Clues Before Answers: Generation-Enhanced Multiple-Choice QA

Zixian Huang and **Ao Wu** and **Jiaying Zhou**
State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
{zixianhuang,awu,jyzhou}@smail.nju.edu.cn

Yu Gu
The Ohio State University, Columbus, USA
gu.826@osu.edu

Yue Zhao and **Gong Cheng**
State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
yuezhao@smail.nju.edu.cn, gcheng@nju.edu.cn

Abstract

A trending paradigm for multiple-choice question answering (MCQA) is using a text-to-text framework. By unifying data in different tasks into a single text-to-text format, it trains a generative encoder-decoder model which is both powerful and universal. However, a side effect of twisting a generation target to fit the classification nature of MCQA is the underutilization of the decoder and the knowledge that can be decoded. To exploit the generation capability and underlying knowledge of a pre-trained encoder-decoder model, in this paper, we propose a generation-enhanced MCQA model named GenMC. It generates a clue from the question and then leverages the clue to enhance a reader for MCQA. It outperforms text-to-text models on multiple MCQA datasets.

1 Introduction

Multiple-choice question answering (MCQA) aims at selecting the correct answer from a set of options given a question. This long-standing challenge in natural language processing (NLP) requires machines to have a wealth of knowledge, such as commonsense knowledge (Talmor et al., 2019; Mihaylov et al., 2018) and scientific knowledge (Clark et al., 2018; Khot et al., 2020; Huang et al., 2019; Li et al., 2021), and have reasoning skills such as multi-hop reasoning (Khot et al., 2019) and logical reasoning (Yu et al., 2020; Liu et al., 2020b; Li et al., 2022).

MCQA has made great progress with the development of pre-trained language models (PLMs). Basically there are two types of PLMs that are suitable for different tasks. BERT (Devlin et al., 2019) and its variants such as RoBERTa (Liu et al., 2019) and ALBERT (Lan et al., 2020) are encoder-only models, being more suitable for natural language understanding (NLU) tasks including MCQA and other classification and regression tasks. T5 (Raffel et al., 2020) and BART (Lewis et al., 2020) are encoder-decoder models, being more suitable for natural language generation (NLG) tasks. However, encoder-decoder models can also be applied to MCQA (Khashabi et al., 2020; Zhou et al., 2021). This is enabled by the text-to-text framework, which transforms data in different tasks into a unified text-to-text format so that knowledge spanning many and various tasks can be learned, aggregated, and used by a single model.

Research Question To fit MCQA, existing implementations of the text-to-text framework take all the options as input and are trained to generate one of the options, i.e., to copy some tokens from the input. However, this is inconsistent with how encoder-decoder models are pre-trained so that their underlying knowledge may not be sufficiently exploited. Indeed, Liu et al. (2021) have found that in classification and regression tasks, the decoder layer is often under-utilized. One research question is *how to apply pre-trained encoder-decoder models in a more natural way to MCQA*, in particular, to exploit their NLG capabilities.

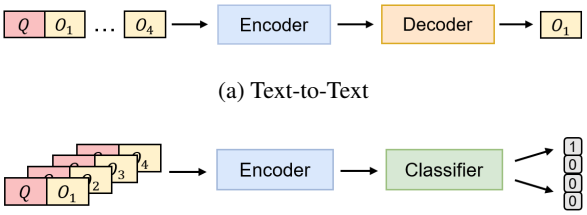
Our Contribution Our idea is inspired by human behavior. When reading a question, humans are sometimes triggered to associate the question with their background knowledge to form some *clues* even before reading the options. For simple questions, a clue may be exactly the correct answer,

Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3272 - 3287

Question:
A company makes notebooks for college courses, so their main material is ?
(A) Chips (B) Water (C) Grass (D) **Trees**

Clue: Paper

Figure 1: An example MCQA task and a generated clue. Bold underline indicates the correct answer.



(a) Text-to-Text

(b) Encoder-Only

Figure 2: Paradigms for MCQA.

while for complex questions, clues may play an auxiliary role to help humans connect the question with the correct answer. For example, for the question shown in Figure 1, the clue "paper" forms an intermediate concept between "notebook" in the question and "tree" in the correct answer.

With this idea, we propose to employ a pre-trained encoder-decoder model to *generate* a clue from the question by exploiting its underlying knowledge, without seeing and being strictly confined to the options as in the text-to-text framework. The clue representation is then leveraged by an encoder-based model to read the options and make prediction. We refer to this generation-enhanced MCQA model as **GenMC**. It significantly outperforms comparable models, in particular, text-to-text models, on five MCQA datasets.

Outline We discuss related work in Section 2, introduce GenMC in Section 3, describe the experimental setup in Section 4, report the results in Section 5, and conclude in Section 6.

Code Our code is available on GitHub[1] under the Apache Licence 2.0.

2 Related Work

2.1 Text-to-Text Paradigm for MCQA

Recently, the text-to-text paradigm has achieved breakthrough results on many NLP tasks (Raffel et al., 2020; Lewis et al., 2020). As illustrated in Figure 2a, adopting this paradigm for MCQA, the question Q and all the options $\{O_1, O_2, O_3, O_4\}$ are spliced into a text as input, and the correct answer O_1 is used as the generation target. One benefit is that extensive training data can be shared across different tasks. Using such a framework, UnifiedQA (Khashabi et al., 2020) integrates 20 QA datasets into a unified format for training, and achieves state-of-the-art results on multiple MCQA datasets. Similarly, CALM (Zhou et al., 2021) learns concept-centric knowledge from text for commonsense QA.

However, it might be debatable whether it is appropriate to train a classification task via a generation target. Liu et al. (2021) point out that the decoder layers of T5 are under-utilized when fine-tuning on classification and regression tasks. Therefore, they propose a method to reduce the number of T5 parameters to improve efficiency without reducing accuracy. By contrast, we address this issue from a different perspective of how to exploit the NLG capability of pre-trained encoder-decoder models for MCQA to improve accuracy.

Some other works propose new pre-trained models for unified generation and classification tasks by designing universal encoders and task-specific decoders (Shao et al., 2021; Sun et al., 2021). They are orthogonal to our work as we leverage existing pre-trained encoder-decoder models instead of pre-training new models at an additional cost.

2.2 Encoder-Only Paradigm for MCQA

Benefiting from the powerful NLU capabilities of BERT-style PLMs (Devlin et al., 2019; Liu et al., 2019; Lan et al., 2020), the encoder-only paradigm has been popular for MCQA. As illustrated in Figure 2b, in this paradigm, the question Q and each option in $\{O_1, O_2, O_3, O_4\}$ are interacted to calculate a score, and the option with the highest score is chosen as the answer. Building on this, some works study how to design better attention-based models to identify evidence (Chen et al., 2019; Zhang et al., 2020; Zhu et al., 2020). Other efforts mimic human behavior of reading evidence and answering questions (Ran et al., 2019; Tang et al., 2019; Sun et al., 2019). There, evidence is derived from the given passage or retrieved from external corpora. By contrast, we aim at exporting clues from pre-trained models without resorting to extra sources.

