-to-end question-answering model that combines language models and Knowledge Graphs to handle reasoning tasks. (Zhang et al., 2022) introduced a novel model that incorporates joint information exchange between language models and Knowledge Graphs, achieving superior performance across multiple domains.

While Knowledge Graph integration has shown promising results in Question-Answering tasks, its application to Distractor Generation remains relatively limited (Ren and Zhu, 2021).

Therefore, our goal is to leverage KGs as auxiliary tools to enhance distractor generation, improving the relevance of generated options to the question. This approach not only creates more challenging options for multiple-choice questions but also enhances the model’s efficiency in utilizing Knowledge Graphs, leading to more effective Knowledge Augmented Generation.

3 Methodology

3.1 Retrieval Augmented Pretraining

Our approach involves generating pseudo questions to continuously train a pre-trained language model (e.g., T5 or BART) before fine-tuning it for downstream Distractor Generation (DG) tasks.

Here’s how our idea works: Given an MCQ training instance consisting of a question stem ($q$), an answer option ($a$), and a set of distractors ($D$), we utilize $a$ to retrieve a single sentence or a short passage from a corpus (such as Wikipedia). Subsequently, we mask out $a$ from the retrieved sentence/passage, replacing the answer position with a mask token ($\text{[Mask]}$). This process results in a modified sentence/passage known as a pseudo question ($\tilde{q}_{\text{[Mask]}}$).

With $\tilde{q}_{\text{[Mask]}}$, our goal is to pretrain a LM as follows.

$$L(\theta) = - \sum_{i=1}^{[D]} t_i \log p(\tilde{t}_i|\tilde{q}_{\text{[Mask]}}, a; \theta)$$

- $t_i$: the token sequence given by the concatenation of the ground truth distractors $d_1||d_2||d_3$.

Alternatives for RAP Training Setting

There are two alternatives for RAP training setting.

- **Data Augmentation RAP**: We can treat RAP as a data augmentation mechanism by using the same dataset for RAP and DG.

- **Cross-Domain RAP**: We can use two different distractor generation datasets (SciQ and MCQ datasets in this study). One dataset is used for RAP and the other for DG fine-tuning.

There are different impacts and implications for RAP training adoption. We provide experimental study and discussion for this part in Subsection 4.6.

Boosting RAP with ground-truth distractor

The RAP idea can be extended to using ground-truth distractors to generate pseudo questions. That is, we can use ground-truth distractors to retrieve a sentence/passage and set the rest of options (i.e., $a$ and other distractors). That is,

$$L(\theta) = - \sum_{i=1}^{[D]} t_i \log p(\tilde{t}_i|\tilde{q}_{\text{[Mask]}}, d_1; \theta)$$

- $t_i$: the token sequence given by the concatenation of the other options $a||d_2||d_3$.

3.2 Knowledge Augmented Generation

We extend the idea of the candidate augmentation strategy proposed by (Wang et al., 2023). We explore incorporating knowledge triplets for DG. Specifically, as shown in Figure 2, our KAG proceeds by (1) a knowledge triplet retrieval stage.
(Subsection 3.2.1) and (2) a triplet re-ranker stage (Subsection 3.2.2). The retriever stage is to retrieve triplets with respect to $q$ and $a$, and the re-ranker stage aims to rank the triplets by their estimated relevancy with respect to $q$ and $a$.

### 3.2.1 Retrieving Triplet from KG

Our process for obtaining knowledge triplets is as follows. First, for a given $q$ and answer $a$, we employ an language model trained for distractor generation (e.g., (Chiang et al., 2022)) to generate a set of candidate distractors $\{d_1, \ldots, d_k\}$. In addition, we conduct keyword extraction over $q$ and $a$ to extract keywords from them. We then union the two sets of keywords (denoted by $W$).

