

KEPR: Knowledge Enhancement and Plausibility Ranking for Generative Commonsense Question Answering

Zhifeng Li

School of Computer Science and Technology
Soochow University, Suzhou, China
li_2hifeng@outlook.com

Yifan Fan

School of Computer Science and Technology
Soochow University, Suzhou, China
yifanfannlp@gmail.com

Bowei Zou

Institute for Infocomm Research
A*STAR, Singapore
zou_bowei@i2r.a-star.edu.sg

Yu Hong✉

School of Computer Science and Technology
Soochow University, Suzhou, China
tianxianer@gmail.com

Abstract—Generative commonsense question answering (GenCQA) is a task of automatically generating a list of answers given a question. The answer list is required to cover all reasonable answers. This presents the considerable challenges of producing diverse answers and ranking them properly. Incorporating a variety of closely-related background knowledge into the encoding of questions enables the generation of different answers. Meanwhile, learning to distinguish positive answers from negative ones potentially enhances the probabilistic estimation of plausibility, and accordingly, the plausibility-based ranking. Therefore, we propose a Knowledge Enhancement and Plausibility Ranking (KEPR) approach grounded on the Generate-Then-Rank pipeline architecture. Specifically, we expand questions in terms of Wiktionary commonsense knowledge of keywords, and reformulate them with normalized patterns. Dense passage retrieval is utilized for capturing relevant knowledge, and different PLM-based (BART, GPT2 and T5) networks are used for generating answers. On the other hand, we develop an ELECTRA-based answer ranking model, where logistic regression is conducted during training, with the aim of approximating different levels of plausibility in a polar classification scenario. Extensive experiments on the benchmark ProtoQA show that KEPR obtains substantial improvements, compared to the strong baselines. Within the experimental models, the T5-based GenCQA with KEPR obtains the best performance, which is up to 60.91% at the primary canonical metric Inc@3. It outperforms the existing GenCQA models on the current leaderboard of ProtoQA.

I. INTRODUCTION

Great efforts were made on the study of Commonsense Question Answering (CQA) in the area of natural language processing. The conventional CQA tasks can be roughly divided into two categories, including assertion correctness judgment [1] and multiple-choice question answering [2], [3]. Both stimulate the exploration of discriminative CQA methods, as well as supportive neural models. Different from the aforementioned CQA tasks, Generative CQA (GenCQA) produces answers in a generative way, as claimed in the task

TABLE I
A GENCQA CASE WHICH IS SELECTED FROM PROTOQA.

Prototypical Question	Name something that an <i>athlete</i> would not keep in her <i>refrigerator</i> .
Ground-truth answer list	unhealthy food (36): chocolate, junk food, ... unhealthy drinks (24): coke, alcohol, ... clothing/shoes (24): gloves, clothes, shoe, ... accessories (7): handbag, medal, tennis, ...
[BART]	food, ice, vitamins, beer, alcohol
Background Knowledge	<i>athlete</i> : A person who actively participates in physical sports, especially with great skill. <i>refrigerator</i> : A household appliance used for keeping food fresh by refrigeration.

The ground-truth answers are organized into different classes, conditioned on plausibility scores. [BART] corresponds to the generated results by BART-based, GenCQA without considering the knowledge of “*athlete*” and “*refrigerator*”.

definition towards the benchmark ProtoQA [4]. In particular, models necessarily generate diverse answers via commonsense reasoning within a specific prototypical scenario and rank them in terms of plausibility (see the example in Table I).

Commonsense-aware neural models contribute to the solution of CQA problems. The pre-trained generative language models such as GPT2 [5], BART [6] and T5 [7] are knowledgeable, and therefore serve as highly competitive baselines for CQA. However, empirical findings show that they fail to perform perfectly in the GenCQA task, where the answers they produced are of less diversity and generally disordered. For example, within the five answers generated by a fine-tuned BART (in Table I), “*beer*” appears as a redundant case when “*alcohol*” has been generated, and there are merely one class of plausible answers produced, i.e., “unhealthy drinks”. More seriously, the incorrect answers such as “*ice*” and “*food*” are ranked higher than plausible answers.

We suggest that the above issues can be alleviated by the following two solutions. First, the supplementation of exclusive background knowledge of keywords in the questions is beneficial for pursuing a wider range of answers. For

✉Corresponding author.

The research is supported by the National Key R&D Program of China (2020YFB1313601) and the National Science Foundation of China (No.62076174, No.61836007).

example, the nature of “athlete” like “participating in physical sports” (see the row of Background Knowledge in Table I) is informative for sampling the answer “junk food”, during the evidential reasoning process. Similarly, the attributes of “refrigerator” are useful for inferring the answer classes “clothing/shoes” and “accessories”. Second, it is crucial to construct a separate ranking mechanism conditioned on the probabilistic estimation of plausibility, instead of the direct use of beam search at the decoding stage.

Accordingly, we propose a Knowledge Enhancement and Plausibility Ranking (abbr., KEPR) approach¹ for the GenCQA task. KEPR is grounded on the generate-then-rank [8] framework, which decouples the answer generation and ranking tasks. For answer generation, we enhance the PLM-based generative model by incorporating commonsense knowledge of keywords into the modeling process, where Wiktionary² is used. A series of easy-to-implement assistive technologies are applied, including statistical keyword extraction, dense passage retrieval and pattern-based question rewriting. For answer ranking, we construct an ELECTRA-based [9] ranker which estimates the plausibility of answers with logistic regression. The ranker is separately trained in a binary classification scenario using the most plausible answers and randomly-selected negative cases. Besides, answer deduplication is used.

