Answering Ambiguous Questions via Iterative Prompting

Weiwei Sun[1] **Hengyi Cai**[2] **Hongshen Chen**[2] **Pengjie Ren**[1]
Zhumin Chen[1] **Maarten de Rijke**[3] **Zhaochun Ren**[1]*
[1]Shandong University, Qingdao, China
[2]JD.com, Beijing, China
[3]University of Amsterdam, Amsterdam, The Netherlands
{sunnweiwei,hengyi1995}@gmail.com
ac@chenhongshen.com,m.derijke@uva.nl
{renpengjie,chenzhumin,zhaochun.ren}@sdu.edu.cn

Abstract

In open-domain question answering, due to the ambiguity of questions, multiple plausible answers may exist. To provide feasible answers to an ambiguous question, one approach is to directly predict all valid answers, but this can struggle with balancing relevance and diversity. An alternative is to gather candidate answers and aggregate them, but this method can be computationally costly and may neglect dependencies among answers. In this paper, we present *AmbigPrompt* to address the imperfections of existing approaches to answering ambiguous questions. Specifically, we integrate an answering model with a prompting model in an iterative manner. The prompting model adaptively tracks the reading process and progressively triggers the answering model to compose distinct and relevant answers. Additionally, we develop a task-specific post-pretraining approach for both the answering model and the prompting model, which greatly improves the performance of our framework. Empirical studies on two commonly-used open benchmarks show that AmbigPrompt achieves state-of-the-art or competitive results while using less memory and having a lower inference latency than competing approaches. Additionally, AmbigPrompt also performs well in low-resource settings. The code are available at: https://github.com/sunnweiwei/AmbigPrompt.

1 Introduction

Recent years have witnessed substantial advances in open-domain question answering (QA) systems (Karpukhin et al., 2021; Lewis et al., 2020; Izacard and Grave, 2021b), which aim to find the answer for the given question from a large knowledge corpus (Chen et al., 2017). While a dominating scenario is the single-answer QA setting, i.e., only one exact answer is required for a given question (Karpukhin et al., 2021), this work focuses on the more realistic scenario of *Multi-answer QA*, where multiple plausible answers are associated with a user-issued question (Min et al., 2020), given that questions posed by humans are often open-ended and ambiguous.[1]

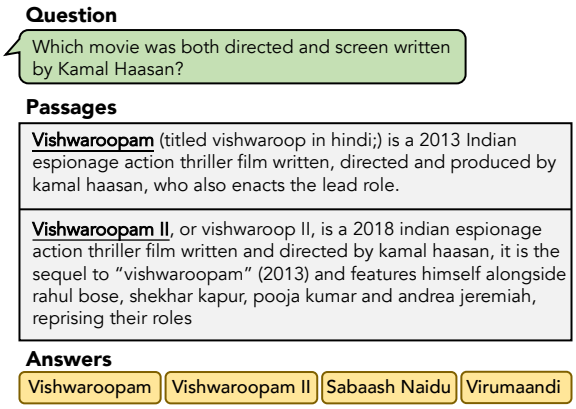

Figure 1: An example of an open-domain question, a subset of its evidential Wikipedia passages and multiple answers they lead to.

A natural approach for answering ambiguous open-domain questions would be to fine-tune a pre-trained answer generation model, e.g., T5 (Raffel et al., 2020), using supervised data of the form (evidential passages, question, all plausible answers) (Min et al., 2020, 2021). However, this approach often leads to sub-optimal solutions since it requires the model to balance the relevance and diversity of the generated multiple answers within a single-round decoding procedure, which is non-trivial. To manage the relevance-diversity trade-off, another approach is to decompose multi-answer QA into candidate answer prediction and answer post-processing. This typically requires a high-capacity model with billions of parameters

*Corresponding author.

[1]The task of this paper primarily focuses on the occurrence of multiple answers resulting from different interpretations caused by question ambiguity. However, it's worth to note that question ambiguity is just one factor contributing to the presence of multiple answers. In this study, we adhere to the conceptual definition of Min et al. (2020).

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 7669–7683
July 9-14, 2023

to construct candidate answers and sophisticated answer aggregation pipelines to obtain the final results (Shao and Huang, 2022; Gao et al., 2021b), incurring high computational costs. In addition, this approach suffers from the dilemma of having to predict diverse candidate answers before knowing which answer has been predicted, which is unnatural and intricate. For example, in Figure 1, given the question "*Which movie was both directed and screenwritten by Kamal Haasan?*," with the existence of the answer *Vishwaroopam*, the model excludes its eponymous translation version *Vishwaroop* and deduces that *Vishwaroopam II* is another potential answer.

When facing an ambiguous question, people are capable of providing multiple valid answers by introspectively composing new content on the basis of what has already been devised, usually in an iterative manner. Inspired by this observation, in this paper, we conceptualize **AmbigPrompt** as an approach to mimic this mechanism by iteratively guiding the answering model with a lightweight prompting model. As shown in Figure 2, this prompting model steers the answering model to progressively generate valid answers whose content the prompting model will then condition on for the next-round prompt construction. Essentially, our proposed framework comprises two key components: (i) an encoder-decoder *answering model* and (ii) an interleaving answer-conditional *prompting model*. By conditioning on preceding generated contents, the proposed framework introspectively perceives which answer has been predicted before updating the hidden activation for the generation of subsequent answers. Furthermore, we devise a task-adaptive post-pretraining strategy, in which pseudo multi-QA training instances are constructed to facilitate the training of the proposed framework.

We carry out extensive experiments on the AmbigQA (Min et al., 2020) and WebQSP (tau Yih et al., 2016) datasets. The results demonstrate that AmbigPrompt attains superior performance despite having a significantly smaller parameter scale, 14 times less than state-of-the-art models. Furthermore, as a lightweight approach, AmbigPrompt improves the answer relevance and diversity with a tiny fraction of the memory footprint and inference latency of competing approaches. Notably, AmbigPrompt achieves the best performance in the low-resource setting. The effectiveness of the proposed method is also verified by ablation experiments and analytical experiments.