As illustrated in Figure 3, with $W$ and a given knowledge graph $G(V, E)$, the triplet set is given by

$$K = \{(u, e_u, v) \mid u, v \in W \text{ and } e_{u, v} \in E\}$$

### 3.2.2 Triplet Ranker

The input to triplet ranker is a triplet set $T$ from KG Retrieval stage and let $q||a$ be the concatenation of a question $q$ and an answer $a$. We experiment with the following two alternatives for ranking triplets.

1. **Unsupervised Sentence Embedding Distance**: The ranking process begins by encoding $q||a$ and each knowledge triplet $\tau_i$ into their respective vector representations. Next, we compute a relevancy score for each triplet by comparing its vector representation with that of $q||a$. The higher the relevancy score, the more likely the triplet is deemed relevant to $q||a$. Specifically,

   $$\text{scores} = |\text{Cos-Similarity}(\tau_i, q||a), \forall \tau_i \in K|$$

   We select top-$k$ triplets for the later knowledge augmented generation.

2. **Supervised Triplet Classification**: As an alternative, we propose a binary classification formulation to estimate the relevancy of a knowledge triplet $\tau_i$ with respect to $q||a$. During the training phase, we label a triplet as relevant if it contains the answer or ground-truth distractors, and as irrelevant otherwise. Specifically, we concatenate $q||a$ and $\tau_i$, separated by a special token [SEP] as input and the relevancy as output. During the inference phase, we utilize the LM to re-sort the top-$k$ triplet set based on the confidence scores.

### 3.2.3 KAG Training

For a given training instance $(q, a, D, K)$, our goal is to train a generation model conditioned on $q$, $a$, and $K$ by minimizing the negative log-likelihood of the correct token $t_i$ of $D$ given the preceding tokens and the conditions.

- $K$: the set of knowledge triplet $\{\tau_1, \ldots, \tau_k\}$ retrieved from a KG.

The idea is to retrieve a set of knowledge triplet $\{\tau_1, \ldots, \tau_k\}$ from a KG and concatenate the knowledge triplet with the original input text as an augmented text input for generation. Specifically, the loss function is

$$L(\theta) = -\sum_{i=1}^{|D|} t_i \log p(\hat{t}_i|\hat{t}_{<i}, q, a, \{\tau_i\}; \theta)$$

The observation behind the knowledge triplet augmentation strategy is to inject more information for generation through the knowledge graph in hope to boost the DG performance.

### 4 Experiment

#### 4.1 Dataset

We use SciQ (Welbl et al., 2017) and MCQ dataset (the dataset released by Ren and Zhu, 2021) for performance evaluation. For the details about the two benchmark datasets and our dataset setting.

**Sciq dataset** The Sciq dataset is a multi-domain multiple-choice question dataset consisting of 13,000 questions in the fields of physics, chemistry, biology, and other natural sciences. These questions are open-ended and require reading and understanding relevant scientific knowledge, followed by reasoning and answering the questions. The questions are presented in a multiple-choice format, with each question accompanied by an answer and three incorrect options. According to the Sciq dataset configuration, the 13,000 data points are split into train/dev/test datasets with a ratio of 11,700/1,000/1,000.

Since the Sciq question sentences do not contain **blanks**, in the candidate generation method, we utilize generative LM, specifically ChatGPT, to generate the candidate set.

**MCQ dataset** MCQ dataset is a cross-domain cloze-style dataset, that includes the domains of science, vocabulary, common sense, and trivia. Each data is composed of a sentence containing
**Figure 2: Knowledge Augmented Generation**

**Figure 3: Retrieve Triplet from KG: we extract keyword from a given question, answer and candidate set as a entity set $W$ and retrieve relevant triplet set from KG with keyword entities**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MCQ</th>
<th>Sciq</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Questions</td>
<td>Train</td>
<td>Dev</td>
</tr>
<tr>
<td></td>
<td>2088</td>
<td>233</td>
</tr>
</tbody>
</table>

Table 1: The statistics of the training, development and test sets of MCQ, Sciq.

**blank** of cloze stem, answer, and distractors. According to the setting reported by (Ren and Q. Zhu, 2021), MCQ contains 2880 questions and is randomly divided into train/dev/test with a ratio of 8:1:1.