We experiment on the benchmark corpus ProtoQA, and the experimental results demonstrate that KERP is effective and multi-model compatible. The main contributions of this paper are as follows:

- We propose a novel method, i.e., KEPR, to enhance GenCQA, where knowledge enhancement and plausibility ranking are used. KEPR produces considerable improvements at all canonical metrics, compared to GPT2, BART and T5, as well as their robust versions.
- When cooperating KEPR with BART, a relatively-weak PLM, we achieve a comparable performance to the state-of-the-art GenCQA model [10] for Inc@3 (55.38% vs 55.77%) but with a smaller model size.
- When cooperating KEPR with a strong PLM like T5-3B, we achieve the best performance, where the Inc@3 score is up to 60.91%.

II. RELATED WORKS

Generative CQA Ma et al. [11] use fine-tuning, auto-prompting and prefix-tuning to enhance PLM-based GenCQA models. Their systematical analysis reveals that fine-tuning leads to the forgetting of global knowledge gained during pre-training. This is unavoidable due to the representational approximation or even overfitting to the task-specific data. Luo et al. [10] develop a popularity-aware answer ranker. It is not only used for re-ranking the generated answers in practice, but serves as a guider to refine the generator for producing typical answers. Policy gradient based rewarding is utilized to reinforce the training of answer generator.

Commonsense Generation The studies of concept-oriented commonsense description generation on CommonGen [12] have been widely conducted. They can be safely considered as references because the ProtoQA [4] and CommonGen projects share the same goal, i.e., verifying linguistic intelligence from the perspective of generative commonsense reasoning. In this field, research effort has been devoted to the incorporation of external knowledge into generative language models. Specifically, EKI-BART [13] retrieves prototypes from in-domain and out-of-domain corpora, and utilizes them as the scenario knowledge to guide the generation process. KFCNet [14] acquires high-quality prototypes and applies contrastive learning to constrain both intrinsic representations of encoder and extrinsic representations of decoder. KG-BART [15] utilizes a knowledge graph to augment BART for text generation, where concept relationships are incorporated. Moreover, it applies graph attention for semantic enrichment.

Discriminative CQA It is noteworthy that a variety of discriminative CQA models [1], [2], [16] leverage external knowledge to implement the deeper reasoning. A wider range of knowledge bases is considered, such as ConceptNet [17], Cambridge Dictionary [18], Open Mind Common Sense [19] and Wiktionary [20]. Unfortunately, such sophisticated techniques cannot be transferred to the generative CQA task. It is because that they heavily rely on the exposed answer candidates for retrieving and modeling relevant external knowledge. Instead, all possible answers in GenCQA aren’t given, but on the contrary, they need to be generated eventually.

III. APPROACH

The overall architecture of KEPR-based GenCQA model is shown in Fig. 1. The considered PLM in this case is BART. It can be replaced by other PLMs, such as GPT2 and T5. The GenCQA model consists of three components, including knowledge-oriented retriever, answer generator and ranker.

Retriever first extracts keywords from the question q using statistical information. Dense passage retrieval is further used to acquire relevant knowledge $K = \{k_1, k_2, \dots, k_m\}$ of the keywords from Wiktionary, where k_i denotes the knowledge item of the i -th keyword. Each knowledge item describes the nature of a keyword in a sentence.

Generator learns to generate different answers $A = \{a_1, a_2, \dots, a_n\}$ conditioned on the question q and relevant knowledge items. The BART-based encoder is used for knowledge-aware semantic feature representation. The accompanying decoder infers the possible answers in terms of the features. Question rewriting and answer deduplication are used to assist the generator.

Ranker learns to rate the generated answers with the real-valued plausibility scores. Each score is normalized and signals the probability that a certain answer is commonly recognized to be reasonable. The ranker is constructed with ELECTRA. It is separately trained to compute plausibility scores, where logistic regression is conducted within the binary classification scenario of truly-plausible instances and negative ones.