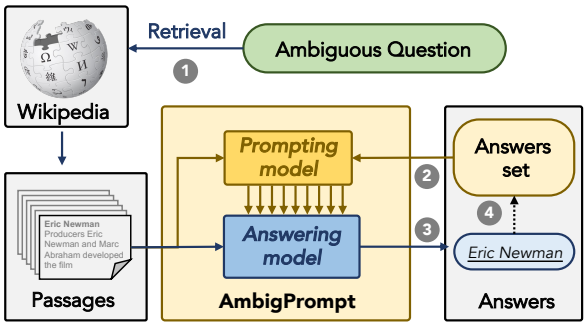

Figure 2: Given the retrieved passages, AmbigPrompt alternates between (2) generating prompts based on previous answers, (3) generating a new answer using a question-answering model, and (4) appending the new answer to the answers set. Note that steps (2) and (3) operate in an interleaving way.

In summary, this paper makes the following contributions: (i) We propose AmbigPrompt, which tackles ambiguous question answering by iterative prompting. (ii) We propose an interleaving answer–conditional prompting model to generate meaningful continuous prompts. (iii) Experiments on multi-QA datasets verify the effectiveness of the proposed approach.

2 Preliminaries

2.1 Problem formalization

Formally, given an open-domain question q, a multi-answer question answering (QA) model is required to make use of (multiple pieces of) evidence from a large-scale text corpus Ω (e.g., Wikipedia) to find multiple plausible answers $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$, where a_i denotes one answer and we suppose there are n answers. The QA model aims to infer $p(\mathcal{A}|q, \Omega)$. In open-domain QA, the QA model typically follows a two-step pipeline, comprising *passage retrieval* and *answer generation*. In the passage retrieval step, a retrieval model $p(\mathcal{C}|q, \Omega)$ retrieves m evidence passages $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ according to the question q from Ω. In the answer generation step, an answering model $p(\mathcal{A}|q, \mathcal{C})$ reads the evidential passages and finds the answers to the question.

2.2 Answering model

We use Fusion-in-Decoder (FiD) as a basic single-answer answering model (Izacard and Grave, 2021b). In particular, FiD has an encoder-decoder architecture. FiD first concatenates each retrieved

passage with the question with a [SEP] token:

$$X = \{x_1, x_2, \ldots, x_m\},\ x_i = q\,\texttt{[SEP]}\,c_i \quad (1)$$

where we use X to denote the concatenated sequence. Then, for each x_i, the FiD encoder Enc encodes it to $\mathbf{x}_i$:

$$\mathbf{X} = \mathrm{Cat}(\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}),\ \mathbf{x}_i = \mathrm{Enc}(x_i) \quad (2)$$

where Cat denotes a concatenation function. Finally, the decoder Dec attends to the representations of all passages and generates an answer a:

$$p(a|q, \mathcal{C}) = \mathrm{Dec}(\mathbf{X}) \quad (3)$$

2.3 Prompt-tuning

Prompt-tuning adapts pre-trained transformer models to downstream tasks by optimizing continuous prompting vectors (Li and Liang, 2021; Liu et al., 2022). Suppose x is the input sequence of the model, we denote $Q(x)^j$, $K(x)^j$, $V(x)^j$ as the query, key, and value representations of x in the j-th attention layer in the transformer encoder. Prompt-tuning prepends learnable prompting vectors $\mathbf{E}^j$ to $K(x)^j$ and $V(x)^j$ to modify the attention distribution as well as the output $\mathbf{x}^j$ of the j-th layer as follows:

$$\begin{aligned}\mathbf{x}^j = \mathrm{Attn}(Q(x)^j, \mathrm{Cat}(\mathbf{E}^j, K(x)^j),\\ \mathrm{Cat}(\mathbf{E}^j, V(x)^j)),\end{aligned} \quad (4)$$

where $\mathbf{x}^j$ denotes the output of layer j, $\mathrm{Attn}(\cdot)$ represents the attention operation in the transformer, and $\mathrm{Cat}(\cdot)$ is the concatenation function.

3 AmbigPrompt

Conventionally, the question answering model generates the desired answer given the input context in a single pass (Izacard and Grave, 2021b). While it suffices to tackle the single-answer QA scenario, managing ambiguous questions with multiple answers can be more nuanced – the answering model is required to balance the relevance and diversity of the generated answers in a single pass, and precisely modeling dependencies among the answers can be non-trivial. In this paper, we propose AmbigPrompt, a question-answering model that answers ambiguous questions via iterative prompting, inferring more accurate answers progressively. Figure 2 gives an overview of the proposed method.

Overall, AmbigPrompt decomposes the generation of answers $\mathcal{A}$ into multiple steps instead of one single pass, i.e.,

$$p(\mathcal{A}|q, \mathcal{C}) = \prod_{t=1}^{n} p(a_t|\phi(a_{<t}), q, \mathcal{C}), \quad (5)$$

where $a_{<t}$ denotes the set of answers that have been generated at time t, and $\phi(\cdot)$ denotes a prompting model that generates prompt vectors for answer generation at the t-th step. The prompting model shares parameters with the answering model, allowing for seamless integration. AmbigPrompt iteratively composes a new answer a_t, conceiving the prompt of previous answers, i.e., $\phi(a_{<t})$, and appends a_t to the answers set, till all feasible answers are found.

The proposed framework is optimized in a two-stage manner: *task-adaptive post-pretraining* and *prompt-based tuning*. In the former stage, the model is trained on a large synthesized multi-answer QA dataset, while in the latter stage, the model is tuned on the annotated multi-answer QA dataset. We first detail the prompting model (§3.1) and the iterative question answering procedure (§3.2), and then introduce the optimization scheme (§3.3).

3.1 Retrospective prompting mechanism for answer generation

To capture intricate dependencies among answers, we devise an interleaving answer-conditional prompting model $\phi(a_{<t})$, which generates the prompt vector $\mathbf{E} = \phi(a_{<t})$ conditioned on antecedent generated answers $a_{<t}$, as depicted in Figure 3. Specifically, the prompting model ϕ is a transformer encoder that shares the same parameters with the encoder of the answering model. ϕ processes the $a_{<t}$ in three steps:

(1) **Templating answers.** First, $a_{<t}$ is transformed into a text sequence $e = \mathcal{T}(a_{<t})$ using a template $\mathcal{T}$. Here we use semicolons to splice answers.