We obtain the MCQ dataset from GitHub link shared by (Ren and Q. Zhu, 2021). However, we find there is a slight difference between the numbers in the shared dataset and reported in the paper. In the shared dataset, it only contains train and test data (with 2321/258). Thus, we use this data setting in our experiments. For dev data, we use 9:1 split from train as dev data.

### 4.2 Evaluation Metrics

**Automatic Metric** Following the SOTA DG by (Wang et al., 2023), we evaluate the DG quality by F1 score (F1@3), precision (P@1, P@3), and recall (R@1, R@3). P@k represents the ratio of correctly labeled top-k generated distractors, while R@k indicates the ratio of correctly predicted labels among the ground truth. F1@k is the harmonic mean of P@k and R@k. Notably, when the label size is 3, P@3 and R@3 will be the same, resulting in the same F1@3 score. Since both the Sciq test data and MCQ test data contain 3 distractors, we report the scores of P@1 and F1@3 in the experiments.

**Human Evaluation Metric** In addition to Automatic Metrics, we also consider human evaluation to assess the performance of the model on various aspects. During the evaluation process, we randomly select 14 questions and generate distractor sets using various models, which are then assessed by five human labelers. For the evaluation of the
Distractor Generation models, we consider the following criteria for human evaluation:

1. **Relevance**: This metric assesses the relevance of the generated distractors to the question. This means that the distractors should be related to the theme, content, or context of the question.

2. **Distractiveness**: This metric evaluates the level of distraction that the generated distractors bring to the correct answer. The distractors should be able to divert the answer away from the correct answer, increasing the difficulty of the question.

3. **Utility**: This is an overall quality assessment of the generated distractors, considering aspects such as fluency, relevance, duplication, and distractiveness. This is a comprehensive metric that takes into account all facets of performance.

We employ a five-point rating system, where each score interval represents different levels of relevance, ranging from "totally irrelevant" (1 point) to "completely relevant" (5 points). These evaluation metrics will help us assess the quality and performance of the distractors generated by the models in a comprehensive manner.

### 4.3 Implementation Details

See Appendix.

### 4.4 Evaluation Results

Table 2 presents the results of the compared methods on the two benchmarking datasets. We have the following notes for the results.

**RAP indeed improve performance** By comparing the performance difference between Text2Text (T5) and Text2Text (RAP-T5), we can see that the introduction of RAP elevates the performance of MCQ experiments from 11.45 to 14.80 (F1@3). In the case of Sciq, it increases from 15.92 to 16.06.

**RAP more effective in low resource settings** Furthermore, continuing the above observations, we found that the improvement brought by RAP in the Sciq experiment is not as significant as in the MCQ experiment. We attribute this result to the much smaller size of the MCQ dataset compared to Sciq. When the training data is abundant, the gains from pretraining naturally become limited. Therefore, in the MCQ setting with only 2088 training examples, the introduction of RAP has a noticeable impact.

**KG also boosts the performance** By comparing T5 candidate augmentation with KAG (T5), we observe that the introduction of KG indeed brings a significant improvement. In Sciq, we observe an increase in F1@3 from 12.83 to 15.38. In MCQ, it increases from 14.80 to 16.47.

**Our methods outperform ChatGPT** We also compare the performance of ChatGPT (in a zero-shot manner) on both datasets. From the experimental results, it is evident that our best-performing models consistently outperform ChatGPT. In the comparison with Sciq, KAG (RAP-T5) achieves an F1@3 score of 15+ compared to ChatGPT’s 10.61. An interesting observation is that while ChatGPT still lags behind the KAG+RAP-based approach in the MCQ comparison, the performance gap is smaller. We attribute this to the fact that some questions in Sciq require common-sense reasoning in addition to knowledge-based answering. Since ChatGPT generates responses in a zero-shot learning manner during testing, it does not fully capture the characteristics of the Sciq dataset. In the case of the MCQ dataset, due to the limited training data, the differences between various methods and ChatGPT are not as substantial as in the Sciq comparison.