[1] https://github.com/nju-websoft/GenMC

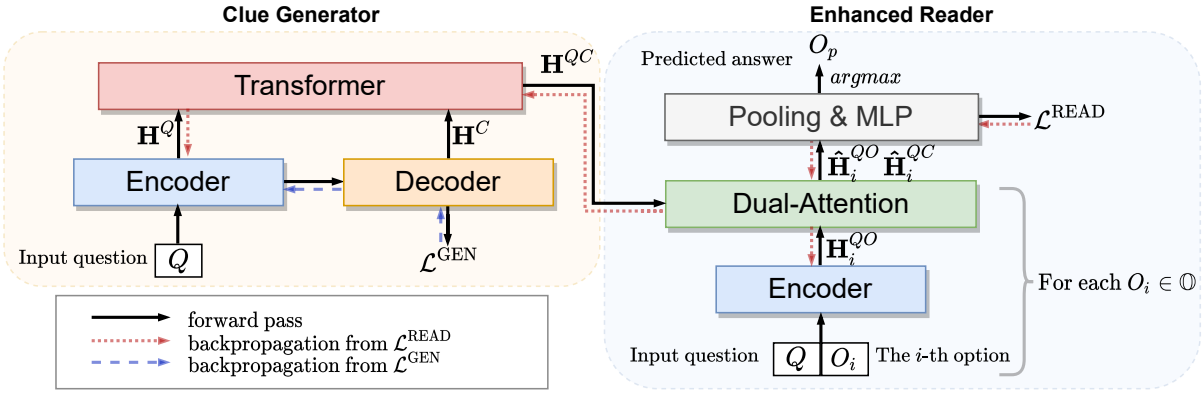


Figure 3: Architecture of GenMC. To make the prediction $O_p \in \mathbb{O}$, the clue generator first takes Q as input and outputs a clue representation $\mathbf{H}^{QC}$ which is indicative of the correct answer. The enhanced reader then relies on the generated clue representation to better attend to options from $\mathbb{O}$ and makes the final prediction. The whole model is trained in an end-to-end manner with both the generation loss $\mathcal{L}^{\text{GEN}}$ and the classification loss $\mathcal{L}^{\text{READ}}$.

2.3 Knowledge in PLMs

Recently, PLMs have been used as knowledge bases (Petroni et al., 2019), and the knowledge in parameters can be exported via methods such as Prompt (Jiang et al., 2020; Shin et al., 2020). Exploiting the knowledge in PLMs for QA tasks has come into play in many forms including question expansion (Mao et al., 2021) and question generation (Shwartz et al., 2020).

There is also research on MCQA trying to exporting knowledge from PLMs before answering. Rajani et al. (2019) propose CAGE as a framework for generating explanations for commonsense QA. However, CAGE relies on explanations annotated by humans, which are not available in many real scenarios and datasets. Latcinnik and Berant (2020) propose a joint generator-classifier model where the generator produces a human-readable textual hypothesis. Although it somewhat improves the explainability of MCQA, in terms of accuracy of MCQA there is little advancement. CEGI (Liu et al., 2020c) is probably the most similar work to ours. It first uses a generative model to generate evidence, and then uses a reading model to incorporate the evidence and predict the answer, both using answer supervision. However, the generative model and the reading model are separate steps in a pipeline and are connected only via the evidence text. Such token-level interaction can lead to significant losses in accuracy as we will see in our experiments, where our representation-level interaction exhibits better performance.

3 GenMC Model

In MCQA, a question Q is given together with a set of n options $\mathbb{O} = \{O_1, \ldots, O_n\}$ with exactly one option being the correct answer. The key to finding the correct answer is to capture and deeply understand the connection between Q and each $O_i \in \mathbb{O}$, which oftentimes is beyond the lexical level and requires a non-trivial entailment process. We follow the trend of building on a pre-trained encoder-decoder model and use the encoder to jointly encode Q and each O_i. However, previous works directly use the decoder to generate an option in $\mathbb{O}$, i.e., using the decoder as a classifier, which may have under-exploited the model's NLG capability (Liu et al., 2021). Moreover, a simple joint encoding of Q and each O_i can only enable lexical-level reasoning (Zellers et al., 2019) which is insufficient for MCQA tasks.

Our proposed model GenMC overcomes these limitations. Building on a pre-trained encoder-decoder model, GenMC firstly generates a clue which is indicative of the correct answer, thereby exploiting the NLG capability and underlying knowledge of the pre-trained encoder-decoder model. Then GenMC employs the generated clue representation as intermediate knowledge connecting the question and the correct answer to interact with and enhance a reader for solving MCQA. Our model design mimics how humans solve an MCQA task, i.e., after reading a question, humans may firstly associate it with some of their background knowledge (i.e., looking for clues) that helps them to later identify the correct answer.

The overall architecture of GenMC is shown in

Figure 3. The clue generator (Section 3.1) first generates a clue representation only given Q. Then the enhanced reader (Section 3.2) uses the generated clue to augment question-option understanding.

3.1 Clue Generator

The clue generator takes the question Q as input and autoregressively outputs a clue $C = c_1, \ldots, c_{|C|}$ using a pre-trained encoder-decoder model.[2] Note that not the clue text C but its representation $\mathbf{H}^C$ will be used in our model, although one could output C as evidence for explainability.

Specifically, we obtain the question representation $\mathbf{H}^Q \in \mathbb{R}^{d\times|Q|}$ and the clue representation $\mathbf{H}^C \in \mathbb{R}^{d\times|C|}$ from the last layer of the encoder and of the decoder, respectively, where d denotes the representation dimension. $\mathbf{H}^C_j$, denoting the representation of the j-th token $c_j \in C$, is computed as follows:

$$\mathbf{p}^C_j, \mathbf{H}^C_j = \texttt{Decoder}(c_{j-1}, \mathbf{H}^C_{<j}, \mathbf{H}^Q)\,, \tag{1}$$

where $\texttt{Decoder}\,(\cdot,\cdot,\cdot)$ takes the last token c_{j-1}, the representation for the decoding history $\mathbf{H}^C_{<j}$, and $\mathbf{H}^Q$ as input, and outputs the hidden state $\mathbf{H}^C_j$ together with the probability distribution $\mathbf{p}^C_j$ over the decoding vocabulary at the j-th step.