(2) **Generating prompts.** Then, given the answer sequence e and context X (i.e., the concatenated question and passages in Eq. 1), the prompting model ϕ computes the hidden activations $\mathbf{E}^j$ of each layer j via cross-attending the contextual representation $\mathbf{X}^{j-1}$:

$$\begin{aligned}\mathbf{E}^j = \mathrm{Attn}(Q(e)^j, \mathrm{Cat}(K(e)^j, \mathbf{X}^{j-1}),\\ \mathrm{Cat}(V(e)^j, \mathbf{X}^{j-1})),\end{aligned} \quad (6)$$

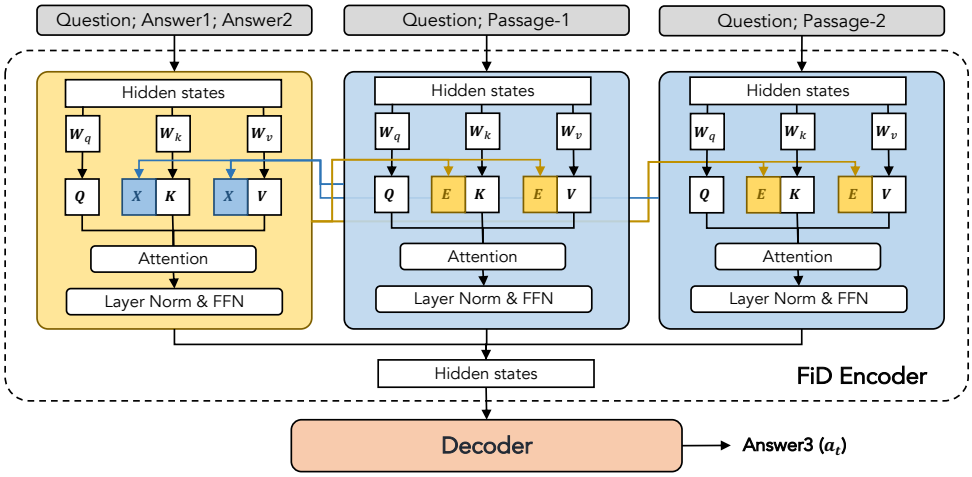

Figure 3: Details of the retrospective prompting mechanism. The prompting model produces the prompt vectors $\mathbf{E}$ by cross-attending the contextual representation $\mathbf{X}$. And the answering model predicts a new answer a_t using the prompt $\mathbf{E}$. The prompting and answering models operate in an interleaving manner.

where $Q(e)^j$, $K(e)^j$, and $V(e)^j$ denote the query, key, and value representations of e in the j-th attention layer in the prompting model; $\mathbf{X}^{j-1} = \mathrm{Cat}(\{\mathbf{x}_1^{j-1}, \mathbf{x}_2^{j-1}, \ldots, \mathbf{x}_m^{j-1}\})$ denotes the concatenated context representations of the $(j-1)$-th layer in the answering model. We write $\mathbf{E}$ for the last layer output of the prompting model.

(3) **Prompting answering model.** Finally, the generated prompt $\mathbf{E}^j$ is prepended to the attention layer of the encoder Enc of the answering model as in Eq. 4. Meanwhile, the decoder Dec of answering model attends to $\mathrm{Cat}(\mathbf{E}, \mathbf{X})$ and generates the target answer a_t:

$$p(a_t|\phi(a_{<t}), q, \mathcal{C}) = \mathrm{Dec}(\mathrm{Cat}(\mathbf{E}, \mathbf{X})). \quad (7)$$

Capturing long-range dependencies among derived answers via a retrospective prompting mechanism enables the answering model to compose new contents grounding on what has already been devised, and thus the model is able to strike a good relevance-diversity balance for answering ambiguous questions.

3.2 Answering ambiguous questions via iterative prompting

Given the input context, i.e., the question and retrieved evidential passages, AmbigPrompt iteratively performs attention operations over the input context and the generated answers, addressing the answer generation and prompt construction interactively. The key is to pass the attention activations between the prompting model and answering model so that they can inspect each other's internal states and make harmonious predictions. Specifically, we start from an empty answer set and progressively append newly generated answers to it. As depicted in Figure 2, in each iteration, we first use the previously generated answer sequence to obtain the introspective prompts, and then interwoven the resultant prompting vectors into the answering model to predict the next answer. Our algorithm terminates if the model reaches the `[EOI]` token.

3.3 Optimization

To enhance the pre-training model towards multi-answer QA, one straightforward approach is to leverage a question-answering dataset such as NQ (Kwiatkowski et al., 2019) for domain-adaptive pre-training (Min et al., 2021). However, the effectiveness of such a trivial approach is limited to the inherent defect of the one-pass prediction process; that is, the lack of the modeling capability of the interactions between answer generation and answer perception, which is critical to achieving superior performance in multi-QA scenarios. To explicitly align the pre-training objective to task-specific preferences, we further propose to conduct task-adaptive post-pretraining on pseudo multi-answer QA dataset, and then finetune the proposed model using the task data.

Task-adaptive post-pretraining. We first pre-train the model on NQ, in which only one answer $\mathcal{A} = \{a_1\}$ is labeled for each question q. To ex-

plicitly characterize the pretraining stage as the efforts for finding *which part of preceding answers to interact with regarding the input context*, we construct the pseudo multi-answer dataset $\hat{\mathcal{A}}$ for post-pretraining the proposed framework to mimic the iterative question answering process. Specifically, we first train an *auxiliary reader* $g(a|q, c_i)$, which learns to find an answer from the passage c_i given a question q. Then, we use this auxiliary reader to generate a pseudo answer for each retrieved passage in $\mathcal{C}$:

$$\hat{\mathcal{A}} = \{\hat{a} \mid \forall i \in [1, m], \hat{a} \sim g(a|q, c_i)\}, \quad (8)$$

where $\hat{\mathcal{A}}$ denotes the pseudo-answer set of q.