**Combining KAG with RAP did not yield additive effects** We also observed that combining KAG with RAP did not yield additive effects. Taking the comparison of the T5 model on the Sciq dataset as an example, we found that the performance did not improve when using both KAG and RAP together (KAG(RAP-T5): 15.15) compared to using RAP alone (Text2Text(RAP-T5): 16.06) or KAG alone (KAG(T5): 15.38). Similar observations were made in other settings. We speculate that the additional noise introduced by KAG during the incorporation of knowledge triplets may be the reason behind this. Although we designed a reranker to select triplets, the results were not consistently stable. To validate this hypothesis, we attempted a variation of KAG (RAP-T5) (denote by with-only-answer triplet) where, instead of using a reranker, we selected triplets containing the answer to input into the Text2Text generator. We observed significant improvement with this approach in the Sciq dataset, but no improvement in the MCQ dataset. This suggests that there is considerable room for improvement in the design of the triplet ranker. Se-
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>P@1</th>
<th>R@1</th>
<th>F1@3</th>
<th>MRR</th>
<th>NDCG@3</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCQ</td>
<td>(Ren and Zhu, 2021)</td>
<td>10.58</td>
<td>-</td>
<td>9.19</td>
<td>17.51</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>(Chiang et al., 2022)</td>
<td>10.81</td>
<td>3.60</td>
<td>7.72</td>
<td>18.15</td>
<td>15.39</td>
</tr>
<tr>
<td></td>
<td>(Wang et al., 2023) (BART)</td>
<td>14.28</td>
<td>4.76</td>
<td>11.45</td>
<td>21.49</td>
<td>23.70</td>
</tr>
<tr>
<td></td>
<td>Text2text (RAP-BART)</td>
<td>12.16</td>
<td>4.57</td>
<td>9.25</td>
<td>16.35</td>
<td>17.80</td>
</tr>
<tr>
<td></td>
<td>(Wang et al., 2023) (BART w/ c.a.)</td>
<td>14.28</td>
<td>4.76</td>
<td>11.45</td>
<td>21.49</td>
<td>23.70</td>
</tr>
<tr>
<td></td>
<td>KAG (BART)</td>
<td>20.07</td>
<td>6.69</td>
<td>14.41</td>
<td>27.02</td>
<td>29.37</td>
</tr>
<tr>
<td></td>
<td>KAG (RAP-BART)</td>
<td>16.60</td>
<td>5.75</td>
<td>12.22</td>
<td>22.65</td>
<td>24.13</td>
</tr>
<tr>
<td></td>
<td>KAG (BART) (with only answer triplet)</td>
<td>14.28</td>
<td>4.76</td>
<td>11.45</td>
<td>21.49</td>
<td>23.70</td>
</tr>
<tr>
<td></td>
<td>KAG (RAP-BART)(with only answer triplet)</td>
<td>13.12</td>
<td>4.37</td>
<td>11.84</td>
<td>20.01</td>
<td>22.35</td>
</tr>
<tr>
<td></td>
<td>(Wang et al., 2023) (T5)</td>
<td>18.53</td>
<td>6.17</td>
<td>11.45</td>
<td>23.61</td>
<td>25.08</td>
</tr>
<tr>
<td></td>
<td>Text2text (RAP-T5)</td>
<td>12.16</td>
<td>4.57</td>
<td>9.25</td>
<td>16.35</td>
<td>17.80</td>
</tr>
<tr>
<td></td>
<td>(Wang et al., 2023) (T5 w/ c.a.)</td>
<td>14.28</td>
<td>4.76</td>
<td>11.45</td>
<td>21.49</td>
<td>23.70</td>
</tr>
<tr>
<td></td>
<td>KAG (T5)</td>
<td>20.07</td>
<td>6.69</td>
<td>14.41</td>
<td>27.02</td>
<td>29.37</td>
</tr>
<tr>
<td></td>
<td>KAG (RAP-T5)</td>
<td>16.60</td>
<td>5.75</td>
<td>12.22</td>
<td>22.65</td>
<td>24.13</td>
</tr>
<tr>
<td></td>
<td>KAG (T5)(with only answer triplet)</td>
<td>14.28</td>
<td>4.76</td>
<td>11.45</td>
<td>21.49</td>
<td>23.70</td>
</tr>
<tr>
<td></td>
<td>KAG (RAP-T5)(with only answer triplet)</td>
<td>13.12</td>
<td>4.37</td>
<td>11.84</td>
<td>20.01</td>
<td>22.35</td>
</tr>
<tr>
<td></td>
<td>(Wang et al., 2023) (T5 w/ c.a.)</td>
<td>14.28</td>
<td>4.76</td>
<td>11.45</td>
<td>21.49</td>
<td>23.70</td>
</tr>
<tr>
<td></td>
<td>KAG (T5)(with only answer triplet)</td>
<td>13.12</td>
<td>4.37</td>
<td>11.84</td>
<td>20.01</td>
<td>22.35</td>
</tr>
<tr>
<td></td>
<td>KAG (T5)(with only answer triplet)</td>
<td>14.28</td>
<td>4.76</td>
<td>11.45</td>
<td>21.49</td>
<td>23.70</td>
</tr>
<tr>
<td></td>
<td>KAG (RAP-T5)(with only answer triplet)</td>
<td>13.12</td>
<td>4.37</td>
<td>11.84</td>
<td>20.01</td>
<td>22.35</td>
</tr>
<tr>
<td></td>
<td>ChatGPT</td>
<td>18.91</td>
<td>6.30</td>
<td>13.38</td>
<td>25.86</td>
<td>27.97</td>
</tr>
</tbody>
</table>