To encourage the tokens in C to thoroughly interact with each other and with Q, we strengthen the clue representation by passing it to a transformer layer (Vaswani et al., 2017) and obtain $\mathbf{H}^{QC}$:

$$\mathbf{H}^{QC} = \texttt{Transformer}([\mathbf{H}^Q; \mathbf{H}^C])\,, \tag{2}$$

where $[\cdot;\cdot]$ denotes concatenation. $\mathbf{H}^{QC}$ carries the information of C which can be helpful to better understand and answer Q.

3.2 Enhanced Reader

Previous works often directly model the relevance of each $O_i \in \mathbb{O}$ to Q via joint encoding using a pre-trained encoder, which largely performs superficial lexical reasoning (Zellers et al., 2019). By contrast, we use the previously generated clue representation to enhance our reader for a deeper understanding of each question-option pair.

Specifically, we first concatenate Q and each O_i independently[3] and feed the concatenated input into the pre-trained encoder (which is shared with our clue generator) to obtain O_i's contextualized representation $\mathbf{H}^{QO}_i$, which constitutes a column of $\mathbf{H}^{QO} \in \mathbb{R}^{d\times n}$ where $n = |\mathbb{O}|$.

Next, based on the clue representation $\mathbf{H}^{QC}$, our model intensively reads each question-option pair and obtains the matching signal between the clue and the option. Specifically, inspired by Huang et al. (2021), we first use dual-attention (Liu et al., 2020a) to fuse information from $\mathbf{H}^{QO}_i$ to $\mathbf{H}^{QC}$ and from $\mathbf{H}^{QC}$ to $\mathbf{H}^{QO}_i$. Then we perform max-pooling to aggregate the matching features:

$$\begin{aligned}(\hat{\mathbf{H}}^{QO}_i, \hat{\mathbf{H}}^{QC}_i) &= \text{DualAttention}(\mathbf{H}^{QO}_i, \mathbf{H}^{QC})\,,\\ \mathbf{f}^{QO}_i &= \text{Max-Pooling}(\hat{\mathbf{H}}^{QO}_i)\,,\\ \mathbf{f}^{QC}_i &= \text{Max-Pooling}(\hat{\mathbf{H}}^{QC}_i)\,.\end{aligned} \tag{3}$$

To obtain the final score s_i for each O_i, we concatenate the dual matching features $\mathbf{f}^{QO}_i$ and $\mathbf{f}^{QC}_i$ and feed them into a two-layer multi-layer perceptron (MLP):

$$s_i = \text{Linear}(\text{ReLU}(\text{Linear}([\mathbf{f}^{QO}_i; \mathbf{f}^{QC}_i]))\,. \tag{4}$$

We select the option with the highest score as the predicted answer, denoted as O_p.

3.3 Training Objective

We jointly train the clue generator and the enhanced reader in an end-to-end fashion with a combined loss:

$$\mathcal{L} = \mathcal{L}^{\text{GEN}} + \mathcal{L}^{\text{READ}}\,. \tag{5}$$

Generator Loss For $\mathcal{L}^{\text{GEN}}$, assuming that $O_t \in \mathbb{O}$ is the correct answer containing m tokens $a_1, \ldots, a_m$, we first use O_t as the target to calculate our clue generator loss with teacher forcing:

$$\begin{aligned}\mathbf{p}^{O_t}_j, \mathbf{H}^{O_t}_j &= \texttt{Decoder}(a_{j-1}, \mathbf{H}^{O_t}_{<j}, \mathbf{H}^Q)\,,\\ \mathcal{L}^{\text{GEN}} &= -\frac{1}{m}\sum_{j=1}^{m}\log \mathbf{p}^{O_t}_{j,a_j}\,,\end{aligned} \tag{6}$$

where $\mathbf{p}^{O_t}_j$ denotes the probability distribution over the decoding vocabulary at the j-th step, and $\mathbf{p}^{O_t}_{j,a_j}$ is the probability of token a_j.

Reader Loss For $\mathcal{L}^{\text{READ}}$, we simply calculate a cross-entropy loss given the correct answer $O_t \in \mathbb{O}$ as follows:

$$\mathcal{L}^{\text{READ}} = -\log\frac{\exp(s_t)}{\sum_{i=1}^{n}\exp(s_i)}\,. \tag{7}$$

Note that we update the encoder using the joint loss $\mathcal{L}$, while we do not allow $\mathcal{L}^{\text{READ}}$ to be backpropagated to the decoder part to reduce the memory consumption.

[2] For efficiency, we decode the clue greedily without performing beam search.

[3] A delimiter "\n" is inserted between Q and each O_i.

	Train set size	Dev set size	Test set size	Option number	Question average length	Option average length
CSQA	8,500	1,241	1,221	5	13.38	1.52
OBQA	4,957	500	500	4	10.65	2.85
ARC-Easy	2,241	567	2,365	4	19.36	3.73
ARC-Challenge	1,117	295	1,165	4	22.30	4.93
QASC	7,320	814	926	8	8.12	1.64

Table 1: Dataset statistics. For CSQA and QASC, their official dev sets are used as our test sets, and our dev sets are in-house split from their official training sets.

The above training objective exploits the double properties of the correct answer O_t in MCQA: as a text and as an index. We use O_t as a text to supervise our clue generator, and as an index (i.e., classification label) to supervise our enhanced reader. Such usage is more natural than the text-to-text paradigm (Khashabi et al., 2020; Zhou et al., 2021), thus having the potential to outperform.

4 Experimental Setup

4.1 Data

We conducted experiments on five popular MCQA datasets spanning from commonsense questions to scientific questions. The former requires commonsense knowledge and reasoning, and the latter requires inference over scientific facts.

Datasets CSQA (Talmor et al., 2019) and OBQA (Mihaylov et al., 2018) are two commonsense MCQA datasets created by crowd workers based on commonsense facts. Each question is given with 5 options in CSQA and 4 options in OBQA. ARC-Easy and ARC-Challenge, denoting two disjointed subsets of ARC (Clark et al., 2018), contain natural grade-school science questions with 4 options, where ARC-Challenge comprises difficult questions which require more advanced reasoning. QASC (Khot et al., 2020) is collected from elementary and middle school level science with 8 options for each question.

Train-Dev-Test Split For OBQA, ARC-Easy, and ARC-Challenge we used their official train, dev, and test sets. For CSQA and QASC, since the correct answers in the official test set are not public, we took their official dev set as our test set for experiments and randomly held out an in-house dev set from the training set. The dataset statistics are shown in Table 1.

External Knowledge For all these datasets, our experiments did not rely on any provided documents or external corpora; a question was solely provided with its options to form the input. It means that pre-trained models were used as the primary source of knowledge in the experiments.

4.2 Implementation Details

We used two popular encoder-decoder models as a basis, BART (Lewis et al., 2020) and T5 (Raffel et al., 2020). For each model, we experimented with its BASE and LARGE versions.