Then, we aggregate the generated answers to construct the previously known answers $a_{<t}$ in Eq. 5. In particular, we randomly sample t answers from $\hat{\mathcal{A}}$ and filter out those that are equivalent to the ground-truth answer a_1; we denote the sampled set as $\hat{a}_{<t}$. With the pseudo answers, we define the post-pretraining objective as:

$$\mathcal{L}_{\text{Pre}} = -\log p(a_1|\phi(\hat{a}_{<t}), q, \mathcal{C}), \quad (9)$$

where the number of answers in $\hat{a}_{<t}$, i.e., t, is sampled from a Bernoulli distribution.

Prompt-based fine-tuning. We fine-tune the pretrained model on downstream multi-answer QA datasets. Specifically, in multi-answer QA, n answers $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ corresponding to a question q are provided. The model is tuned by the following objective:

$$\mathcal{L}_{\text{FT}} = -\log p(a_t|\phi(a_{<t}), q, \mathcal{C}), \quad (10)$$

where $t \in [1, n]$ is sampled from a Bernoulli distribution. Since $\mathcal{A}$ is unordered, we shuffle $\mathcal{A}$ when constructing the $a_{<t}$ and a_t to improve the robustness. Besides, we explicitly optimize the model to generate `[EOI]` to stop the iteration. Specifically, we define a parameter $\alpha \sim \mathcal{U}(0, 1)$ and a threshold λ, which controls the propensity of generating `[EOI]`. If $\alpha < \lambda$, we replace the a_t and $a_{<t}$ as `[EOI]` and $\mathcal{A}$, respectively.

4 Experimental Setup

4.1 Datasets

We evaluate AmbigPrompt on the AmbigQA (Min et al., 2020) and WebQSP (tau Yih et al., 2016) datasets. **AmbigQA**: AmbigQA is constructed to address the ambiguity of questions in open-domain QA. It samples 14,042 questions from NQ-Open (Kwiatkowski et al., 2019), and asks annotators to search for, navigate and read multiple Wikipedia pages to find as many answers as possible. **WebQSP**: WebQSP consists of questions from Google Suggest API, originally from Berant et al. (2013). The answer is a set of distinct entities in Freebase; we use the modified versions by Min et al. (2021), which recasts WebQSP as textual question answering based on Wikipedia.

	NQ	AmbigQA	WebQSP
Training size	307,373	10,036	2,752
Validation size	6,000	2,002	245
Test size	6,000	2,004	1,582
Mean # Answers	1.0	2.2	22.6
Median # Answers	1.0	2.0	1.0

Table 1: Data statistics of NQ, AmbigQA, and WebQSP.

The statistical details of these two datasets and NQ are shown in Table 1.

4.2 Evaluation metrics

Following previous studies (Min et al., 2020), we adopt F1 as the evaluation metric, which measures the precision and recall between the ground-truth answers and the predicted answers. The test set is further divided into two subsets: *full* and *multi*. The *full* subset evaluates the model on all the questions in the test set, while the *multi* subset evaluates the model on the questions with multiple answers (i.e., $n > 1$). To assess the computational efficiency of various approaches, we also report the number of parameters, average latency, and peak memory usage during model inference. All the models are tested on the same device. We estimate the latency and memory usage of those baselines without public code using randomly initialized models since these metrics are independent of their parameters given a fixed number of encoded tokens and decoding length.

4.3 Baselines

The following models are adopted as baselines: **DPR** (Karpukhin et al., 2021): A dual-encoder is trained using contrastive loss for passage retrieval, and a BERT-based reader is used for answer extraction. **SpanSeqGen** (Min et al., 2020): DPR reranks the passages, and a BART-based generator is used for answer generation. **FiD** (Izacard and Grave, 2021b): The retrieved passages are encoded by a T5 encoder independently, and the representations are then concatenated and fed into the T5 Decoder to generate answers. **Refuel** (Gao et al.,

Methods	AmbigQA		WebQSP		#Params		Latency		Memory	
	Full	Multi	Full	Multi						
High-capacity baselines										
JPR† (Min et al., 2021)	48.5	37.6	53.1	47.2	3B	8.7×	0.88s	2.3×	14GB	3.5×
RECTIFY† (Shao and Huang, 2022)	52.1	41.6	55.8	48.8	6B	17.4×	19.72s	51.3×	14GB	3.5×
Comparable low-capacity baselines										
DPR (Karpukhin et al., 2021)	38.9	29.9	44.7	35.5	345M	1.0×	0.37s	1.0×	4GB	1.0×
SpanSeqGen (Min et al., 2020)	39.7	29.3	48.8	36.1	400M	1.2×	0.49s	1.3×	5GB	1.3×
FiD-Base (Izacard and Grave, 2021b)	45.5	35.8	52.6	46.3	220M	0.6×	0.38s	1.0×	4GB	1.0×
Refuel† (Gao et al., 2021b)	48.3	37.3	–	–	400M	1.2×	22.19s	58.6×	8GB	2.0×
AmbigPrompt	**48.7**	**38.8**	**53.2**	**47.9**	220M	0.6×	0.68s	1.8×	4GB	1.0×

Table 2: Results on AmbigQA dev and WebQSP test in terms of effectiveness and efficiency. Full and Multi denote the full set and multi-answer set, respectively. The reported value is F1. Methods with † have no publicly available codes; therefore, we estimate the latency and memory footprint with randomly initialized parameters. We divide baselines into two groups: (i) *high-capacity baselines* that use significantly larger models than AmbigPrompt, and (ii) *comparable low-capacity baselines* that use a low-capacity model like AmbigPrompt and can be reasonably compared with AmbigPrompt. **Boldface** indicates best performance among comparable baselines.