Table 2: DG Results on the Compared Datasets: In this set of experiment, we mainly compare our designs with the SOTA method (Wang et al., 2023). c.a. represents candidate augmentation strategy introduced by (Wang et al., 2023).

Table 3: The statistics of the RAP training. Note that our RAP has the following variations. First, in RAP, we can choose to retrieve a single sentence or a passage, giving rise to different RAP strategies. Thus, we denote RAP-S as retrieving sentence for pseudo question generation and RAP-P for retrieving passage. Second, we have the option to use the answer or the distractor as the starting point for retrieving the sentence/passage. We denote the method using distractor as starting point as RAP-S w/GTD or RAP-S/RAP-P (as using answer as starting point).

**Human Evaluation Results** Table 5 presents the results of various methods in generating distractor sets for the MCQ cloze tasks. Upon analysis, we observe that in terms of Relevance, Distractiveness, and Utility, both RAP and KAG scored relatively high and exceeded the previous soft method, T5 Candidate Augmentation.

These results suggest that we have successfully guided our model to leverage additional knowledge to generate higher quality distractors. The reason why RAP outperforms KAG might be that the KG (ConceptNet) cannot encompass knowledge from all natural domains, whereas RAP, having access to...
Table 4: RAP Performance Overview with In-Domain and Cross-Domain Study.