We used PyTorch 1.7. We used the Adam optimizer and set warmup fraction $= 0.1$, weight decay $= 0.01$, maximum source length $= 64$, maximum target length $= 32$, epoch $= 30$, and early stop training when there was no better result on the dev set after 5 epochs. For each model, we searched for the best learning rate from $\{1e-4, 5e-5, 1e-5\}$, and for the best batch size out of $\{8, 64\}$.

Because neural models are known to be sensitive to different random seeds, especially when the training set is small, we performed multiple experiments for all models with different random seeds, and reported the mean and standard deviation. For CSQA, OBQA, ARC-Easy, and QASC, we used three random seeds $\{1, 10, 20\}$. For the smallest dataset ARC-Challenge, we used five random seeds $\{1, 10, 20, 30, 40\}$.

All the experiments were performed on a GeForce RTX 3090 with 24G memory.

4.3 Evaluation Metric

For each model, we reported its proportion of correctly answered questions in each dataset.

5 Experimental Results

5.1 Main Results: Comparison with Text-to-Text Models

To empirically evaluate GenMC in terms of whether it better exploits the potential of pre-trained encoder-decoder models for MCQA, we compare GenMC with a standard text-to-text implementation and with a variant thereof for analysis.

	CSQA		OBQA		ARC-Easy		ARC-Challenge		QASC	
	dev	test	dev	test	dev	test	dev	test	dev	test
$\text{BART}_{\text{BASE}}$										
$\text{Text2Text}_{\text{vanilla}}$	51.62 (±0.04)	53.26 (±0.57)	54.93 (±0.83)	52.73 (±1.00)	51.55 (±1.38)	50.51 (±1.82)	30.05 (±1.25)	24.95 (±1.10)	46.72 (±1.21)	26.78 (±1.21)
$\text{Text2Text}_{\text{enc}}$	50.63 (±0.66)	52.22 (±1.64)	55.87 (±1.10)	51.00 (±1.83)	49.03 (±1.86)	49.94 (±1.49)	32.32 (±4.87)	26.24 (±2.01)	48.08 (±1.35)	17.06 (±0.39)
GenMC	**54.82** (±0.61)	**56.40** (±0.61)	**58.53** (±0.31)	**57.53** (±2.91)	**59.38** (±1.60)	**56.80** (±0.28)	**38.64** (±0.90)	**33.82** (±1.66)	**57.70** (±0.43)	**35.96** (±1.70)
T5_{BASE}										
$\text{Text2Text}_{\text{vanilla}}$	57.59 (±0.81)	60.93 (±0.73)	59.53 (±0.81)	57.53 (±0.70)	52.20 (±0.31)	51.75 (±0.89)	29.38 (±2.63)	23.69 (±2.47)	54.55 (±1.01)	37.94 (±1.47)
$\text{Text2Text}_{\text{enc}}$	58.96 (±1.21)	59.49 (±1.41)	60.67 (±2.86)	57.07 (±3.03)	56.55 (±1.17)	52.92 (±0.29)	29.49 (±5.13)	26.09 (±0.23)	56.84 (±0.84)	39.60 (±2.38)
GenMC	**60.65** (±0.47)	**63.45** (±0.29)	**62.07** (±1.01)	**61.67** (±0.58)	**62.38** (±0.67)	**58.82** (±0.37)	**43.62** (±0.52)	**39.00** (±0.30)	**58.93** (±1.76)	**41.72** (±1.18)
$\text{BART}_{\text{LARGE}}$										
$\text{Text2Text}_{\text{vanilla}}$	65.58 (±2.72)	66.91 (±2.14)	62.66 (±1.18)	61.46 (±1.74)	63.49 (±1.89)	62.81 (±2.15)	29.94 (±2.32)	28.55 (±4.97)	64.57 (±2.21)	47.80 (±2.22)
$\text{Text2Text}_{\text{enc}}$	65.00 (±0.66)	67.35 (±0.90)	63.80 (±1.44)	62.47 (±1.53)	68.20 (±2.04)	65.33 (±1.74)	35.37 (±6.07)	31.13 (±5.86)	65.07 (±0.94)	47.19 (±0.71)
GenMC	**69.57** (±0.89)	**72.26** (±0.70)	**68.93** (±1.17)	**68.07** (±1.70)	**72.43** (±0.54)	**68.68** (±0.34)	**48.93** (±0.98)	**45.52** (±1.54)	**68.39** (±0.68)	**55.90** (±0.92)
T5_{LARGE}										
$\text{Text2Text}_{\text{vanilla}}$	67.53 (±0.43)	70.63 (±0.74)	66.80 (±0.87)	63.53 (±1.10)	65.61 (±0.18)	62.55 (±0.54)	43.05 (±1.69)	42.83 (±2.00)	64.13 (±1.47)	57.74 (±0.82)
$\text{Text2Text}_{\text{enc}}$	68.41 (±0.73)	70.30 (±0.82)	65.93 (±1.03)	63.67 (±0.46)	69.61 (±0.20)	66.65 (±0.34)	30.73 (±3.15)	28.76 (±4.85)	65.27 (±1.55)	55.65 (±0.45)
GenMC	**71.10** (±0.41)	**72.67** (±1.02)	**71.60** (±0.92)	**66.87** (±1.33)	**72.49** (±0.77)	**69.01** (±1.97)	**49.83** (±2.06)	**47.41** (±2.00)	**67.61** (±1.14)	**58.06** (±0.92)

Table 2: Comparison with text-to-text models.

5.1.1 Baselines

$\text{Text2Text}_{\text{vanilla}}$ The vanilla usage of pre-trained encoder-decoders for MCQA is to reform the input and output in a way that can be directly processed by a encoder-decoder model. Specifically, following Raffel et al. (2020), we concatenate the input question with all candidate options, where each option is also preceded by its option ID, and then prepend the sequence with a dataset name. The concatenated sequence is fed into the encoder part to get a joint representation for the question and all options. Based on the joint representation, the decoder finally outputs an option ID. In this setting, the decoder is basically used as a classifier.

$\text{Text2Text}_{\text{enc}}$ Similar to Liu et al. (2021), we use only the encoder part of a pre-trained encoder-decoder model. Each option is independently paired with the question to obtain a joint representation using the encoder. Then the representation is fed into a scorer (i.e., an MLP) to obtain a matching score for each question-option pair. The model then predicts the option with the highest score. In this setting, the decoder is totally unused. Though Liu et al. (2021) find that their encoder-only model performs comparably to using the decoder as a classifier, we argue that the decoder part can further improve the performance, if being properly used.