¹ Access code and appendix via <https://github.com/Zaaachary/CSQA-KEPR>

² <https://www.wiktionary.org/>

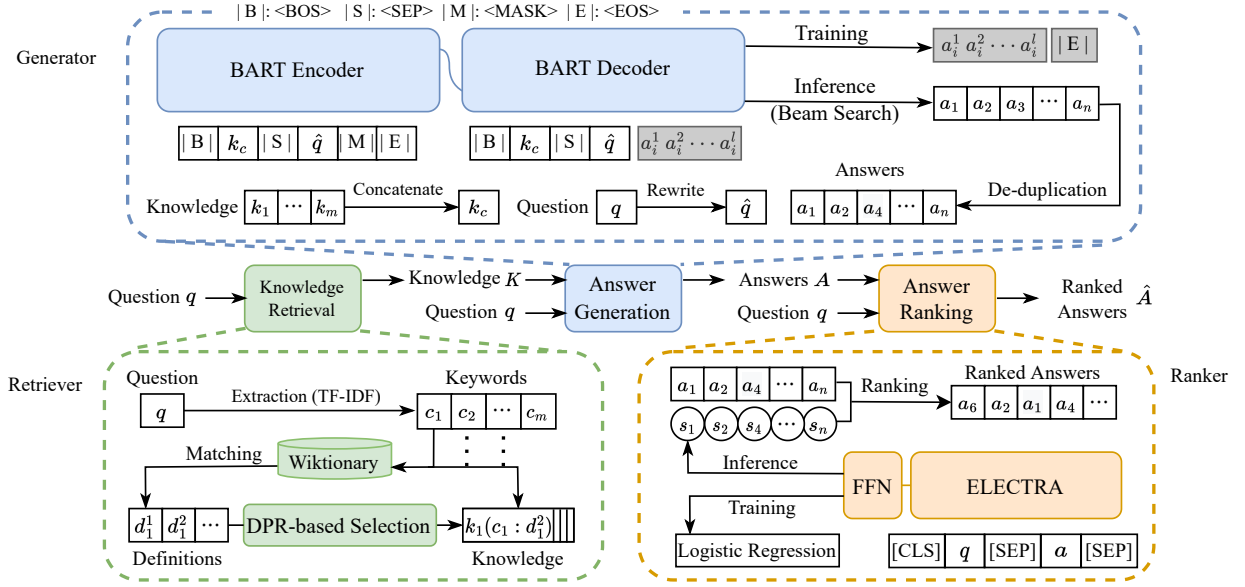


Fig. 1. The overall architecture of KEPR. The grey blocks in Answer Generation show the input and output of each generation process, where a_i^l denotes the l -th token of the i -th answer of the given question.

A. Knowledge Acquisition (Retriever)

We retrieve knowledge items using the keywords of the question q as queries. The large-scale online dictionary Wiktionary is used as the knowledge source, which contains about 7M English sentence-level knowledge items. We extract the top- m keywords which, statistically, are of higher Term Frequency-Inverse Document Frequency (TF-IDF) [21]. For each token t_j in the question q , we estimate TF in terms of the occurrence frequency of t_j in q , while IDF is calculated using the non-repetitive occurrence frequency of t_i in all the questions in the training set. We lemmatize the keywords by NLTK, and conduct string matching between each keyword and Wiktionary entries to acquire the corresponding knowledge item.

However, multiple knowledge items can be obtained for each keyword. It is because that, in Wiktionary, one entry (e.g., a polysemous word) generally corresponds to multiple knowledge items, which specify different definitions and natures. In order to obtain the knowledge item that solely applies to the question q , we use the Dense Passage Retriever (DPR) [22] to perform context-aware text matching. Specifically, we reformulate the query by the fixed prototype as below: “What is the meaning of word <keyword> in the sentence <question>?”. The slots “<keyword>” and “<question>” can be dynamically filled with concrete instances. On the basis, we use DPR to respectively encode the query and each candidate knowledge item. This allows their sentence-level embeddings to be produced. DPR further calculates the relevant score over embeddings using the scalar product. The knowledge item that holds the highest relevance score will be adopted, while the rest will be abandoned.

Accordingly, we construct an exclusive knowledge set $K = \{k_1, k_2, \dots, k_m\}$ for each question q , where k_i is the concatenation of i -th keyword and its relevant knowledge item.

B. Knowledge-aware Answer Generation (Generator)

1) *Question Rewriting*: We rewrite the questions to produce Cloze-style equivalents that contain the masked suffixes. We provide an example below, where the sequence in orange font serves as the prefix which is rewritten, while the sequence in green font appears as the main body text which isn’t falsified.

Original Question: *Name something that an athlete would not keep in her refrigerator?*

Rewritten Question: *One thing that an athlete would not keep in her refrigerator is <MASK>?*

The rewritten questions, ideally, are PLM-friendly because text infilling (masked span prediction) is generally considered as the task for pre-training. A series of predefined patterns are utilized for rewriting questions. They can be easily ascertained and invoked by looking up the prefix-pattern mapping table, as shown in Appendix A.

2) *Knowledge-enhanced Generation*: To demonstrate the input and output of our knowledge-enhanced answer generator, we chose BART as an example here, which is based on the encoder-decoder architecture. The input of BART encoder is constructed by concatenating knowledge items with the rewritten question, in the format as below:

$$\langle \text{BOS} \rangle k_c \langle \text{SEP} \rangle \hat{q} \langle \text{MASK} \rangle \langle \text{EOS} \rangle \quad (1)$$

where, k_c denotes the concatenated sentence-level knowledge items, i.e., the ones in K , while \hat{q} denotes the rewritten question. The input of BART decoder is constructed in the similar way except that the special tokens <MASK> and <EOS> are pruned, since they are the generation target of decoder.

We also consider GPT2 and T5 for answer generation in our experiments. GPT2 is a decoder-only model that performs unidirectional autoregressive generation. When using GPT2, we adopt the decoder methodology used by BART, which

generates answers by predicting the next token based on the knowledge and question. T5 is structurally similar to BART, and we use it in a way that is similar to BART, except that T5 uses a special token, $\langle \text{extra_id} \rangle$, to mask the answer.

3) *Training Stage*: We train the generator in the teacher-forcing manner [23], where the truly-correct preceding context (i.e., $t-1$ ground-truth tokens) are necessarily exposed to the generator when the t -th token is being predicted. The training objective is to maximize the global transition probability P_θ as below:

$$\begin{aligned} \theta &= \operatorname{argmax}_{\theta} \sum_{q_i \in Q} \sum_{a_j \in A_i} P_\theta(a_j | \hat{q}_i, \mathbf{K}_i) \\ P_\theta(a_j | \hat{q}_i, \mathbf{K}_i) &= \sum_{t=1}^{\text{len}} P_\theta(a_j^t | \hat{q}_i, \mathbf{K}_i) \end{aligned} \quad (2)$$

where θ denotes all the parameters of the generator. \mathbf{K}_i is the knowledge set of the i -th question q_i , while a_j^t is the t -th token of the j -th answer of q_i .