<table>
<thead>
<tr>
<th>Method</th>
<th>P@1</th>
<th>R@1</th>
<th>F1@3</th>
<th>MRR</th>
<th>NDCG@3</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BART off-shelf None</td>
<td>14.28</td>
<td>4.76</td>
<td>11.45</td>
<td>21.49</td>
<td>23.70</td>
</tr>
<tr>
<td>RAP-S_BART MCQ-Train</td>
<td>16.21</td>
<td>5.4</td>
<td>11.96</td>
<td>22.2</td>
<td>23.99</td>
</tr>
<tr>
<td>RAP-P_BART MCQ-Train</td>
<td>18.14</td>
<td>6.04</td>
<td>12.35</td>
<td>24.06</td>
<td>25.78</td>
</tr>
<tr>
<td>RAP-P_BART /w GTD MCQ-Train</td>
<td>15.05</td>
<td>5.01</td>
<td>11.58</td>
<td>20.65</td>
<td>22.33</td>
</tr>
<tr>
<td>RAP-S_BART Sciq-all</td>
<td>15.05</td>
<td>5.01</td>
<td>13.25</td>
<td>22.77</td>
<td>25.44</td>
</tr>
<tr>
<td>RAP-P_BART Sciq-all</td>
<td>20.84</td>
<td>6.94</td>
<td>15.57</td>
<td>28.57</td>
<td>30.85</td>
</tr>
<tr>
<td>RAP-P_BART /w GTD Sciq-all</td>
<td>22.77</td>
<td>7.59</td>
<td>17.88</td>
<td>31.33</td>
<td>33.97</td>
</tr>
<tr>
<td>T5 off-shelf None</td>
<td>18.53</td>
<td>6.17</td>
<td>11.45</td>
<td>23.61</td>
<td>25.08</td>
</tr>
<tr>
<td>RAP-S_T5 MCQ-Train</td>
<td>23.93</td>
<td>7.97</td>
<td>14.15</td>
<td>29.27</td>
<td>30.72</td>
</tr>
<tr>
<td>RAP-P_T5 MCQ-Train</td>
<td>22.39</td>
<td>7.46</td>
<td>14.8</td>
<td>29.02</td>
<td>30.72</td>
</tr>
<tr>
<td>RAP-P_T5 /w GTD MCQ-Train</td>
<td>21.62</td>
<td>7.2</td>
<td>13.77</td>
<td>27.15</td>
<td>28.70</td>
</tr>
<tr>
<td>RAP-P_T5 Sciq-all</td>
<td>25.86</td>
<td>8.62</td>
<td>15.57</td>
<td>31.46</td>
<td>33.12</td>
</tr>
<tr>
<td>RAP-P_T5 /w GTD Sciq-all</td>
<td>31.66</td>
<td>10.55</td>
<td>18.91</td>
<td>37.7</td>
<td>39.43</td>
</tr>
<tr>
<td>Sciq</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BART off-shelf None</td>
<td>19.3</td>
<td>6.46</td>
<td>14.94</td>
<td>26.40</td>
<td>28.36</td>
</tr>
<tr>
<td>RAP-S_BART Sciq-train</td>
<td>19.3</td>
<td>6.45</td>
<td>14.66</td>
<td>25.61</td>
<td>27.45</td>
</tr>
<tr>
<td>RAP-P_BART Sciq-train</td>
<td>20.1</td>
<td>6.73</td>
<td>16.32</td>
<td>27.56</td>
<td>29.68</td>
</tr>
<tr>
<td>RAP-P_BART /w GTD Sciq-train</td>
<td>22.4</td>
<td>7.48</td>
<td>16.39</td>
<td>28.9</td>
<td>30.81</td>
</tr>
<tr>
<td>RAP-S_T5 MCQ-all</td>
<td>20.4</td>
<td>6.83</td>
<td>15.65</td>
<td>27.19</td>
<td>29.09</td>
</tr>
<tr>
<td>RAP-P_T5 MCQ-all</td>
<td>20.2</td>
<td>6.75</td>
<td>15.58</td>
<td>27.41</td>
<td>29.5</td>
</tr>
<tr>
<td>RAP-P_T5 /w GTD MCQ-all</td>
<td>20.7</td>
<td>6.91</td>
<td>16.06</td>
<td>28.16</td>
<td>30.32</td>
</tr>
<tr>
<td>T5 off-shelf None</td>
<td>24.27</td>
<td>8.09</td>
<td>15.92</td>
<td>30.32</td>
<td>31.99</td>
</tr>
<tr>
<td>RAP-S_T5 Sciq-train</td>
<td>23.6</td>
<td>7.86</td>
<td>15.62</td>
<td>29.13</td>
<td>30.69</td>
</tr>
<tr>
<td>RAP-P_T5 Sciq-train</td>
<td>24.3</td>
<td>8.09</td>
<td>16.06</td>
<td>29.98</td>
<td>31.49</td>
</tr>
<tr>
<td>RAP-P_T5 /w GTD Sciq-train</td>
<td>25.0</td>
<td>8.34</td>
<td>16.46</td>
<td>30.18</td>
<td>31.54</td>
</tr>
<tr>
<td>RAP-S_T5 MCQ-all</td>
<td>24.1</td>
<td>8.04</td>
<td>15.26</td>
<td>29.63</td>
<td>31.12</td>
</tr>
<tr>
<td>RAP-P_T5 MCQ-all</td>
<td>23.1</td>
<td>7.71</td>
<td>16.31</td>
<td>29.56</td>
<td>31.49</td>
</tr>
<tr>
<td>RAP-P_T5 /w GTD MCQ-all</td>
<td>23.7</td>
<td>7.89</td>
<td>16.36</td>
<td>29.73</td>
<td>31.25</td>
</tr>
</tbody>
</table>