5.1.2 Results

The main results (see Table 2) show that GenMC consistently and significantly (with p-value < 0.01) outperforms $\text{Text2Text}_{\text{vanilla}}$ and $\text{Text2Text}_{\text{enc}}$ on all datasets. For several settings, GenMC even obtains an absolute gain of over 10%. For example, on the test set of the challenging scientific MCQA dataset ARC-Challenge, T5_{BASE} + GenMC improves T5_{BASE} + $\text{Text2Text}_{\text{vanilla}}$ from an accuracy of 23.69% to 39.00%, suggesting a relative gain of around 65%. These results demonstrate that GenMC is a more effective usage of pre-trained encoder-decoder models than existing ones.

Moreover, we interestingly find that the decoder-free baseline $\text{Text2Text}_{\text{enc}}$ outperforms $\text{Text2Text}_{\text{vanilla}}$ on over half of the experiments. This indicates that the decoder's general language knowledge gained from pre-training is largely wasted by only using it as a classifier, which may further explain the superior performance of our model because GenMC can exploit the pre-trained decoder more effectively. In addition, all LARGE models significantly outperform their BASE counterparts. This suggests that the embedded knowledge gained from pre-training is critical to MCQA tasks, strengthening our point to make full use of pre-trained encoders and decoders.

5.2 Comparison with Other Models

5.2.1 Baselines

UnifiedQA Existing methods that rely on external documents or corpora have achieved state-of-the-art performance on several MCQA datasets. However, to enable a fair comparison, we only compare with models that adopt the same setting as ours, where a question and its options are the only input to the model. Among these models, UnifiedQA (Khashabi et al., 2020) is the current best model. While UnifiedQA reports the best score using its T5-11B version, since for T5 we experiment with its BASE and LARGE versions, we only report and compare under T5_{BASE} and T5_{LARGE}. Note that instead of training on each dataset separately, UnifiedQA converts a line of popular QA datasets with four formats (e.g., retrieval-based QA, MCQA) into a unified format, and trains a single model over all training data, while GenMC only uses each

	CSQA		OBQA		ARC-Easy		ARC-Challenge		QASC	
	dev	test	dev	test	dev	test	dev	test	dev	test
BASE										
RoBERTa	56.51 (±0.34)	58.91 (±0.79)	58.67 (±1.03)	49.67 (±0.76)	56.56 (±0.91)	52.32 (±0.70)	38.64 (±0.90)	34.85 (±2.20)	55.28 (±0.12)	34.38 (±1.72)
ALBERT	53.16 (±0.58)	53.95 (±0.49)	54.53 (±1.10)	49.20 (±2.27)	48.32 (±0.88)	45.84 (±1.94)	34.80 (±1.53)	30.21 (±1.74)	40.99 (±(1.78)	24.55 (±1.23)
UnifiedQA$_{T5}$ *	-	45.00 (±0.00)	-	59.00 (±0.00)	-	53.00 (±0.00)	-	42.40 (±0.00)	-	25.80 (±0.00)
UnifiedQA$_{T5}$	41.02 (±0.00)	44.80 (±0.00)	59.20 (±0.00)	59.60 (±0.00)	54.85 (±0.00)	53.66 (±0.00)	44.75 (±0.00)	**42.58** (±0.00)	17.94 (±0.00)	25.70 (±0.00)
UnifiedQA$_{T5\text{-}FT}$	56.81 (±0.49)	62.35 (±0.80)	60.80 (±0.72)	58.47 (±0.64)	54.97 (±0.20)	53.88 (±0.39)	**45.31** (±0.39)	42.43 (±0.47)	55.57 (±0.58)	**43.20** (±0.57)
GenMC$_{T5}$	**60.65** (±0.47)	**63.45** (±0.29)	**62.07** (±1.01)	**61.67** (±0.58)	**62.38** (±0.67)	**58.82** (±0.37)	43.62 (±0.52)	39.00 (±0.30)	**58.93** (±1.76)	41.72 (±1.18)
LARGE										
RoBERTa	68.92 (±0.76)	71.88 (±0.26)	67.80 (±1.22)	64.47 (±1.41)	65.73 (±0.80)	62.40 (±0.89)	38.08 (±1.99)	35.97 (±1.74)	67.32 (±0.58)	50.22 (±1.88)
ALBERT	60.62 (±0.57)	59.32 (±0.91)	54.50 (±1.40)	49.27 (±0.64)	54.03 (±0.45)	53.77 (±1.81)	33.90 (±1.22)	31.19 (±3.79)	51.11 (±1.72)	33.12 (±1.24)
UnifiedQA$_{T5}$ *	-	60.90 (±0.00)	-	68.40 (±0.00)	-	65.90 (±0.00)	-	54.40 (±0.00)	-	43.30 (±0.00)
UnifiedQA$_{T5}$	55.28 (±0.00)	61.34 (±0.00)	70.40 (±0.00)	68.40 (±0.00)	69.31 (±0.00)	66.43 (±0.00)	56.61 (±0.00)	54.33 (±0.00)	29.24 (±0.00)	43.74 (±0.00)
UnifiedQA$_{T5\text{-}FT}$	69.00 (±0.51)	**73.60** (±0.45)	70.53 (±0.23)	**68.80** (±0.69)	69.72 (±0.71)	66.92 (±0.85)	**56.84** (±0.39)	**54.42** (±0.15)	66.63 (±1.56)	**58.71** (±0.90)
GenMC$_{T5}$	**71.10** (±0.41)	72.67 (±1.02)	**71.60** (±0.92)	66.87 (±1.33)	**72.49** (±0.77)	**69.01** (±1.97)	49.83 (±2.06)	47.41 (±2.00)	**67.61** (±1.14)	58.06 (±0.92)

Table 3: Comparison with other models. (* indicates the results reported by Khashabi et al. (2020).)

	CSQA		OBQA		ARC-Easy		ARC-Challenge		QASC	
	dev	test	dev	test	dev	test	dev	test	dev	test
BASE										
UnifiedQA$_{T5\text{-}FT}$	56.81 (±0.49)	62.35 (±0.80)	60.80 (±0.72)	58.47 (±0.64)	54.97 (±0.20)	53.88 (±0.39)	45.31 (±0.39)	42.43 (±0.47)	55.57 (±0.58)	43.20 (±0.57)
GenMC$_{T5\text{-}U}$	**61.24** (±0.45)	**63.45** (±0.76)	**62.33** (±0.81)	**59.20** (±1.91)	**61.73** (±0.35)	**59.35** (±0.43)	**45.54** (±0.20)	**43.98** (±0.36)	**60.16** (±0.07)	**45.43** (±0.87)
LARGE										
UnifiedQA$_{T5\text{-}FT}$	69.00 (±0.51)	**73.60** (±0.45)	70.53 (±0.23)	68.80 (±0.69)	69.72 (±0.71)	66.92 (±0.85)	56.84 (±0.39)	54.42 (±0.15)	66.63 (±1.56)	58.71 (±0.90)
GenMC$_{T5\text{-}U}$	**71.58** (±0.25)	72.26 (±0.31)	**71.67** (±0.46)	**69.00** (±0.69)	**73.90** (±0.47)	**72.87** (±0.50)	**59.55** (±1.09)	**55.97** (±0.62)	**68.55** (±0.81)	**58.75** (±0.56)

Table 4: Comparison with UnifiedQA after unifying training sets.

dataset's own training data.

