Answering Open-Domain Multi-Answer Questions via a Recall-then-Verify Framework

Anonymous ACL submission

Abstract

Open-domain questions are likely to be open-ended and ambiguous, leading to multiple valid answers. Existing approaches typically adopt the rerank-then-read framework, where a reader reads top-ranking evidence to predict answers. According to our empirical analysis, this framework faces three problems: first, to leverage a large reader under a memory constraint, the reranker should select only a few relevant passages to cover diverse answers, while balancing relevance and diversity is non-trivial; second, the small reading budget prevents the reader from accessing valuable retrieved evidence filtered out by the reranker; third, when using a generative reader to predict answers all at once based on all selected evidence, whether a valid answer will be predicted also pathologically depends on the evidence of some other valid answer(s). To address these issues, we propose to answer open-domain multi-answer questions with a recall-then-verify framework, which separates the reasoning process of each answer so that we can make better use of retrieved evidence while also leveraging large models under the same memory constraint. Our framework achieves state-of-the-art results on two multi-answer datasets, and predicts significantly more gold answers than a rerank-then-read system that uses an oracle reranker.

1 Introduction

Open-domain question answering (Voorhees, 1999; Chen et al., 2017) is a long-standing task where a question answering system goes through a large-scale corpus to answer information-seeking questions. Previous work typically assumes that there is only one well-defined answer for each question, or only requires systems to predict one correct answer, which largely simplifies the task. However, humans may lack sufficient knowledge or patience to frame very specific information-seeking questions, leading to open-ended and ambiguous questions with multiple valid answers. According to Min et al. (2020b), over 50% of a sampled set of Google search queries (Kwiatkowski et al., 2019) are ambiguous. Figure 1 shows an example with at least three interpretations. As can be seen from this example, the number of valid answers depends on both questions and relevant evidence, which challenges the ability of comprehensive exploitation of evidence from a large-scale corpus.

Existing approaches mostly adopt the rerank-then-read framework. A retriever retrieves hundreds or thousands of relevant passages which are later reranked by a reranker; a generative reader then predicts all answers in sequence conditioned on top-ranking passages. With a fixed memory constraint, there is a trade-off between the size of the reader and the number of passages the reader can process at a time. According to Min et al. (2021), provided that the reranker is capable of selecting a small set of highly-relevant passages with high coverage of diverse answers, adopting a

Original Question: When did [You Don’t Know Jack] come out?

Interpretation #1: When did the first video game called [You Don’t Know Jack] come out?

Evidence #1: You Don’t Know Jack is a video game released in 1995, and the first release in ...

Answer #1: 1995

Interpretation #2: When did the Facebook game [You Don’t Know Jack] come out on Facebook?

Evidence #2: In 2012, Jackbox Games developed and published a social version of the game on Facebook ...

Answer #2: 2012

Interpretation #3: When did the film [You Don’t Know Jack] come out?

Evidence #3: “You Don’t Know Jack” premiered April 24, 2010 on HBO.

Answer #3: April 24, 2010

Table 1: An example of open-domain multi-answer questions. We display only a subset of valid answers. In fact, [You Don’t Know Jack] can also be a song.
larger reader can outperform a smaller reader using more passages. However, as shown by Section 4.4, this framework is faced with three problems: first, due to the small reading budget, the reranker has to balance relevance and diversity, which is non-trivial as it is unknown beforehand that which answers should be distributed with more passages to convince the reader and which answers can be safely distributed with less to save the budget for the other answers; second, the reader has no access to more retrieved evidence that may be valuable but is filtered out by the reranker, while combining information from more passages was found to be beneficial to open-domain QA (Izacard and Grave, 2021b); third, as the reader predicts answers in sequence all at once, the reader learns pathological dependencies among answers, i.e., whether a valid answer will be predicted also depends on passages that cover some other valid answer(s), while ideally, prediction of a particular answer should depend on the soundness of associated evidence itself.