4) *Inference Stage*: We sample a fixed number of answer candidates for each question during inference stage, where beam search [24] is used. By beam search, the answers which are of higher confidence will be adopted. The sum of logarithmic generation probabilities of tokens is used for estimating the confidence. Note that there might be certain different forms of the same answer in the list. To address the duplication issue and obtain diverse answers that possess different senses, we conduct dictionary-based deduplication. Firstly, we remove stop words from the answer candidates and perform lemmatization, so as to convert them into content words which are of original morphology. For example, given the list of answer candidates like “a bike”, “her bikes” and “the bicycle”, we convert them into “bike” and “bicycle”. On this basis, we identify the answer candidates which are thoroughly consisted of synonyms. The NLTK is used for determining synonymy conditioned on WordNet [25]. Given a group of synonymous candidates, we merely retain the one holding the highest confidence score and abandon the rest.

C. Plausibility-based Ranking (Ranker)

The canonical GenCQA evaluation system additionally evaluates the rankings of the generated answers. Accordingly, if the manually-designated high-ranking answer classes are actually ranked lower, the GenCQA performance will be considered to be less promising. Therefore, we build a ranker to arrange the generated answers in the order of plausibility. We approximate the plausibility by estimating the probability that an answer derives from the class of absolutely positive instances. We separately train a binary classification model (Positive versus Negative), and use its discriminative layer along with the Sigmoid activation function to compute the probabilistic plausibility score.

1) *Training Data Collection*: In order to train the classifier, we construct a training set containing positive and negative instances. All the instances are collected from the ProtoQA training set itself, without using external data. For each question in the ProtoQA training set, we select top- n highly-weighted

answers as the absolutely positive instances. The weight is given as the ground truth in ProtoQA, and it is equivalent to the proportion of “yes” vote towards the plausibility of a certain answer category.

Given a question, we collect negative instances from the answers of other questions, where random sampling is used. To ensure the absolute implausibility, we verify whether the sampled instances are synonymous or contain synonyms with the ground-truth answers. Such cases will be abandoned. In our experiments, we set n to 2, and thus obtain about 16.5K QA pairs of 8,782 positive cases and the same number of negative cases. They are used for training. In the same way, we collect 3.5K instances to build the validation set.

2) *Plausibility Approximation*: The binary classification model is constructed with the ELECTRA-based encoder [9] and a fully-connected linear layer with Sigmoid. We simply use the latter as the discriminator. For a question q and one of the generated answers a , we concatenate them to form the input of ELECTRA encoder: “[CLS] q [SEP] a [SEP]”. Given the output of ELECTRA, we take the encoded global representation [CLS], and feed it into the discriminator. On this basis, we use the linear layer to project [CLS] into the 1-dimensional vector $v \in \mathbb{R}$ that represents the plausibility level. Sigmoid function is further used to activate v , producing a probabilistic value \hat{v} . We sort all the generated answers of q in terms of \hat{v} . The whole classifier is trained using the aforementioned training set (Section III.C.1), in the manner of logistic regression. Minimizing the binary cross-entropy loss is considered as the objective:

$$\mathcal{L}(q, a) = - \sum y \cdot \log(\hat{v}) + (1 - y) \cdot \log(1 - \hat{v}) \quad (3)$$

Through the knowledge-enhanced answer generation, we deliberately retain 12 answers in total for each question. The number (12) is larger than the specified maximum amount (10) of answers in the task of GenCQA. In practice, we utilize the Ranker to evaluate the plausibility of every generated answer and arrange them based on their plausibility, subsequently eliminating the two answers that rank the lowest.

IV. EXPERIMENTATION

A. Corpus and Datasets

We carry out experiments on ProtoQA [4], a benchmark corpus of GenCQA. ProtoQA comprises a training set and a test set as usual, though it provides two validation sets, namely VSet1 and VSet2 for short. VSet1 (also known as scraped Dev) is built in the same way as the training set, where the answers of each question are (1) collected by questionnaire survey, (2) weighted by voting scores and (3) purified by automatic deduplication, without being double-checked and classified. By contrast, the questions in both VSet2 (also known as crowd-sourced Dev) and test set are answered by annotators from multiple perspectives, and the answers are carefully verified, ranked and classified. The statistics are shown in Table II.

We train our KEPR-based GenCQA model over the sole training set, and develop it on VSet1. In the self-contained

TABLE II
THE STATISTICS OF PROTOQA DATASETS.

Data Split	Questions	Answers*
Training set	8,782	5.13
Test set	102	/
VSet1	4,963	5.06
VSet2	52	10.40

*The average answer number per question

experiments such as that in the ablation study, we follow the common practice [10], [11] to test the model in VSet2. When comparing to the previous work in the main test, we report the performance released on the leaderboard³ of ProtoQA. It is obtained on the test set, where the questions are accessible, but the answers are undisclosed. Note that the evaluation process is implemented by the ProtoQA-evaluator, grounded on the submitted answers towards the disclosed questions.

B. Evaluation Metrics

We apply the official toolkit ProtoQA-evaluator to assess the GenCQA models. There are two kinds of metrics used for evaluation, including $\text{Ans}@k$ and $\text{Inc}@k$ [4]. Both calculate the weighted accuracy ($\check{\text{Acc}}$) given a certain truncation method. $\check{\text{Acc}}$ belongs to a task-specific evaluation scheme. It rewards a GenCQA model with higher scores if the popular-in-voting answers can be generated and highly ranked. We detail the calculation method of $\check{\text{Acc}}$ in Appendix B.