2021b): A question disambiguation module is proposed to generate disambiguated questions. The disambiguated questions are then used to find more answers. **JPR** (Min et al., 2021): JPR is a passage reranker that reranks the passages using an autoregressive model. With the additional reranking stage, JPR selects ten diverse passages from 100 retrieved passages and uses a T5-3B FiD answering model to compose answers in one pass. **RECTIFY** (Shao and Huang, 2022): RECTIFY proposes the recall-then-verify framework, which separates the reasoning process of each answer. An answering model operates on each passage to recall surplus answers. Then, a sophisticated verifier based on T5-3B FiD verifies each answer with an aggregation module.

We divide the baseline models into two categories depending on the number of parameters of the models: (i) *high-capacity baselines* that use large models with billions of parameters, while requiring more computational resources and memory; (ii) *comparable low-capacity baselines* that use low-capacity models with a similar number of parameters and computational effort as AmbigPrompt, which can be reasonably compared with AmbigPrompt.

4.4 Implementation details

We choose T5-Base (Raffel et al., 2020) as the backbone of the answering model. Regarding the passage retrieval model, we fine-tune the pre-trained model from Gao and Callan (2021) on the NQ dataset (See Appendix C for details). The retrieval corpus is the English Wikipedia on 12/20/2018, and the documents are split into chunks with 100 words following Karpukhin et al. (2021). We set m=100, λ=0.5, the batch size to 32, and the model is trained using the AdamW optimizer (Loshchilov and Hutter, 2017) with a constant learning rate of $5e{-}5$. We train the model up to 5k steps on on 4 V100-16G GPUs and choose the hyperparameters and checkpoints on the validation set.[2]

5 Experimental Results

5.1 Main results

Table 2 reports the evaluation results on AmbigQA and WebQSP. Based on the results, we have three main observations.

First, AmbigPrompt achieves comparable performance to the state-of-the-art. Specifically, AmbigPrompt obtains 48.7 F1 on the *full* test set and 38.8 F1 on the *multi* test set, which exceeds all baselines except RECTIFY. The improvements are particularly significant on the *multi* test set; AmbigPrompt improves 1.2% over JPR and 1.5% over Refuel. Besides, compared with FiD, which concatenates all the answers in $\mathcal{A}$ with `[SEP]` and generates them in one pass, the proposed method, which benefits from the iterative design and answer-conditional prompting mechanism, achieves 3% and 5% improvements on *full* and *multi* of AmbigQA. Similar results can also be observed on WebQSP.

Second, AmbigPrompt uses fewer resources compared to previous high-capacity models. AmbigPrompt uses a lightweight model with 220M

[2]Since we test on the AmbigQA dev set, we slice about 1k examples in the AmbigQA training set as the validation set.

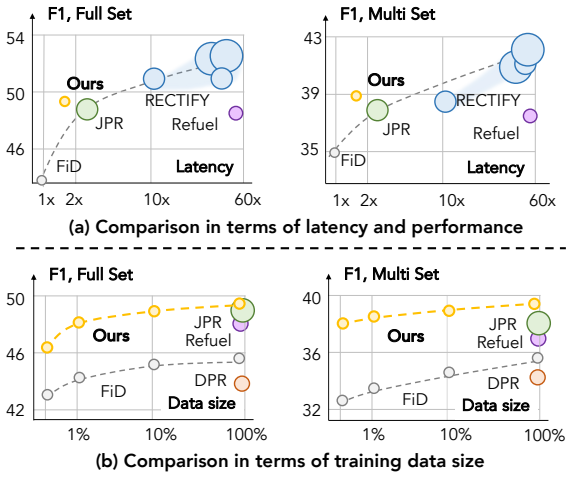

Figure 4: (a) Latency (in log scale) versus performance (F1) on AmbigQA full dev and multi dev. The size of the circle indicates the number of parameters of these models. (b) Dataset size (in %) versus performance (F1) on AmbigQA full dev and multi dev.

parameters. Still, AmbigPrompt achieves superior performance compared to the high-capacity models, e.g., JPR, that use 3B parameters. The state-of-the-art model RECTIFY uses 6B parameters (3B for the answering model and 3B for the verifier), which is 27× as much as ours, significantly increasing the training and inference overhead. Similar results are witnessed in terms of latency. In particular, RECTIFY is 29× slower than our model due to the heavy design of the answering model and verifier. Refuel's iterative passage retrieval and clarifying question generation procedure results in a 32.6× latency compared with our approach. Finally, the comparison of peak memory usage also confirms our approach's lightweight nature. The lightweight design allows our approach to be adapted to academically accessible devices and reduces the carbon footprint for model training and deployment.

Third, we find that AmbigPrompt achieves a better resource-performance balance. In Figure 4 (a), we display the existing methods under the speed-performance coordinate system. Note that we place RECTIFY with different sizes (i.e., latency) on the diagram according to Shao and Huang (2022). AmbigPrompt improves the optimal latency-performance curve (the dashed lines), especially on the multi-answer test set, demonstrating the effectiveness of our approach in answering ambiguous questions.

Methods	AmbigQA		WebQSP	
	Full	Multi	Full	Multi
AmbigPrompt	48.7	38.8	53.3	46.7
- w/o task-adaptive pre-training	42.8	32.7	42.5	38.7
- w/o prompting model	46.0	34.3	49.7	44.6
- w/o interleaving prompting	47.8	36.9	50.9	45.4

Table 3: Ablation study. The base model is compared with several ablative variants on two datasets.

5.2 Low-resource setting

Figure 4 (b) shows the results under different training data sizes to investigate the effectiveness of the proposed method in the low-resource setting. The proposed method achieves favorable results for different data sizes. Remarkably, AmbigPrompt achieves promising performance with little data, surpassing the fully supervised high-capacity model JPR on a multi-answer test set. This result suggests that the proposed prompting mechanism can better elicit the capabilities of the pre-trained model and effectively adapt the model trained on single-answer QA data to multi-answer scenarios.

5.3 Ablation study

To understand the contribution of each component of AmbigPrompt, we conduct an ablation study. The results are listed in Table 3. The compared variants and the findings are:

W/o task-adaptive pre-training. The models are trained only on multi-QA data with $\mathcal{L}_{PT}$. A notable performance decline can be seen. This observation suggests that task-adaptive pre-training is an important contributor to the model's performance since the size of multi-answer QA data is small.