To address these issues, we propose to answer open-domain multi-answer questions with a recall-then-verify framework. Specifically, we first use an answer recaller to predict possible answers from each retrieved passage individually; this can be done with high recall, even when using a weak model for the recaller, but at the cost of low precision due to insufficient evidence to support or refute a candidate. We then aggregate retrieved evidence relevant to each candidate, and verify each candidate with a large answer verifier. By separating the reasoning process of each answer, our framework avoids the problem of multiple answers sharing a limited reading budget, and makes better use of retrieved evidence while also leveraging strong large models under the same memory constraint.

Our contributions are summarized as follows:

- We empirically analyze the problems faced by the rerank-then-read framework when dealing with open-domain multi-answer QA.
- To address these issues, we propose to answer open-domain multi-answer questions with a recall-then-verify framework, which makes better use of retrieved evidence while also leveraging the power of large models under the same memory constraint.
- Our framework establishes a new state-of-the-art record on two multi-answer QA datasets with significantly more valid predictions.

2 Related Work

Open-domain QA requires question answering systems to answer factoid questions by searching for evidence from a large-scale corpus such as Wikipedia (Voorhees, 1999; Chen et al., 2017). The presence of many benchmarks has greatly promoted the development of this community, such as questions from real users like NQ (Kwiatkowski et al., 2019) and WebQuestions (Berant et al., 2013), and trivia questions like Quasar-T (Dhingra et al., 2017) and TriviaQA (Joshi et al., 2017). All these benchmarks either assume that each question has only one answer with several alternative surface forms, or only require a system to predict one valid answer. A typical question answering system is a pipeline as follows: an efficient retriever retrieves relevant passages using sparse (Mao et al., 2021; Zhao et al., 2021) or dense (Karpukhin et al., 2020; Xiong et al., 2021; Izacard and Grave, 2021a; Khattab et al., 2021) representations; an optional passage reranker (Asadi and Lin, 2013; Nogueira and Cho, 2019; Nogueira et al., 2020) further narrows down the evidence; an extractive or generative reader (Izacard and Grave, 2021b; Cheng et al., 2021) predicts an answer conditioned on retrieved or top-ranking passages. Nearly all previous work focused on locating passages covering at least one answer, or tried to predict one answer precisely.

However, both Kwiatkowski et al. (2019) and Min et al. (2020b) reported that there is genuine ambiguity in open-domain questions, resulting in multiple valid answers. To study the challenge of finding all valid answers for open-domain questions, Min et al. (2020b) proposed a new benchmark called AMBIGQA where questions are annotated with as many answers as possible. In this new task, the passage reranker becomes more vital in the rerank-then-read framework, particularly when only a few passages are allowed to feed a large reader due to memory constraints. This is because the reranker has to ensure that top-ranking passages are highly relevant and also cover diverse answers. Despite state-of-the-art performance on AMBIGQA (Min et al., 2021), according to our empirical analysis, applying the rerank-then-read framework to open-domain multi-answer QA faces the following problems: balancing relevance and diversity is non-trivial for the reranker due to unknown effect on the performance of the subsequent reader; when using a large reader under a fixed memory constraint, the small reading budget prevents it from making use
of more retrieved evidence that is valuable but filtered out; when using a generative reader to predict all answers in sequence based on all selected evidence, it learns pathological dependencies among answers. To address these issues, we propose to tackle this task with a recall-then-verify framework, which separates the reasoning process of each answer with a higher level of evidence usage while also leveraging large models under the same memory constraint.

Some previous work argued that a reader can be confused by similar but spurious passages, resulting in wrong predictions. Therefore, they proposed answer rerankers (Wang et al., 2018a,b; Hu et al., 2019; Iyer et al., 2021) to rerank top predictions from readers. Our framework is related to answer reranking but with two main differences. First, a reader typically aggregates available evidence and already does a decent job of answer prediction even without answer reranking; an answer reranker is introduced to filter out hard false positive predictions from the reader. By contrast, our answer verifier aims at finding possible answers with high recall, most of which are invalid. Evidence focused on each answer is then aggregated and reasoned about by our answer verifier. It is also possible to introduce another model analogous to an answer reranker to filter out false positive predictions from our answer verifier. Second, answer reranking typically compares answer candidates to determine the most valid one, while our answer verifier selects multiple valid answers mainly based on the soundness of their respective evidence but without comparisons among answer candidates.