RoBERTa and ALBERT In addition, we compare with two encoder-only models, RoBERTa (Liu et al., 2019) and ALBERT (Lan et al., 2020), which have served as the basis of many MCQA models.

All models are of comparable model size to ours.

5.2.2 Results

The results in Table 3 show that GenMC$_{T5}$ significantly (with p-value < 0.01) outperforms the two encoder-only strong baselines RoBERTa and ALBERT. More interestingly, GenMC$_{T5}$ also performs better than UnifiedQA$_{T5}$ on most datasets. Moreover, for UnifiedQA$_{T5\text{-}FT}$, which further fine-tunes the model on the training set of the target dataset, GenMC$_{T5}$ outperforms it on the test sets of CSQA, OBQA, and ARC-Easy for the base models and ARC-Easy for the large models. It also achieves comparable results on the remaining datasets. These results are impressive because UnifiedQA uses more datasets (i.e., eight different QA datasets) for training. The promising results of GenMC further reveals that our model can learn to effectively extract knowledge from pre-trained encoder-decoders with limited training data.

As a fairer comparison in Table 4, by unifying the training sets of all the five datasets, our GenMC$_{T5\text{-}U}$ outperforms UnifiedQA$_{T5\text{-}FT}$ on all datasets except for CSQA with large models.

5.3 Ablation Study: Influence of Clues

Our main results in Section 5.1 have demonstrated the effectiveness of our model. To better understand its superior results and the influence of our clue generation, we compare with two variants.

5.3.1 Variants of GenMC

Weak Clue We train this variant only using the classification loss $\mathcal{L}^{\text{READ}}$, so only the encoder part is updated, while the decoder part is left untouched from pre-training. Intuitively, under this setting, the generated clue is weaker than GenMC which learns how to generate a clue with supervision.

Token Clue In this setting, we separately train a clue generator and a reader. We first collect the generated clue text C (instead of its representation) from the decoder. We then directly concatenate C with Q and O_i to compute a score for O_i using the model's encoder part stacked with an MLP layer. This variant is indeed very similar to Liu et al. (2020c), which also adopts a pipeline framework to first generate a token-level evidence and then use the evidence to expand the question.

5.3.2 Results

Table 5 shows that masking out generation loss leads to substantial performance drops across all datasets, demonstrating that fine-tuning the decoder with generation loss $\mathcal{L}^{\text{GEN}}$ helps to derive useful clues from pre-trained encoder-decoder models. We also observe that the performance of using token-level clues lags much behind GenMC. This

	CSQA		OBQA		ARC-Easy		ARC-Challenge		QASC	
	dev	test	dev	test	dev	test	dev	test	dev	test
$\text{BART}_{\text{BASE}}$										
GenMC	**54.82** (±0.61)	**56.40** (±0.61)	**58.53** (±0.31)	**57.53** (±2.91)	**59.38** (±1.60)	**56.80** (±0.28)	38.64 (±0.90)	**33.82** (±1.66)	**57.70** (±0.43)	**35.96** (±1.70)
Weak Clue	53.96 (±1.01)	54.35 (±1.97)	55.53 (±1.27)	54.27 (±0.92)	57.20 (±1.80)	55.42 (±1.26)	**39.89** (±0.20)	32.62 (±0.31)	54.05 (±0.21)	25.99 (±0.82)
Token Clue	45.53 (±1.28)	46.41 (±1.79)	54.07 (±1.72)	52.93 (±1.10)	48.97 (±0.91)	48.87 (±1.29)	31.19 (±0.59)	27.64 (±0.69)	49.06 (±0.39)	21.31 (±1.03)
T5_{BASE}										
GenMC	**60.65** (±0.47)	**63.45** (±0.29)	**62.07** (±1.01)	**61.67** (±0.58)	**62.38** (±0.67)	**58.82** (±0.37)	**43.62** (±0.52)	**39.00** (±0.30)	**58.93** (±1.76)	**41.72** (±1.18)
Weak Clue	58.80 (±0.70)	60.88 (±1.89)	61.47 (±0.95)	59.73 (±0.90)	58.97 (±0.54)	57.10 (±0.72)	42.26 (±2.21)	37.54 (±0.64)	57.37 (±1.40)	36.29 (±1.66)
Token Clue	50.55 (±0.44)	48.79 (±0.87)	56.00 (±1.25)	54.93 (±1.63)	46.50 (±0.83)	46.65 (±0.54)	32.66 (±0.20)	26.01 (±1.28)	43.69 (±1.52)	27.50 (±1.56)
$\text{BART}_{\text{LARGE}}$										
GenMC	**69.57** (±0.89)	**72.26** (±0.70)	**68.93** (±1.17)	**68.07** (±1.70)	**72.43** (±0.54)	**68.68** (±0.34)	**48.93** (±0.98)	**45.52** (±1.54)	**68.39** (±0.68)	**55.90** (±0.92)
Weak Clue	67.28 (±2.39)	69.64 (±2.76)	66.20 (±0.53)	64.47 (±1.40)	70.66 (±1.50)	65.71 (±1.47)	27.80 (±2.06)	24.92 (±2.06)	65.68 (±1.31)	52.02 (±1.44)
Token Clue	53.85 (±0.47)	55.23 (±0.62)	61.20 (±3.14)	59.20 (±0.69)	58.02 (±0.98)	54.22 (±1.27)	41.81 (±1.19)	37.60 (±0.90)	48.65 (±1.23)	32.47 (±1.11)
T5_{LARGE}										
GenMC	**71.10** (±0.41)	**72.67** (±1.02)	**71.60** (±0.92)	**66.87** (±1.33)	**72.49** (±0.77)	**69.01** (±1.97)	**49.83** (±2.06)	**47.41** (±2.00)	**67.61** (±1.14)	**58.06** (±0.92)
Weak Clue	68.33 (±1.62)	71.66 (±1.28)	69.27 (±0.42)	65.87 (±0.90)	69.66 (±0.77)	66.24 (±0.79)	47.57 (±2.04)	46.24 (±1.29)	64.99 (±0.74)	53.35 (±1.35)
Token Clue	59.47 (±0.08)	60.74 (±0.29)	62.80 (±1.44)	57.73 (±1.10)	48.85 (±1.62)	48.36 (±2.15)	37.97 (±0.90)	30.50 (±1.46)	49.22 (±0.62)	38.77 (±1.74)

Table 5: Influence of clues.