W/o prompting model. We remove the prompting model in this variant and instantiate the learnable prompt vector to each step t separately, like Liu et al. (2021a). The performance drops by about 3% and 4% on the two datasets, respectively. The results verify the effectiveness of the proposed answer-conditional prompting mechanism.

W/o interleaving prompting. We remove the interaction mechanism between the prompting model and answering model, i.e., the FiD encoder encodes the e and X independently without cross-attention. The results drop by about 2% and 2% on two datasets, respectively, which reveals that enabling the answering model to generate new answers conditioned on the introspective prompts effectively improves the model's performance.

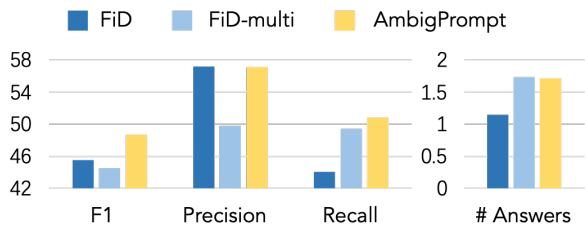

Figure 5: The F1, Precision, Recall, and the average number of answers (#Answers) of AmbigPrompt and FiD model variants on AmbigQA dev.

5.4 Analytical experiments

Conceptually, our proposed framework AmbigPrompt equips the FiD model with the ability to progressively compose the answers using retrospective prompts, i.e., iterative prompt learning. To further analyze the capability of such an iterative prompt learning approach in managing the relevance-diversity trade-off, we present the F1, precision, recall, and average answer numbers of AmbigPrompt and FiD model variants in Figure 5. In particular, FiD-multi denotes a variant of FiD in which we reduce the generation probability of the end-of-sequence token </s> to ensure that the number of generated answers is approximately the same as AmbigPrompt. We see that FiD-multi obtains comparable recall but gets significantly lower precision. In contrast, AmbigPrompt generates more answers than FiD without sacrificing precision, indicating that the designed iterative prompting mechanism induces the model with a superior ability to manage the trade-off between relevancy and diversity for ambiguous question answering.

6 Related work

6.1 Ambiguous question answering

In open-domain QA, given a question about any topic, the model finds the answer from a large knowledge corpus (Chen et al., 2017). Typically, a retrieval model and an answering model are employed. The two modules can be trained separately (Karpukhin et al., 2021; Izacard and Grave, 2021b; Qu et al., 2021) or jointly (Lee et al., 2022; Lewis et al., 2020; Izacard and Grave, 2021a). Ambiguity is inherent to open-domain QA; especially when exploring new topics, it can be difficult to ask questions that have a single, unambiguous answer (Min et al., 2020; Rubin et al., 2022). Min et al. (2020) identify the challenge of *multi-answer QA* and collect the dataset AmbigQA. Based on that, Min et al. (2021) propose an autoregressive passage reranking model JPR, which reranks the top-retrieved passages and improves their diversity. Gao et al. (2021b) propose a round-trip prediction approach, where clarification questions are generated and fed back into the model to find more answers. Shao and Huang (2022) propose a recall-and-verify framework, where surplus answers are generated first, and a verifier model then determines each candidate answer. Compared with existing methods, we propose a lightweight yet effective approach to answering ambiguous questions by iterative prompting.

6.2 Prompt-based learning

Prompt-based learning has received much attention recently (Liu et al., 2021a). Existing studies on prompt-based learning mainly focus on discrete and continuous prompts. The former designs text-based prompts (Jiang et al., 2020; Gao et al., 2021a; Schick and Schütze, 2021), while the latter prepend a learnable prompt vector to word embeddings (Lester et al., 2021; Liu et al., 2021b) or attention layers (Li and Liang, 2021; Liu et al., 2022). Prompt-based learning has demonstrated advantages in low-parameter tuning (He et al., 2022) and few-shot/zero-shot performance (Brown et al., 2020; Wei et al., 2022a). We propose an iterative prompting method for multi-answer QA based on answer-conditional continuous prompts.

6.3 Iterative generation

Iterative generation (a.k.a. progressive generation) aims to decompose a challenging generation task into multiple steps and progressively produce the target sequence. Iterative generation has been applied to the tasks of machine translation (Lee et al., 2018), controllable text generation (Casas et al., 2020; Zhang et al., 2020), storytelling (Hua and Wang, 2020; Tan et al., 2021), data-to-text (Kasner and Dusek, 2020), etc. Recently, Wang et al. (2022) introduced an iterative prompting framework to progressively elicit knowledge from language models for commonsense reasoning and multi-hop question answering tasks (Qi et al., 2019; Xiong et al., 2021). Compared to existing work, we propose an answer-conditional prompting model and an effective task-specific pre-training scheme for multi-answer QA.

7 Conclusions

In this paper, we have proposed AmbigPrompt for multi-answer QA. AmbigPrompt is a simple yet effective model that answers ambiguous questions by iterative prompting. We have proposed an answer-conditional prompting model for prompt generation, and a task-adaptive post-pretraining scheme for model training. Extensive experiments suggest that AmbigPrompt achieves comparable performance as high-capacity models and achieves the best results in a low-resource setting.

Limitations

The limitations of this paper include the absence of experiments on large language models. Previous studies have shown that using high-capacity pre-trained language models can significantly improve the accuracy of answers but also entails an increase in computational overhead. Due to (academic) limitations of computational resources, this paper employs a low-capacity T5 model for experiments. Our experiments have suggested that the proposed iterative prompting method that works with the low-capacity model can achieve comparable results with baseline methods equipping with large models.

In future work, we would like to scale up the proposed model to improve the model's performance. Recent research on large language models (LLMs) has shown that they can learn from few examples and reason well. We believe that it is worth exploring ways to enhance the prompting of LLMs to improve their completeness when responding to ambiguous questions and reduce model hallucination in generation (OpenAI, 2023; Zhao et al., 2023; Sun et al., 2023b,a). Another direction worth exploring in the future is the application in low-resource scenarios, such as low-resource languages. Low-resources in our study are characterized by limited multi-answer-QA annotations, which aims to examine how data size impacts model performance. Other low-resource languages may behave differently with less training data and large models (Xue et al., 2020; Sun et al., 2021). Besides, we would like to explore more effective prompting methods, such as chain-of-thought prompting (Wei et al., 2022b).