3 Task Formulation

Open-domain multi-answer QA can be formally defined as follows: given an open-ended question \( q \), a question answering system is required to make use of evidence from a large-scale text corpus \( C \) and predict a set of valid answers \( \{a_1, a_2, \ldots, a_n\} \). Questions and their corresponding answer sets are provided for training.

Evaluation To evaluate passage retrieval and reranking, we adopt the metric MRECALL@\( k \) from (Min et al., 2021), which measures whether the top-\( k \) passages cover at least \( k \) distinct answers (or \( n \) answers if the total number of answers \( n \) is less than \( k \)). To evaluate question answering performance, we follow (Min et al., 2020b) to use F1 score between gold answers and predicted ones.

4 Rerank-then-Read Framework

In this section, we will briefly introduce the representative and state-of-the-art rerank-then-read pipeline from (Min et al., 2021) for open-domain multi-answer questions, and provide empirical analysis of this framework.

4.1 Passage Retrieval

Dense retrieval is widely adopted by open-domain question answering systems (Min et al., 2020a). A dense retriever measures relevance of a passage to a question by computing the dot product of their semantic vectors encoded by a passage encoder and a question encoder, respectively. Given a question, a set of the most relevant passages, denoted as \( B (|B| < |C|) \), is retrieved for subsequent processing.

4.2 Passage Reranker

To improve the quality of evidence, previous work (Nogueira et al., 2020; Gao et al., 2021) finds it effective to utilize a passage reranker, which is more expressive than a passage retriever, to rerank retrieved passages, and select the \( k \) best ones to feed a reader for answer generation (\( k < |B| \)). With a fixed memory constraint, there is a trade-off between the number of selected passages and the size of the reader. As shown by (Min et al., 2021), with good reranking, using a larger reader is more beneficial. To balance relevance and diversity of evidence, Min et al. (2021) proposed a passage reranker called JPR for joint modeling of selected passages. Specifically, they utilized T5-base (Raffel et al., 2020) to encode retrieved passages following (Izacard and Grave, 2021b) and decode the indices of selected passages autoregressively using a tree-decoding algorithm. JPR is designed to seek for passages that cover new answers, while also having the flexibility to select more passages covering the same answer, especially when there are less than \( k \) answers for the question.

4.3 Reader

A reader takes as input the top-ranking passages, and predicts answers. Min et al. (2021) adopted a generative encoder-decoder reader initialized with T5-3b, and used the fusion-in-decoder method from (Izacard and Grave, 2021b) which efficiently aggregates evidence from multiple passages. Specifically, each passage is concatenated with the question and is encoded independently by the encoder; the decoder then attends to the representations of
all passages and generates all answers in sequence, separated by a [SEP] token.

### 4.4 Empirical Analysis

To analyze performance of the rerank-then-read framework for open-domain multi-answer questions, we built a system that resembles the state-of-the-art pipeline from (Min et al., 2021) but with two differences\(^2\). First, we used the retriever from (Izacard and Grave, 2021a). Second, instead of using JPR, we used an oracle passage reranker (OPR): a passage \(p\) is ranked higher than another passage \(p'\) if and only if 1) \(p\) covers some answer while \(p'\) covers none 2) or both \(p\) and \(p'\) cover or fail to cover some answer but \(p\) has a higher retrieval score. Following (Min et al., 2021), we retrieved \(|\mathcal{B}|=100\) Wikipedia passages, \(k=10\) of which were selected by the reranker. Table 2 shows model performance on a representative multi-answer dataset called AMBIGQA (Min et al., 2020b). Compared with JPR, OPR is better in terms of reranking, with similar question answering results\(^3\).