Clue Type	Percentage	Example	
		Instance	Clue
Irrelevant	23.60%	Which would you likely find inside a beach ball? (A) cheese (B) *steam* (C) water (D) **air**	a squid
Relevant but unhelpful	52.40%	What may have been formed by a volcano? (A) **Mt. McKinley** (B) Lake Pontchartrain (C) The great lakes (D) *Niagara Falls*	a lake
Helpful	24.00%	Where would there be an auditorium with only a single person speaking? (A) lights (B) crowd (C) **university campus** (D) *theater* (E) park	school

Table 6: Distribution of clue types in negative cases with examples. Bold underline indicates the correct answer, and italic indicates the predicted label.

demonstrates that naively using explicit knowledge in plain text, instead of using implicit clues from the decoder's hidden state, is inferior as it may unnecessarily bring information loss and noise.

5.4 Error Analysis

We analyze the clues generated by GenMC using T5_{LARGE} with a focus on instances that are correctly predicted by the baseline in our main experiments (i.e., T5_{LARGE} + $\text{Text2Text}_{\text{vanilla}}$), while our GenMC fails. The intuition is that in these *negative cases*, the clues generated by GenMC may play a negative role. By studying these potentially negative clues, we can gain more insights into how GenMC fails and discuss venues for future improvement.

Specifically, we randomly sample 50 negative cases from T5_{LARGE} + GenMC for each dataset. We show six graduate students of computer science[4] an instance along with the generated clue, correct answer, and predicted answer. We then ask them to categorize clues into the following families:[5]

- **Irrelevant:** The clue is off topic or is not understandable.
- **Relevant but unhelpful:** Though relevant, the clue makes a factually incorrect statement, often on the contrary of the main question, or the clue contributes relevant but insufficient knowledge for prediction, such as repetition of the question or other distractors.
- **Helpful:** The clue adds helpful information to answer the question.

To ensure the annotation quality, we aggregate annotated results from three students for every dataset using majority vote. If all three students annotate differently from each other for an instance, we introduce a fourth student to arbitrate.

Table 6 shows the percent of each clue type across all datasets with an example for each type. Figure 4 breaks down by dataset. Though the majority of our clues are relevant (i.e., 76.4% of them are relevant across all datasets), which seems positive, only 24% of the clues are deemed as helpful. This suggests a great room for improvement. In our future research, we will focus on how to generate more helpful clues from questions.

[4]They are volunteers recruited from the contact author's research group. They know and agree that their annotations will be used for error analysis in a research paper.

[5]We follow a similar definition by Shwartz et al. (2020).

	CSQA	OBQA	ARC-Easy	ARC-Challenge	QASC
$T5_{BASE}$					
$Text2Text_{vanilla}$	0.040 (±0.007)	0.035 (±0.002)	0.035 (±0.002)	0.039 (±0.004)	0.035 (±0.002)
UnifiedQA	0.059 (±0.041)	0.089 (±0.047)	0.097 (±0.055)	0.129 (±0.075)	0.068 (±0.027)
GenMC	0.069 (±0.019)	0.107 (±0.046)	0.113 (±0.060)	0.121 (±0.053)	0.072 (±0.027)
$T5_{LARGE}$					
$Text2Text_{vanilla}$	0.077 (±0.008)	0.083 (±0.012)	0.081 (±0.012)	0.084 (±0.014)	0.078 (±0.011)
UnifiedQA	0.108 (±0.037)	0.178 (±0.096)	0.190 (±0.107)	0.257 (±0.127)	0.130 (±0.052)
GenMC	0.105 (±0.027)	0.178 (±0.078)	0.219 (±0.120)	0.242 (±0.112)	0.127 (±0.048)

Table 7: Inference time for answering a question (seconds).

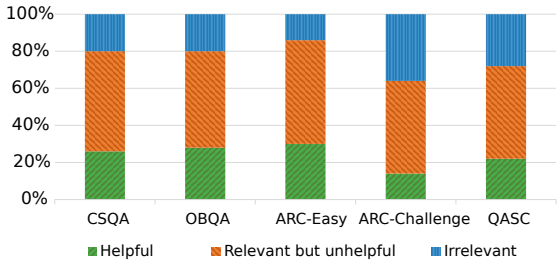


Figure 4: Distribution of clue types in negative cases on each dataset.

5.5 Inference Time and Model Size

Table 7 shows the inference time for answering a question. GenMC is slower than $Text2Text_{vanilla}$, but their inference time has the same scale, suggesting that GenMC is more cost-effective considering its superior accuracy. GenMC and UnifiedQA are comparable in inference time.

Among $T5_{BASE}$ based models, $Text2Text_{vanilla}$ and UnifiedQA have 223 M parameters, while GenMC is slightly larger with 234 M parameters. Among $T5_{LARGE}$ based models, $Text2Text_{vanilla}$ and UnifiedQA have 738 M parameters, while GenMC has 757 M parameters.

6 Conclusion

We present GenMC, a simple yet effective model which tailors pre-trained encoder-decoders for MCQA tasks. Compared with existing usages of pre-trained encoder-decoders for MCQA, our model fully exploits the pre-trained encoder-decoders' NLG capabilities to generate a clue from the input question, which facilitates deep understanding of question-option pairs. Experimental results further verify the superiority of GenMC over existing usages. Notably, our model achieves promising results without using any provided documents or external corpora, showing an interesting application of PLMs by directly inducing either commonsense or scientific knowledge from them through clue generation.

In the future, we will focus on how to further improve the clue generation quality, which remains a bottleneck of GenMC. We hope this work will spur more research in how to better use pre-trained encoder-decoders for not only MCQA, but also beyond; for tasks with divergent structures from the pre-training, a smarter use of PLMs can boost the performance significantly.

Acknowledgments

This work was supported in part by the NSFC (62072224) and in part by the Beijing Academy of Artificial Intelligence (BAAI).

References

Zhipeng Chen, Yiming Cui, Wentao Ma, Shijin Wang, and Guoping Hu. 2019. Convolutional spatial attention model for reading comprehension with multiple-choice questions. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6276–6283. AAAI Press.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Zixian Huang, Yulin Shen, Xiao Li, Yuang Wei, Gong Cheng, Lin Zhou, Xinyu Dai, and Yuzhong Qu. 2019.

Geosqa: A benchmark for scenario-based question answering in the geography domain at high school level. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 5865–5870. Association for Computational Linguistics.