The calculation of $\check{\text{Acc}}$ in $\text{Ans}@k$ and $\text{Inc}@k$ is different from each other due to inconsistent truncation methods. For $\text{Ans}@k$, the answer list is truncated at the k -th answer, and the lower-ranked answers won't be considered for evaluation. For $\text{Inc}@k$, it is truncated at the k -th incorrect answer. We follow the canonical evaluation scheme of ProtoQA to set k to 1, 3, 5 and 10 respectively in the separate experiments. Within all the metrics, $\text{Inc}@3$ is appointed as the most critical one, which accords with the tolerance of general users for errors. It is used as the gold standard for rating different submitted models on ProtoQA leaderboard.

C. Implementation Details

To speed up knowledge acquisition, we organize English Wiktionary dump into a hash map. The number of keywords used for retrieval is set to 2. During training, we optimize all our models with AdamW, where ϵ is set to $1e-6$. Both generator and ranker are trained for one epoch with a batch size of 8. We respectively use GPT2, BART and T5 to construct different generators. All of them share the same hyperparameter settings, where the learning rate is set to $1e-5$, and the warmup proportion is set to 0.025. When training the binary classifier for plausibility approximation, we set the learning rate to $3e-6$, while the warmup proportion is set to 0.01.

D. Compared Models

We utilize the fine-tuned GPT2 and BART as the baselines. The performance of GPT2 is reported in the pilot study of GenCQA on ProtoQA [4]. The performance of BART is reported on the ProtoQA leaderboard, which is accompanied with the easy-to-follow instructions of reproducing and fine-tuning prototypical BART [26]. In addition, we reproduce and enhance both GPT2 and BART to form their robust versions (GPT2-Robust and BART-Robust), where data cleaning is conducted on the training data (e.g., segmentation for compound answers and noise filtering), and hyperparameters are reset for model selection. Besides, the answer deduplication is used to diversify the output of GPT2 and BART.

We compare with Team Cosmic's work [27] (T5-11B + Ranker), where the strong T5-11B possessing 11 billions of trainable parameters is used, and it is coupled with a ranker. We construct a T5-3B baseline (T5-3B-Robust). It is a robust version due to the use of data cleaning, fine-tuning and answer deduplication. In addition, SUDA NLP's work [28] (GPT3-175B Few-shot) is considered for comparison, which indicates the reasoning ability of large-scale language model.

Besides, we compare our models to a series of strong arts that use external knowledge, including:

- BART+DPR [29] which use DPR to build retriever. It acquires knowledge of relevant contexts from an external data source. The generator is built using BART.
- GPT2+ConceptNet [30] uses GPT2 to generate answers. It retrieves commonsense facts from ConceptNet [31], and uses them to augment the input representations. DPR is also leveraged for retrieving relevant concepts in this art.
- T5-large+WordNet+Ranker [10] uses a large T5 network. It retrieves definition descriptions of keywords from WordNet [25], and incorporates them into the encoding process. Its ranker is constructed using DeBERTa [32].

E. Main Results

The performance of all the GenCQA models is shown in Table III. It can be observed that KEPR yield substantial improvements, compared to the baselines and their robust versions. In addition, all the KEPR-based models outperform the state-of-the-art (SoTA) models which utilize the same PLMs. The performance gain is no less than 4.4%, 11.5% and 5.4% at $\text{Inc}@3$ (gold standard) when GPT2, BART and T5-based SoTA are respectively taken for comparison.

More importantly, our BART-KEPR achieves the comparable performance at $\text{Inc}@3$, compared to T5-11B and T5-large based GenCQA models. Though, BART-KEPR merely contains 741M trainable parameters in total. By contrast, the T5-large generator contains 770M parameters, and its accompanying ELECTRA-based ranker additionally possesses 390M parameters. Besides, T5-11B obviously uses a huge number of parameters. Therefore, we suggest that our BART-KEPR is vest-pocket and relatively practicable. It can be adopted by a potential user who intends to set up a baseline but lacks computational power.

³<https://leaderboard.allenai.org/protoqa/submissions>

TABLE III
PERFORMANCE COMPARISON ON THE OFFICIAL (ONLINE) TEST SET OF PROTOQA.

Method	Inc@3	Inc@1	Inc@5	Ans@1	Ans@3	Ans@5	Ans@10
GPT2 (baseline) [5]	41.68	26.09	48.16	36.35	44.42	46.63	53.52
GPT2 + ConceptNet [30]	41.23	26.32	48.60	36.76	41.06	45.84	55.28
GPT3-175B Few-shot [28]	47.63	35.76	53.53	47.08	50.11	51.93	56.94
GPT2 - Robust (reproduced)	50.50	31.86	55.30	49.89	49.75	51.71	60.14
GPT2 - KEPR (ours)	51.93	33.16	60.33	41.61	49.84	54.83	62.61
BART (baseline) [6]	43.96	25.62	50.19	34.45	44.63	48.29	54.89
BART + DPR [29]	41.69	26.64	46.96	32.68	43.61	45.90	51.54
BART - Robust (reproduced)	51.69	35.46	57.28	45.52	49.95	53.03	61.60
BART - KEPR (ours)	55.38	37.83	62.14	47.24	56.28	57.42	64.96
T5-11B + Ranker (baseline) [27]	54.15	43.59	55.59	56.00	60.98	60.66	56.15
T5-large + WordNet + Ranker [10]	55.77	40.81	60.30	54.20	57.00	58.59	64.97
T5-3B - Robust (reproduced)	57.05	38.55	64.12	55.87	58.28	62.65	66.44
T5-3B - KEPR (ours)	60.91	45.72	67.87	60.52	61.14	62.13	69.42

The weighted accuracy \hat{Acc} (%) is used for evaluation, conditioned on two truncation schemes, i.e., Inc@ k and Ans@ k . We follow the common practice to use Inc@3 as the gold standard for rating different GenCQA models.