<table>
<thead>
<tr>
<th>Model</th>
<th>Reranking</th>
<th>QA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MREC@5</td>
<td>MREC@10</td>
</tr>
<tr>
<td>JPR</td>
<td>64.8/45.2</td>
<td>67.1/48.2</td>
</tr>
<tr>
<td>OPR</td>
<td>67.7/46.5</td>
<td>70.3/51.2</td>
</tr>
</tbody>
</table>

Table 2: Reranking results and Question Answering results on the dev set of AMBIGQA using JPR and OPR. The two numbers in each cell are results on all questions and questions with multiple answers, respectively.

Though 3,670 diverse gold answers are covered by OPR on the dev set, the reader predicts only 1,554 of them. Our empirical analysis and findings are detailed as follows.

(1) To leverage a large reader under a fixed memory constraint, a reranker should select only a few highly-relevant passages to cover diverse answers, while balancing relevance and diversity is non-trivial. As shown by Figure 1a (bottom), the number of selected supporting passages\(^4\) of predicted gold answers has a widespread distribution. There may be cases where redundant false positive evidence is selected and can be safely replaced with passages that cover other gold answers. However, it is non-trivial for the reranker to know beforehand whether a passage is redundant, and how many or which supporting passages of an answer are strong enough to convince the reader.

(2) Multiple answers sharing a small reading budget prevents a reader from using more evidence that may be valuable but is filtered out by the reranker. Due to the shared reading budget, it is inevitable that some answers are distributed with less supporting passages. As shown by Figure 1a, a gold answer covered by OPR but missed by the reader generally has significantly less supporting passages fed to the reader (3.13 on average) than a predicted gold answer (5.08 on average), but not because of lacking available evidence. There is more evidence in retrieved passages for missed answers but filtered out by the reranker. As shown by Figure 1b, OPR has a much lower level of evidence usage for missed answers.

(3) As the reader predicts answers all at once conditioned on all selected passages, whether a valid answer will be predicted also pathologically depends on evidence of some other valid answer(s), which partly accounted for the large number of gold answers missed by the reader. For verification, we attacked OPR’s reader on the dev set of AMBIGQA as follows: a question is a target if and only if 1) it has a gold answer covered by OPR but missed by the reader 2) and it has a predicted gold answer whose supporting passages cover no

---

\(^2\)Code and models from (Min et al., 2021) were not publicly available in the period of this work.

\(^3\)With the oracle knowledge of whether a passage contains a gold answer during reranking, OPR is probably still far from being a perfect reranker. Notably, we are not striving for a better rerank-then-read pipeline for multi-answer questions, but use OPR as a representative case to analyze the problems a rerank-then-read pipeline may face.

\(^4\)We abuse the use of supporting passages of an answer to refer to passages that cover the answer.
had a similar observation on another rerank-then-read pipeline, i.e., it is hard to argue that the predicted answers are more likely than the missed ones.

## 5 Recall-then-Verify Framework

### 5.1 Overview

To avoid the issues faced by the rerank-then-read framework, we propose a recall-then-verify framework, which separates the reasoning process of each answer so that answers (1) can be individually distributed with maximum supporting passages allowed on the same hardware (2) and are predicted mainly based on their own evidence. Figure 2 shows our framework. Specifically, we first guess possible answers based on retrieved passages using an answer recaller, an evidence aggregator then aggregates evidence for each candidate, and finally, an answer verifier verifies each candidate and outputs valid ones.

### 5.2 Answer Recaller

Our answer recaller, based on T5, is trained to predict all gold answer(s) in sequence (separated by a [SEP] token) from each retrieved positive passage $p \in B$ that cover some gold answer(s). We also train the recaller to predict the “irrelevant” token given a negative passage so that the recaller can filter out negative candidates; the number of negatives per positive used for training is denoted as $\alpha_{neg}$. The set of answer candidates recalled during inference is denoted as $\mathcal{A} = \{\hat{a}_1, \hat{a}_2, ..., \hat{a}_m\}$. Though a passage may not contain strong enough evidence to support an answer, by exploiting semantic clues in the question and the passage (e.g., the answer type), it is sufficient for even a weak model to achieve high recall. However, this is at the cost of low precision, which necessitates answer
verification based on more supporting passages.