Ethics Statement

The paper has proposed a question-answering model, which is intended to answer factoid open-domain questions. The model-predicted answers still have a considerable amount of misinformation. Besides, the proposed models rely on pre-trained question-answering models, which are trained on large-scale web data that is known to contain biased or discriminatory content.

Acknowledgements

This work was supported by the National Key R&D Program of China with grant No. 2020YFB1406704, the Natural Science Foundation of China (62272274, 61972234, 62072279, 62102234, 62202271), the Natural Science Foundation of Shandong Province (ZR2022QF004), the Key Scientific and Technological Innovation Program of Shandong Province (2019JZZY010129), the Fundamental Research Funds of Shandong University, the Hybrid Intelligence Center, a 10-year program funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organization for Scientific Research, https://hybrid-intelligence-centre.nl.

All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

Jonathan Berant, Andrew K. Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *NeurIPS*.

Noe Casas, José A. R. Fonollosa, and Marta Ruiz Costa-jussà. 2020. Syntax-driven iterative expansion language models for controllable text generation. In *SPNLP*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *ACL*.

Luyu Gao and Jamie Callan. 2021. Unsupervised corpus aware language model pre-training for dense passage retrieval. In *ACL*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021a. Making pre-trained language models better few-shot learners. In *ACL*.

Yifan Gao, Henghui Zhu, Patrick Ng, Cícero Nogueira dos Santos, Zhiguo Wang, Feng Nan, Dejiao Zhang, Ramesh Nallapati, Andrew O. Arnold, and Bing Xiang. 2021b. Answering ambiguous questions through generative evidence fusion and round-trip prediction. In *ACL*.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *ICLR*.

Xinyu Hua and Lu Wang. 2020. Pair: Planning and iterative refinement in pre-trained transformers for long text generation. In *EMNLP*.

Gautier Izacard and Edouard Grave. 2021a. Distilling knowledge from reader to retriever for question answering. In *ICLR*.

Gautier Izacard and Edouard Grave. 2021b. Leveraging passage retrieval with generative models for open domain question answering. In *EACL*.

Zhengbao Jiang, Frank F. Xu, J. Araki, and Graham Neubig. 2020. How can we know what language models know? *TACL*, 8:423–438.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Yu Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2021. Dense passage retrieval for open-domain question answering. In *NAACL*.

Zdeněk Kasner and Ondrej Dusek. 2020. Data-to-text generation with iterative text editing. In *INLG*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc V. Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *TACL*, 7:453–466.

Haejun Lee, Akhil Kedia, Jongwon Lee, Ashwin Paranjape, Christopher D. Manning, and Kyoung-Gu Woo. 2022. You only need one model for open-domain question answering. In *EMNLP*.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *EMNLP*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *EMNLP*.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *NeurIPS*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. In *ACL*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. GPT understands, too. *arXiv preprint arXiv:2103.10385*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. In *ICLR*.

Sewon Min, Kenton Lee, Ming-Wei Chang, Kristina Toutanova, and Hannaneh Hajishirzi. 2021. Joint passage ranking for diverse multi-answer retrieval. In *EMNLP*.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. Ambigqa: Answering ambiguous open-domain questions. In *EMNLP*.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Peng Qi, Xiaowen Lin, Leo Mehr, Zijian Wang, and Christopher D. Manning. 2019. Answering complex open-domain questions through iterative query generation. In *EMNLP*.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. In *NAACL*.

Colin Raffel, Noam M. 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. *JMLR*.

Samuel J. Rubin, Ori Yoran, Tomer Wolfson, Jonathan Herzig, and Jonathan Berant. 2022. QAMPARI: An open-domain question answering benchmark for questions with many answers from multiple paragraphs. *arXiv preprint arXiv:2205.12665*.

Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *NAACL*.

Zhihong Shao and Minlie Huang. 2022. Answering open-domain multi-answer questions via a recall-then-verify framework. In *ACL*.

Weiwei Sun, Chuan Meng, Qi Meng, Zhaochun Ren, Pengjie Ren, Zhumin Chen, and Maarten de Rijke. 2021. Conversations powered by cross-lingual knowledge. In *SIGIR*.

Weiwei Sun, Zhengliang Shi, Shen Gao, Pengjie Ren, Maarten de Rijke, and Zhaochun Ren. 2023a. Contrastive learning reduces hallucination in conversations. In *AAAI*.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Pengjie Ren, Dawei Yin, and Zhaochun Ren. 2023b. Is chatgpt good at search? investigating large language models as re-ranking agent. *arXiv preprint arXiv:2304.09542*.

Bowen Tan, Zichao Yang, Maruan Al-Shedivat, Eric P. Xing, and Zhiting Hu. 2021. Progressive generation of long text with pretrained language models. In *NAACL*.

Wen tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *ACL*.

Boshi Wang, Xiang Deng, and Huan Sun. 2022. Shepherd pre-trained language models to develop a train of thought: An iterative prompting approach. In *EMNLP*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022a. Finetuned language models are zero-shot learners. In *ICLR*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. In *NeurIPS*.

Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Wen tau Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oğuz. 2021. Answering complex open-domain questions with multi-hop dense retrieval. In *ICLR*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. In *NAACL*.

Yizhe Zhang, Guoyin Wang, Chunyuan Li, Zhe Gan, Chris Brockett, and Bill Dolan. 2020. Pointer: Constrained progressive text generation via insertion-based generative pre-training. In *EMNLP*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Appendices

A Results on NQ

Table 4 lists the exact match (EM) score of the baselines and AmbigPrompt on single-answer QA benchmark, NQ-Open test. We see that the high-capacity models (e.g., JPR), which benefit from large language models like T5-3B, achieve better EM score. However, in the multi-answer QA task, the models need to focus not only on the precision of answers, but also on the diversity of answers (i.e., recall rate). In AmbigQA, we can see that the proposed model outperforms JPR, indicating its superior ability to recall multiple feasible answers.