F. Ablation Study

We carry out ablation experiments grounded on GPT2, BART and T5-3B over the secondary validation set VSet2 (i.e., the offline test set). Reverse ablation is conducted, where Knowledge Enhancement (KE) and answer Plausibility Ranking (PR) are coupled with the above PLM generators, respectively and independently. Further, they are jointly used (i.e., KEPR). Fig. 2 shows the experimental results, where the performance curves are plotted over the successive truncation points ($k \in [1, 10]$) for both Ans@ k and Inc@ k . The experimental results show that, in summary, the independent use of either KE or PR obtains minor improvements. By contrast, the joint use of them (i.e., KEPR) yields much more substantial improvements at all the metrics for every PLM baseline.

Another interesting finding is that the Ans@ k curve of KEPR in Fig. 2 rises steeply no matter whether k equals to the minimum value ($k=1$) or largest value ($k=10$). This demonstrates that, on one hand, KEPR helps to push highly-weighted (i.e., popular in voting) answers up to the top of an answer list. On the other hand, it contributes to rolling a larger number of highly-weighted answers into a longer answer list. Though, this advantage cannot be absolutely attributed to KE, despite the performance curves of KEPR and KE seem to fluctuate synchronously in most cases. In fact, both KE and PR probably cause the effect, and each of them is indispensable. This can be demonstrated by the following findings:

- When BART is considered as the baseline, PR causes performance degradation at Inc@5 ($k=5$). On the contrary, KE yields a relatively substantial improvement. It ensures the effectiveness of KEPR at Inc@5.
- When GPT2 is regarded as the baseline, KE results in performance reduction at Inc@1 ($k=1$). By contrast, PR obtains a significant improvement. Similarly, it allows KEPR to have a positive effect at Inc@1.

It can be additionally observed that both KE and PR cause performance reduction at Ans@5 ($k=5$) no matter which PLM is used as the baseline. It is because that PLM baselines overfit the mostly short answer lists in the training and validation

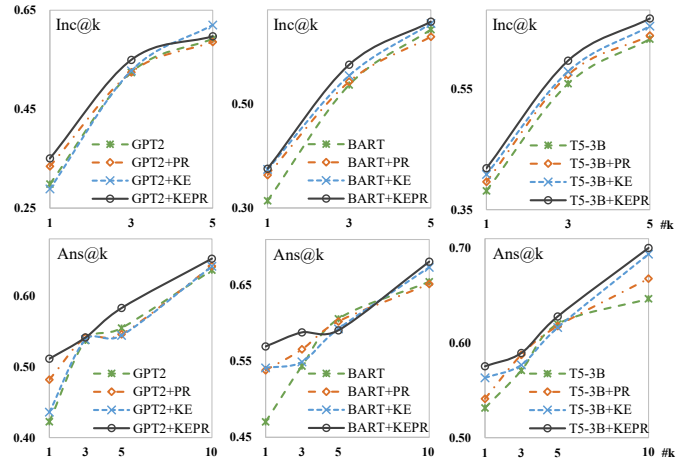


Fig. 2. Ablation study for GPT2, BART and T5-3B on VSet2. The detailed performance metrics of each model are provided in Appendix C.

sets. Specifically, the length of ground-truth answer lists in the training set is 5.13 on average, while 5.06 in the validation set VSet1. By contrast, the average length of answer lists in VSet2 is about 10.40. The PLMs are trained to attain the most perfect states over the shorter answer lists, though uncertainty is increased when a longer answer list needs to be generated during test. By contrast, when KE and PR are used, the whole models (PLMs+KE and PLMs+PR) can be trained to converge to the globally optimal solutions on the longer answer lists, instead of locally optimal solutions. They obviously adapt to the test data better, although this is at the expense of performance reduction for the top-5 answers.

It is noteworthy that, during the training of PLM baselines, arbitrarily enlarging the maximum length of the generated answer lists is ineffective. It is because that the length of ground-truth answer lists is fixed, and thus the estimated cross-entropy loss (between the generated answers and ground truth) for backpropagation will not be changed significantly, along with the modification of maximum length. By contrast, the effects of KE and PR are active when the maximum length is

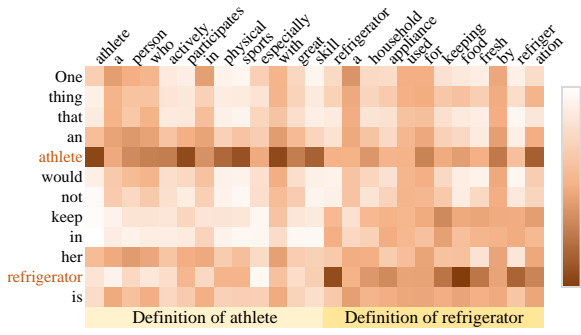


Fig. 3. Attention distribution diagram (BART+KE).

TABLE IV
ANSWER GENERATION EXAMPLES.