5.3 Evidence Aggregator

We aggregate evidence for each answer candidate from retrieved passages, which can be formulated as a reranking task, i.e., to rerank retrieved passages according to their relevance to a question-candidate pair, and select top-ranking ones for answer verification. Our evidence aggregator resembles OPR: for a specific candidate \(\hat{a}_i\), we encode the question-candidate pair with the retriever’s question encoder; a passage \(p\) is ranked higher than another passage \(p'\) if and only if 1) \(p\) covers \(\hat{a}_i\) while \(p'\) does not 2) or both \(p\) and \(p'\) cover or fail to cover \(\hat{a}_i\) but the semantic vector of \(p\) is closer to that of the question-candidate pair. We denote the top-\(k\) relevant passages of \(\hat{a}_i\) as \(E_i\).

5.4 Answer Verifier

Given a candidate \(\hat{a}_i\) and its evidence \(E_i\), our answer verifier, based on T5-3b, predicts whether \(\hat{a}_i\) is valid, using the fusion-in-decoder method from (Izacad and Grave, 2021b). Each passage from \(E_i\) is concatenated with the question and the candidate, and is encoded independently; the decoder then attends to the representations of all passages and is trained to produce the tokens “right” or “wrong” depending on whether the encoded candidate is valid or not. During inference, we compute the validity score of a candidate by taking the normalized probability assigned to the token “right”:

\[
P(\hat{a}_i \text{ is valid}) = \frac{\exp(\logit(\text{"right"}|q, \hat{a}_i, E_i))}{\sum_{i \in \{\text{"right"}; \text{"wrong"}\}} \exp(\logit(\text{t}|q, \hat{a}_i, E_i))}\]

Candiates with their validity scores higher than a threshold \(\tau\) will be produced as final predictions.

6 Experiments

6.1 Datasets

We conducted experiments on two multi-answer QA datasets, whose statistics are shown in Table 3. WEBQSP (Yih et al., 2016) is a semantic parsing dataset for knowledge base question answering, where answers are a set of entities in Freebase. Following (Min et al., 2021), we repurposed this dataset for textual QA based on Wikipedia.

AMBIGQA (Min et al., 2020b) originates from NQ (Kwiatkowski et al., 2019), where questions are annotated with equally valid answers from Wikipedia.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Question</th>
<th># Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>Dev</td>
<td>Test</td>
</tr>
<tr>
<td>WEBQSP</td>
<td>2,752</td>
<td>245</td>
</tr>
<tr>
<td>AMBIGQA</td>
<td>10,036</td>
<td>2,002</td>
</tr>
</tbody>
</table>

Table 3: Statistics of multi-answer QA datasets. Statistics of answers are computed on the dev sets.

6.2 Baselines

We compare our recall-then-verify system with two state-of-the-art rerank-then-read systems. REFUEL (Gao et al., 2021) selects 100 top-ranking passages from 1,000 retrieved passages, and predicts answers with a reader based on BARTlarge (Lewis et al., 2020). It also has a round-trip prediction mechanism, i.e., to generate disambiguated questions based on predicted answers, which are re-fed to the reader to recall more answers.

JPR (Min et al., 2021) is a passage reranker which jointly models selected passages. With improved reranking performance, Min et al. (2021) selected only 10 passages from 100 retrieved passages, and used a reader based on T5-3b which is much larger and more powerful than REFUEL’s reader, while requiring no more memory resources than REFUEL.

6.3 Implementation Details

Our retrieval corpus is the English Wikipedia from 12/20/2018. We finetuned the dense retriever from (Izacad and Grave, 2021a) on each multi-answer dataset. The answer recaller and the answer verifier were initialized with T5-3b; both were pretrained on NQ and then finetuned on each multi-answer dataset. \(\alpha_{neg}\) was 0.1 when finetuning the recaller. We retrieved 100 passages for a question, and verified each candidate with \(k=10\) passages.