Method	#Params	EM
JPR (Min et al., 2021)	3B	54.5
RECTIFY (Shao and Huang, 2022)	6B	54.8
DPR (Karpukhin et al., 2021)	345M	41.5
SpnSeqGen (Min et al., 2020)	400M	45.0
FiD-Base (Izacard and Grave, 2021b)	220M	48.2
FiD-Large (Izacard and Grave, 2021b)	700M	51.4
?	220M	49.6
Refuel (Gao et al., 2021b)	400M	48.9
AmbigPrompt	220M	49.2

Table 4: Model size and EM score on NQ test.

B Zero-shot evaluation on AmbigQA

We also test the proposed model and baselines on AmbigQA in zero-shot setting following Min et al. (2020). In zero-shot evaluation, the models are trained using partial supervision only (i.e., single-answer NQ-Open (Kwiatkowski et al., 2019)), and are evaluated on multi-answer data AmbigQA. This setting provides a practical application where only single-answer datasets are available. Note that the zero-shot evaluation on AmbigQA allows the model to tune some hyper-parameters (e.g., threshold of generation probability (Min et al., 2020)) using development data, which may make the setting not zero-shot in the strictest sense.

The compared models are (1) DPR and SpanSeqGen, in which the models trained on NQ-Open are adopted to predict multiple answers via a thresholding strategy (Min et al., 2020). (2) FiD with various decoding methods, in which FiD trained on NQ-Open produces multiple answers through (a) Nucleus sampling with $\{p{=}0.8, t{=}0.8\}$; (b) Top-k sampling with $\{k{=}40, t{=}0.8\}$; and (c) Diverse beam search with $\{b{=}3, t{=}0.8, \textit{diversity\_penalty}{=}0.5\}$. We also evaluate FiD with greedy decoding that generates one answer for each question as the default setting of FiD. (3) AmbigPrompt, in which the FiD answering model prompted by our proposed answer-conditional prompting model is trained on NQ-Open with our task-adaptive post-pretraining method and produces multiple answers through iterative prompting.

The results are listed in Table 5. FiD series outperform DPR and SpanSeqGen as they utilize more passages that potentially cover more feasible answers. FiD with nucleus sampling obtains the best results among different decoding methods. AmbigPrompt achieves the best zero-shot performance on AmbigQA and also outperforms high-capacity supervised baselines JPR on the multi-answer subset.

Methods	Full	Multi
DPR	35.2	26.5
SpanSeqGen	36.4	24.8
FiD	43.7	33.5
- nucleus sampling	45.7	36.7
- top-k sampling	42.6	34.7
- diverse beam search	45.2	36.1
AmbigPrompt	**46.5**	**37.9**

Table 5: Zero-shot evaluation results on AmbigQA.

C Retrieval results

We train the dense retrieval model on NQ-Open using in-batch negatives with batch size 64. The retrieval model is initialized from CoCondenser (Gao and Callan, 2021). Our retrieval corpus is the English Wikipedia from 12/20/2018. Table 6 lists the retrieval results on NQ-Open and AmbigQA. In NQ-Open, we use Recall@k (R@k for short) as the metric, which considers retrieval to be successful if at least one answer is included in the top-k ranked passages. In AmbigQA, we use MRecall@k (MR@k for short) as the metric, which considers retrieval to be successful if all answers or at least k answers in the answer set $\mathcal{A}$ are covered by the top-k ranked passages. From the results, we see that our retrieval model achieves comparable results against baseline retrieval models, but underperforms reranking models such as KPR and MonoT5.

D Case study

We present some examples in Table 7 and Table 8.

NQ-Open	R@1	R@5	R@10	R@100
DPR	43.1	68.5	76.4	87.9
RECTIFY	-	73.8	-	89.3
Ours	50.9	72.2	78.2	88.2
AmbigQA	MR@1	MR@5	MR@10	-
DPR	-	55.2	59.3	-
RECTIFY	-	53.2	60.0	-
MonoT5†	-	63.4	65.8	-
JPR†	-	64.8	67.1	-
Ours	61.7	56.4	62.6	-

Table 6: Retrieval results on NQ-Open test and AmbigQA dev. † denotes reranking model.

Question	Who holds the record for most passing yards in a season?
Passages	Associated Press NFL Offensive Player of the Year Award ǀ Marino's 5,084 yards stood as the record for 27 years before being broken by Drew Brees in 2011, who won that season's award. In turn, 2013 winner Peyton Manning set league single-season records for passing yards (5,477) and passing touchdowns (55). [...]
FiD	drew brees, peyton manning
Ours	drew brees, dan marino, peyton manning
Human	Peyton Manning, Drew Brees, Dan Marino

Table 7: An example on AmbigQA dev shows that the proposed method AmbigPrompt finds all valid answers.

Question	Who was the bond girl in you only live twice?
Passages	Severine ǀ She had also categorized Aki and Kissy Suzuki, both from "You Only Live Twice" (1967), as falling into this trope. She supported this assessment by pointing to the characterś lack of agency and impact on "Skyfall"ś main narrative, and summed up Sévérine as "one of the most disempowered, pitiful, and tragic women in the Bond film franchise". [...] You Only Live Twice (film) ǀ Sean Connery's then-wife Diane Cilento performed the swimming scenes for at least five Japanese actresses, including Mie Hama. Martial arts expert Donn F. Draeger provided martial arts training, [...]
FiD	akiko wakabayashi
Ours	aki, kissy suzuki, yasuko nagazumi, akiko wakabayashi
Human	Aki, Akiko Wakabayashi, Kissy Suzuki, Mie Hama

Table 8: An example on AmbigQA dev shows that AmbigPrompt finds more valid answers than FiD.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
On page 9, Section "Limitations".

☑ A2. Did you discuss any potential risks of your work?
On page 9, Section "Ethics Statement".

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☒ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ 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)?
No response.

☐ 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?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ 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? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

Section 5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4 and 5

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4

☑ 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?
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.)?
Section 4

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

Left blank.

☐ 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.?
No response.

☐ 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)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.