Question #1	One thing that an <i>athlete</i> would not keep in her <i>refrigerator</i> is
Knowledge	<i>athlete</i> : a person who actively participates in physical sports, especially with great skill. <i>refrigerator</i> : a household appliance used for keeping food fresh by refrigeration.
BART	beer, ice cream, alcohol
BART+KE	beer, ice cream, chocolate, clothes, socks
Question #2	One thing a <i>monk</i> <i>probably</i> would not own is
Knowledge	<i>monk</i> : A male member of a monastic order who has devoted his life for religious service. <i>probably</i> : In all likelihood.
BART	car, cell phone, money
BART+KE	car, cell phone, money, gun, jewelry, wife

The answers generated by BART and its enhanced version (+KE) are considered.

set to a larger number. The answer deduplication module in KE dynamically influences the loss of top-1, 3 or 5 answers (by redundancy elimination and novel answer replacement). Besides, PR serves as the post-treatment which separately refines the rankings of not only top-5 cases but top-10.

G. Effects of Knowledge Enhancement

KE contributes to the diversification of answers. Let's consider the BART-based GenCQA model which is merely coupled with KE. It additionally recalls 34% different classes of answers (see the examples in Table IV), in terms of our verification on VSet2. This contributes to the improvement of Ans@12 with a rate of up to about 14.2%. The positive effects of KE benefit from the increased intention weights of keywords, i.e., the ones obtained by interaction between questions and definition descriptions of keywords. Fig. 3 shows an example of attention distribution diagram. In Appendix D, we present the method for creating attention heat maps.

H. Reliance of Knowledge Enhancement

KE heavily relies on the qualified knowledge items. All the pretreatments (TF-IDF keyword extraction, DPR and the setting of number) may influence the quality.

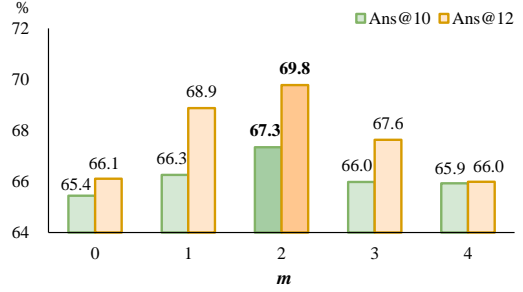


Fig. 4. Effects of m on Knowledge Enhanced Answer Generation. The performance of best m is in **bold**.

1) *Reliability of Keyword Extraction*: We extract top- m keywords for each question conditioned on the statistical distribution feature TF-IDF. The words possessing higher TF-IDF scores will be adopted as the keywords. We verify the reliability of the keywords in a separate experiment. Specifically, we randomly select 50 questions from the validation set VSet1, and use the aforementioned statistical approach to extract keywords. Meanwhile, we manually annotate the keywords, and rank them according to their importance. There are 2.3 keywords on average annotated for each question, and the maximum number is 4. We determine the extracted keywords to be correct if the agreement is reached conditioned on the annotation results. On this basis, we verify the Macro-average accuracy for the top- m ($m \leq 4$) TF-IDF extracted keywords. The performance is 0.88, 0.80, 0.69 and 0.56 for top-1, 2, 3 and 4 keywords, respectively.

2) *Reliability of DPR*: We use DPR to determine the semantic similarity between keyword-based queries and definition descriptions for selecting relevant descriptions (Section III.A). We evaluated DPR's reliability for knowledge acquisition by manually verifying 50 keywords and found an accuracy of 0.86, significantly higher than simply using the primary definition [33] (accuracy of 0.66). Note that the primary definition refers to the top-1 placed definition in Wiktionary.

3) *Number of Knowledge Items*: KE is used to increase the diversity of answers by covering a wider range of answer categories. To determine the optimal setting for the number of available knowledge items m , we conduct a preliminary evaluation of enhanced baselines (PLMs+KE) at different values of m . Ans@10 and Ans@12 serve as evaluation metrics as they reflect overall recall. The performance of using different numbers of knowledge items was evaluated and shown in Fig. 4. Results indicate that using too few ($m = 0$ or $m = 1$) or too many knowledge items for KE decreases the accuracy. Using too few knowledge items is ineffective in capturing various answer classes, while using too many introduces noise from incorrect keywords. Therefore, we set m to 2 to balance accuracy and diversity of knowledge answers.

I. Constructing a Proper PR Corpus

To rank the generated answers, we use a plausibility scoring model based on the likelihood of an answer being completely

correct and reasonable. We train an ELECTRA-based binary classifier using a corpus of positive and negative answers. For selecting positive instances, we choose the top- n answers from the ground-truth list in descending order of popularity. We experiment with different values of n and find that a value of 2 yields the best performance. The detailed experiment result and case study are provided in Appendix E.

V. CONCLUSION

We propose a KEPR approach to strengthen the PLMs-based GenCQA models. Experimental results demonstrate that our approach yields substantial improvements, and the resultant GenCQA models outperform the state of the art. In addition, KEPR comprises three parts, including knowledge retriever, answer generator and plausibility ranker. Each of them is designed to be a general component for GenCQA tasks. However, there is only one GenCQA corpus publicly released for evaluation. Therefore we fail to verify and guarantee the generality on multiple datasets. If any other GenCQA dataset is publicly available, we will report the generality on them, so as to provide a reference for future research.