The threshold \(\tau\) for verification was tuned on the dev set based on the sum of F1 scores on all questions (F1 (all)) and questions with multiple answers (F1 (Multi)); the best \(\tau\) on WEBQSP/AMBIGQA are 0.8/0.5, respectively. Experiments with different model choices for the recaller and different values of \(\alpha_{neg}\), \(k\), and \(\tau\) are shown in Section 6.5.

Memory Constraint: Min et al. (2021) considered a fixed hardware and trained a reader with the maximum number of passages. We follow this memory constraint, under which a reader/verifier based on T5-3b can encode up to 10 passages each.
of length no longer than 360 tokens at a time.

### 6.4 QA Results

Due to candidate-aware evidence aggregation and a fixed sufficient number of passages distributed to each candidate, our *recall-then-verify* framework can make use of most retrieved supporting passages (see our improvements over OPR in Figure 1b). With a higher level of evidence usage, our *recall-then-verify* system outperforms state-of-the-art *rerank-then-read* baselines on both multi-answer datasets, which is shown by Table 4. Though focused on multi-answer questions, our framework is also applicable to single-answer scenario and achieves state-of-the-art results on NQ. Please refer to the Appendix for more details.

### 6.5 Ablation Study

In this section, we present ablation studies on AMBIGQA. Please refer to the Appendix for results on WebQSP, which lead to similar conclusions.

#### 6.5.1 Answer Recalling

**Model Choices for the Answer Recaller** As shown by Table 5, though T5-base is commonly recognized as a much weaker model than T5-3b, a recaller based on T5-base can achieve a high coverage of gold answers, leading to competitive end-to-end performance on the test set.

**Necessity of Verification** To investigate whether the recaller has the potential to tackle multi-answer questions alone, we tuned the precision of the recaller by varying $\alpha_{neg}$. As shown in Table 5, with increased $\alpha_{neg}$, the recaller learns to recall answers more precisely but still significantly underperforms the overall *recall-then-verify* system. It is likely that the recaller is trained on false positive passages, which may mislead the recaller to be over-conservative in filtering out hard negative passages. By contrast, using more evidence for verification is less likely to miss true positive evidence if there is any for a candidate, thus not prone to mislead the verifier.

**Reducing Answer Candidates** Though only using our recaller for multi-answer QA falls short, the recaller can be trained to shrink down the number of candidates so that the burden on the verifier can be reduced. As shown by Table 5, a small value of $\alpha_{neg}$ helps reduce answer candidates without significantly lowering recall.

#### 6.5.2 Answer Verification

![Figure 4: Performance on the dev set of AMBIGQA, with varying $k$ and $\tau$.](image)

Figure 4 shows the benefit of using more evidence for verification. As $k$ increases from 1 to 10, there is a significant boost in F1 scores.

**Effect of $k$** As shown by Figure 4a, the balance between recall and precision can be controlled by $\tau$: a lower $\tau$ leads to higher recall and may benefit performance on questions with multiple answers. With $k=10$, our system outperforms the previous state-of-the-art system for a wide range of $\tau$. As shown by
Figure 4b, under the best setups \((k=10, \tau=0.5)\), our system predicts 31.7% and 34.1% more gold answers than the system using OPR on all questions and questions with evidence and are more robust to distraction by the other gold answers. Specifically, we attacked the verifier as follows: a question-candidate pair is a target if and only if 1) the candidate \(\hat{a}_i\) is a gold answer and 2) the aggregated evidence \(E_i\) includes at least one supporting passage of some other gold answer(s) that do not cover \(\hat{a}_i\); we removed an arbitrary subset of supporting passages of the other gold answer(s) at a time\(^8\) without removing any supporting passages of \(\hat{a}_i\), and recorded the worst changes of the predicted validity scores of \(\hat{a}_i\). As shown by Figure 5, the changes are small, indicating that missed gold candidates with low scores are not mainly suppressed by some other answer(s), and that predicted gold candidates with high scores are verified mainly based on their associated evidence.