Zixian Huang, Ao Wu, Yulin Shen, Gong Cheng, and Yuzhong Qu. 2021. When retriever-reader meets scenario-based multiple-choice questions. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 985–994. Association for Computational Linguistics.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know. *Trans. Assoc. Comput. Linguistics*, 8:423–438.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. Unifiedqa: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 1896–1907. Association for Computational Linguistics.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. QASC: A dataset for question answering via sentence composition. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8082–8090. AAAI Press.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2019. What's missing: A knowledge gap guided approach for multi-hop question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2814–2828. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Veronica Latcinnik and Jonathan Berant. 2020. Explaining question answering models through text generation. *CoRR*, abs/2004.05569.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. 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, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Xiao Li, Gong Cheng, Ziheng Chen, Yawei Sun, and Yuzhong Qu. 2022. Adalogn: Adaptive logic graph network for reasoning-based machine reading comprehension. *CoRR*, abs/2203.08992.

Xiao Li, Yawei Sun, and Gong Cheng. 2021. TSQA: tabular scenario based question answering. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 13297–13305. AAAI Press.

Dayiheng Liu, Yeyun Gong, Jie Fu, Yu Yan, Jiusheng Chen, Daxin Jiang, Jiancheng Lv, and Nan Duan. 2020a. RikiNet: reading Wikipedia pages for natural question answering. In *ACL*, pages 6762–6771.

Frederick Liu, Siamak Shakeri, Hongkun Yu, and Jing Li. 2021. Enct5: Fine-tuning T5 encoder for non-autoregressive tasks. *CoRR*, abs/2110.08426.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020b. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3622–3628. ijcai.org.

Ye Liu, Tao Yang, Zeyu You, Wei Fan, and Philip S. Yu. 2020c. Commonsense evidence generation and injection in reading comprehension. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue, SIGdial 2020, 1st virtual meeting, July 1-3, 2020*, pages 61–73. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2021. Generation-augmented retrieval for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4089–4100. Association for Computational Linguistics.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the 2018 Conference on*

Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018, pages 2381–2391. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2463–2473. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4932–4942. Association for Computational Linguistics.

Qiu Ran, Peng Li, Weiwei Hu, and Jie Zhou. 2019. Option comparison network for multiple-choice reading comprehension. *CoRR*, abs/1903.03033.

Yunfan Shao, Zhichao Geng, Yitao Liu, Junqi Dai, Fei Yang, Li Zhe, Hujun Bao, and Xipeng Qiu. 2021. CPT: A pre-trained unbalanced transformer for both chinese language understanding and generation. *CoRR*, abs/2109.05729.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4222–4235. Association for Computational Linguistics.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Unsupervised commonsense question answering with self-talk. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4615–4629. Association for Computational Linguistics.

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2019. Improving machine reading comprehension with general reading strategies. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2633–2643. Association for Computational Linguistics.

Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, Weixin Liu, Zhihua Wu, Weibao Gong, Jianzhong Liang, Zhizhou Shang, Peng Sun, Wei Liu, Xuan Ouyang, Dianhai Yu, Hao Tian, Hua Wu, and Haifeng Wang. 2021. ERNIE 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *CoRR*, abs/2107.02137.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4149–4158. Association for Computational Linguistics.

Min Tang, Jiaran Cai, and Hankz Hankui Zhuo. 2019. Multi-matching network for multiple choice reading comprehension. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7088–7095. AAAI Press.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2020. Reclor: A reading comprehension dataset requiring logical reasoning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Shuailiang Zhang, Hai Zhao, Yuwei Wu, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. 2020. DCMN+: dual co-matching network for multi-choice reading comprehension. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI*

Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, pages 9563–9570. AAAI Press.

Wangchunshu Zhou, Dong-Ho Lee, Ravi Kiran Selvam, Seyeon Lee, and Xiang Ren. 2021. Pre-training text-to-text transformers for concept-centric common sense. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Pengfei Zhu, Hai Zhao, and Xiaoguang Li. 2020. Dual multi-head co-attention for multi-choice reading comprehension. *CoRR*, abs/2001.09415.

Responsible NLP Research Checklist

Members of the ACL are responsible for adhering to the ACL code of ethics. The ARR Responsible NLP Research checklist is designed to encourage best practices for responsible research, addressing issues of research ethics, societal impact and reproducibility.

Please read the Responsible NLP Research checklist guidelines for information on how to answer these questions. Note that not answering positively to a question is not grounds for rejection.

All supporting evidence can appear either in the main paper or the supplemental material. For each question, if you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Please do not modify, reorder, delete or add questions, question options or other wording of this document.

A For every submission

A1 Did you discuss the *limitations* of your work?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification Section 5.4, Section 6

A2 Did you discuss any potential *risks* of your work?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

N/A

Section or justification Click or tap here to enter text.

A3 Do the abstract and introduction summarize the paper's main claims?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification Abstract, Section 1

B Did you use or create *scientific artifacts*?

If you answer **Yes**, provide the section number; if you answer **No**, you can skip the rest of this section.

Yes

If yes:

B1 Did you cite the creators of artifacts you used?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification Section 4.1

B2 Did you discuss the *license or terms* for use and/or distribution of any artifacts?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification Section 1

B3 Did you discuss if your use of existing artifact(s) was consistent with their *intended use*, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification Section 4.1

B4 Did you discuss the steps taken to check whether the data that was collected/used contains any *information that names or uniquely identifies individual people* or *offensive content*, and the steps taken to protect / anonymize it?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

No

Section or justification They are widely used datasets containing commonsense/scientific information.

B5 Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification Section 4.1

B6 Did you report relevant statistics like the number of examples, details of train/test/dev splits, etc. for the data that you used/created?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification Section 4.1

C Did you run *computational experiments*?

If you answer **Yes**, provide the section number; if you answer **No**, you can skip the rest of this section.

Yes

If yes:

C1 Did you report the *number of parameters* in the models used, the *total computational budget* (e.g., GPU hours), and *computing infrastructure* used?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification Section 4.2, Section 5.5

C2 Did you discuss the experimental setup, including *hyperparameter search* and *best-found hyperparameter* values?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification Section 4.2

C3 Did you report *descriptive statistics* about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification Section 4.2, Section 5

C4 If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification Section 4.2

D Did you use *human annotators* (e.g., crowdworkers) or *research with human subjects*?

If you answer **Yes**, provide the section number; if you answer **No**, you can skip the rest of this section.

Yes

If yes:

D1 Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification Section 5.4

D2 Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such *payment is adequate* given the participants' demographic (e.g., country of residence)?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification Section 5.4

D3 Did you discuss whether and how consent was obtained from people whose data you're using/curating (e.g., did your instructions explain how the data would be used)?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification Section 5.4

D4 Was the data collection protocol *approved* (or *determined exempt*) by an ethics review board?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

N/A

Section or justification Click or tap here to enter text.

D5 Did you report the basic demographic and geographic characteristics of the *annotator* population that is the source of the data?

If you answer **Yes**, provide the section number; if you answer **No**, provide a justification.

Yes

Section or justification Section 5.4