Table 5: The human evaluation for MCQ dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Relevance</th>
<th>Distractiveness</th>
<th>Utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Wang et al., 2023)</td>
<td>3.81</td>
<td>3.35</td>
<td>3.55</td>
</tr>
<tr>
<td>KAG(T5)</td>
<td>4.05</td>
<td>3.25</td>
<td>3.77</td>
</tr>
<tr>
<td>Text2text(RAP-T5)</td>
<td>4.45</td>
<td>3.87</td>
<td>4.07</td>
</tr>
<tr>
<td>ChatGPT</td>
<td>3.14</td>
<td>2.67</td>
<td>3</td>
</tr>
<tr>
<td>ground truth</td>
<td>3.52</td>
<td>3.02</td>
<td>3.4</td>
</tr>
</tbody>
</table>

4.5 Ablation Study on KAG

We conducted ablation experiments on the MCQ dataset to assess the impact of different components in our system. The results are summarized in Table 6, providing insights into the importance and contributions of each component.

Candidate Generation Generating candidate sets using Masked LM or Generative LM is a key component in our system, enhancing KG retrieval. Removing these sets and relying solely on sentence and answer-based triplet retrieval significantly reduced the number of retrievable triplets, potentially introducing irrelevant noise and resulting in a performance drop (F1@3: 16.47 to 13.64). Using Masked LM or Generative LM alone also affected F1@3. This emphasizes the value of incorporating candidate sets, as they provide crucial information
for identifying important triplets in the knowledge graph.

**Reranker** The Reranker, our second key component, sorts triplets to select those more relevant to distractors/answers. Without sorting by the Reranker, randomly selecting triplets for triplet augmentation led to a decrease in F1@3 (16.47 to 14.15). This highlights the Reranker’s importance in identifying important triplets and improving downstream task performance.

### 4.6 Study on Cross-Domain for RAP

In this subsection, we report our studies on cross-domain and in-domain for RAP. The experiment results are summarized in Table 4.

We have the following observations: Firstly, in terms of cross-domain effects (using the MCQ dataset for pretraining and fine-tuning the DG model with the Sciq dataset, or vice versa), we found that using Sciq-all for pretraining the language model resulted in a significant improvement in the downstream MCQ dataset for distractor generation. Specifically, using the T5 model, the RAP-based approach increased the F1@3 score from 11.45 to 18.91 (almost a 1.5-fold improvement). This notable improvement was expected, considering that Sciq is a larger dataset compared to MCQ. However, when we reversed the process and used MCQ for pretraining, followed by testing with the Sciq dataset, the improvement in performance was not as significant. Taking the BART model as an example, the direct use of BART off-shelf achieved a score of 14.94, while RAP-S\_BART only improved it to 16.06. Although there was some improvement, it was not as substantial as the effects observed when using Sciq pretraining.

Regarding in-domain effects (using the same dataset for both pretraining and fine-tuning), we found consistent improvements in the generated results. For example, using the T5 model on the Sciq test dataset, the T5 off-shell score was 15.92, while RAP-P\_T5 improved it to 16.46. We also observed similar improvements in the MCQ dataset.