REFERENCES

- [1] A. Talmor, O. Yoran, R. L. Bras, C. Bhagavatula, Y. Goldberg, Y. Choi, and J. Berant, "Commonsenseqa 2.0: Exposing the limits of ai through gamification," in *NeurIPS Datasets and Benchmarks*, 2021.
- [2] A. Talmor, J. Herzig, N. Lourie, and J. Berant, "CommonsenseQA: A question answering challenge targeting commonsense knowledge," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4149–4158.
- [3] D. Ghosal, S. Shen, N. Majumder, R. Mihalcea, and S. Poria, "CICERO: A dataset for contextualized commonsense inference in dialogues," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, May 2022, pp. 5010–5028.
- [4] M. Boratko, X. Li, T. O’Gorman, R. Das, D. Le, and A. McCallum, "ProtoQA: A question answering dataset for prototypical common-sense reasoning," in *Proceedings of the 2020 Conference on EMNLP*, 2020, pp. 1122–1136.
- [5] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.
- [6] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 7871–7880.
- [7] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, pp. 1–67, 2020.
- [8] W. Shen, Y. Gong, Y. Shen, S. Wang, X. Quan, N. Duan, and W. Chen, "Joint generator-ranker learning for natural language generation," *CoRR*, vol. abs/2206.13974, 2022.
- [9] K. Clark, M. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," in *Proceedings of the 8th International Conference on Learning Representations*, 2020, pp. 1–9.
- [10] X. Luo, C. Fan, Y. Zhang, W. Jiang, B. Qin, and R. Xu, "Masked language models know which are popular: A simple ranking strategy for commonsense question answering," in *Findings of the Association for Computational Linguistics: EMNLP 2022*, 2022, pp. 3200–3213.
- [11] K. Ma, F. Ilievski, J. Francis, S. Ozaki, E. Nyberg, and A. Oltramari, "Exploring strategies for generalizable commonsense reasoning with pre-trained models," in *Proceedings of the 2021 Conference on EMNLP*, 2021, pp. 5474–5483.
- [12] B. Y. Lin, W. Zhou, M. Shen, P. Zhou, C. Bhagavatula, Y. Choi, and X. Ren, "CommonGen: A constrained text generation challenge for generative commonsense reasoning," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp. 1823–1840.
- [13] Z. Fan, Y. Gong, Z. Wei, S. Wang, Y. Huang, J. Jiao, X. Huang, N. Duan, and R. Zhang, "An enhanced knowledge injection model for commonsense generation," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 2014–2025.
- [14] H. Li, Y. Gong, J. Jiao, R. Zhang, T. Baldwin, and N. Duan, "KFCNet: Knowledge filtering and contrastive learning for generative commonsense reasoning," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, Nov. 2021, pp. 2918–2928.
- [15] Y. Liu, Y. Wan, L. He, H. Peng, and P. S. Yu, "KG-BART: Knowledge graph-augmented bart for generative commonsense reasoning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 7, pp. 6418–6425, May 2021.
- [16] P. Banerjee, K. K. Pal, A. Mitra, and C. Baral, "Careful selection of knowledge to solve open book question answering," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 6120–6129.
- [17] B. Y. Lin, X. Chen, J. Chen, and X. Ren, "KagNet: Knowledge-aware graph networks for commonsense reasoning," in *Proceedings of the 2019 Conference on EMNLP and the 9th IJCNLP*, 2019, pp. 2829–2839.
- [18] Q. Chen, F. Ji, H. Chen, and Y. Zhang, "Improving commonsense question answering by graph-based iterative retrieval over multiple knowledge sources," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 2583–2594.
- [19] Y. Li, B. Zou, Z. Li, A. Aw, Y. Hong, and Q. Zhu, "Winnowing knowledge for multi-choice question answering," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2021, pp. 1157–1165.
- [20] Y. Xu, C. Zhu, S. Wang, S. Sun, H. Cheng, X. Liu, J. Gao, P. He, M. Zeng, and X. Huang, "Human parity on commonsenseqa: Augmenting self-attention with external attention," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*, 2021.
- [21] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.
- [22] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020, pp. 6769–6781.
- [23] K. Arora, L. El Asri, H. Bahuleyan, and J. C. K. Cheung, "Why exposure bias matters: An imitation learning perspective of error accumulation in language generation," in *Findings of the Association for Computational Linguistics: ACL 2022*, 2022, pp. 700–710.
- [24] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014, pp. 103–111.
- [25] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.
- [26] U. MOWGLI, "BART-large Fine-tuned," Website, 2021, <https://leaderboard.allenai.org/protoqa/submission/bvr5uqi28mv4hjrkb9fg>.
- [27] Team Cosmic, "T5-11B+Ranked Answer," Website, 2021, <https://leaderboard.allenai.org/protoqa/submission/bvu9ds927eih0j02ltg>.
- [28] SUDA NLP, "Few-shot GPT3 Davinci (175b)," Website, 2022, <https://leaderboard.allenai.org/protoqa/submission/ce414defl39t3rrl6lm0>.
- [29] U. MOWGLI, "DPR+BART," Website, 2021, <https://leaderboard.allenai.org/protoqa/submission/bvu0a9thgebfiaml1ac0>.
- [30] MOWGLI and Stanford, "GPT2+ConceptNet," Website, 2021, <https://leaderboard.allenai.org/protoqa/submission/c5qds30b0tcj6heq4rg>.
- [31] R. Speer, J. Chin, and C. Havasi, "Conceptnet 5.5: An open multilingual graph of general knowledge," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, Feb. 2017. [Online]. Available: <https://ojs.aaai.org/index.php/AAAI/article/view/11164>
- [32] P. He, J. Gao, and W. Chen, "Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing," *CoRR*, vol. abs/2111.09543, pp. 0–17, 2021. [Online]. Available: <https://arxiv.org/abs/2111.09543>
- [33] Y. Xu, C. Zhu, R. Xu, Y. Liu, M. Zeng, and X. Huang, "Fusing context into knowledge graph for commonsense question answering," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021, pp. 1201–1207.