In conclusion, our proposed RAP approach has shown improved generation performance from both data augmentation and task-specific pretraining perspectives. The experiments demonstrate that incorporating RAP into the training process leads to significant enhancements in distractor generation.

## 5 Conclusion

In this paper, we introduce the utilization of task-specific pretraining and knowledge base. Our experimental results highlight a significant performance improvement achieved through the integration. Our work represents a novel contribution to the field of distractor generation for MCQs, showcasing the potential of combining retrieval augmented pretraining and KGs to achieve superior results.

## 6 Limitations

We report the following limitations for the KAG method:

- Combining KAG with RAP did not yield additive effects. We believe that the key factor lies in that the knowledge triplets may introduce additional noise that interferes with the Text2Text model’s generation process. Thus, there is significant room for improvement in the current design of the triplet ranker. Selecting relevant triplets for generation is an aspect that requires further strengthening in this research.

- The current evaluation and training heavily rely on token scores, which can only reflect the similarity to the ground truth but cannot fully represent the quality of the generated output. The research conclusions thus far have also been established solely based on token scores.

- During the pretraining stage of RAP, when retrieving sentences/passages, if the answer is a rare or specialized term, the external corpus may fail to find matching sentences/passages. Therefore, when applying the RAP framework to other knowledge domains, having an abundant knowledge corpus becomes crucial.

## Acknowledgement

This work is supported by NSTC 112-2634-F-005 -002-project Smart Sustainable New Agriculture Research Center (SMARTer), NSTC Taiwan Project under grant 112-2221-E-005 -075 -MY3, and Delta Research Center, Delta Electronics, Inc.

## References


Appendix

11028
A Implementation Details

Our models are implemented based on models from Hugging Face (Wolf et al., 2019). For generating candidate set, we use an language model trained for distractor generation from (Chiang et al., 2022) and gpt3.5 turbo through ChatGPT API with zero-shot prompting. For triplet reranker, we use Sentence-BERT (Reimers and Gurevych, 2019) to estimate the initial sentence embedding distance. For triplet classification, we use Sentence-BERT and BERT (Devlin et al., 2018) as the default PLM. For the triplet augmented generator, we experiment with BART (Lewis et al., 2019) and T5 (Raffel et al., 2020) as base generation models.

During training, we use AdamW as the optimizer and an initial learning rate of 2e-5 for BERT, BART and 1e-4 for T5 models. All experiments are conducted using two NVIDIA TITAN RTX GPUs.

**BART-based generator** With MCQ data, the maximum number of epochs is set to 40 with a batch size of on two NVIDIA TITAN RTX GPUs for the distraction generation with triplet augmentation with a batch size of 32 and a maximum number of triplets is set to 50. With Sciq data, the maximum number of epochs is set to 50 with a batch size of 32 on two NVIDIA TITAN RTX GPUs for the distraction generation with triplet augmentation with a batch size of 32 and a maximum number of triplets is set to 50. The average running time for BART-based generators is 30 minutes (1.5 hours) on MCQ (Sciq).

**T5-based generator** With MCQ data, the maximum number of epochs is set to 40 with a batch size of on two NVIDIA TITAN RTX GPUs for the distraction generation with triplet augmentation with a batch size of 32 and a maximum number of triplets is set to 50. With Sciq data, the maximum number of epochs is set to 50 with a batch size of 32 on two NVIDIA TITAN RTX GPUs for the distraction generation with triplet augmentation with a batch size of 32 and a maximum number of triplets is set to 50. The average running time for BART-based generators is 40 minutes (3 hours) on MCQ (Sciq).

**Knowledge Graph** We utilize ConceptNet (Speer et al., 2017) as our KG, which is a graph that encompasses a wide range of knowledge domains. The KG consists of a total of 800k nodes and 2M edges. For each question’s KG retrieval set, we pre-process the retrieval subgraph using the approach described in (Feng et al., 2020). Specifically, we process the textual corpus of each question, including the question text, answer options, and entities in the candidate set.