6.6 Error Analysis

Among 3,288 recalled gold answers on the dev set of AMBIGQA, the answer verifier misses 1,242 of them and outputs 1,323 wrong predictions. We manually analyzed 50 random samples, 25 of which are missed gold answers and 25 are wrong predictions. Table 6 reports our analysis.

\(^8\)Removed passages were replaced with the same number of top-ranking passages that cover no gold answers.

For 76% of missed gold answers, our evidence aggregator actually aggregates straightforward true positive evidence. Among these missed answers with straightforward evidence, 58% of them have validity scores higher than 0.2 but lower than the threshold 0.5. We attacked the verifier on missed gold answers with their validity scores below 0.2 as in Section 6.5.2, and found that the maximum change of predicted scores on average is small (+0.04), indicating that the low scores cannot be attributed to the negative distraction by the other gold answer(s). We conjecture that, as it is difficult even for human annotators to find all valid answers to an open-domain question (Min et al., 2020b), the verifier was trained to refute false negative candidates, resulting in unexpected low scores on some straightforward valid answers.

Notably, 80% of our “wrong” predictions turn out to be false negatives: 52% of “wrong” predictions are semantically equivalent to some annotated answer but are superficially different (Si et al., 2021); 28% of “wrong” predictions are unannotated false negatives. Therefore, it is likely that our system is underrated.

For more implementation details, inference efficiency, please refer to the Appendix.

7 Conclusion
In this paper, we empirically analyze the problems of the rerank-then-read framework for open-domain multi-answer questions, and propose the recall-then-verify framework, which separates the reasoning process of each answer so that 1) we can have a higher level of evidence usage 2) and predicted answers are mainly based on associated evidence and are more robust to distraction by evidence of the other gold answer(s), 3) while also leveraging large models under the same memory constraint. On two multi-answer datasets, our framework significantly outperforms rerank-then-read baselines with new state-of-the-art records.
References


A Implementation Details

A.1 Retriever

Our retrieval corpus is the English Wikipedia from 12/20/2018, where articles are split into 100-word passages. Both OPR and our recall-then-verify system share the same passage retriever, which was initialized with the checkpoint released by (Izacard and Grave, 2021a) and was finetuned on each multi-answer dataset following DPR (Karpukhin et al., 2020). Specifically, for each question, we retrieved 100 passages with Izacard and Grave (2021a)’s checkpoint; for each gold answer \( a_i \), we treated top-6 retrieved passages covering \( a_i \) as positives, and top-30 retrieved passages covering no gold answer as hard negatives. During finetuning, batch size was set to 128; each question in a batch was paired with one random positive passage and two random hard negatives.

<table>
<thead>
<tr>
<th>[5] Retriever</th>
<th>WebQSP (Test)</th>
<th>AMBIGQA (Dev)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 DPR(^+)</td>
<td>57.0/38.9</td>
<td>55.2/36.3</td>
</tr>
<tr>
<td>Ours</td>
<td>56.1/37.7</td>
<td>53.2/28.9</td>
</tr>
<tr>
<td>10 DPR(^+)</td>
<td>59.0/38.6</td>
<td>59.3/39.6</td>
</tr>
<tr>
<td>Ours</td>
<td>57.6/35.9</td>
<td>60.0/37.7</td>
</tr>
<tr>
<td>100 Ours</td>
<td>68.0/47.8</td>
<td>73.0/57.6</td>
</tr>
</tbody>
</table>

Table 7: Retrieval results in terms of MREC. DPR\(^+\) is the retriever of JPR (Min et al., 2021). We only report results on the test set of WebQSP and the dev set of AMBIGQA because Min et al. (2021) used a different train/dev split on WebQSP and the test set of AMBIGQA is hidden.

Table 7 shows the performance of our retriever. Our retriever underperforms DPR\(^+\), the retriever of JPR (Min et al., 2021), in terms of MREC@5 and MREC@10. As DPR\(^+\) has not been released, it is unknown whether DPR\(^+\) still covers more gold answers than our retriever when retrieving 100 passages.

A.2 Answer Recaller & Answer Verifier

Our answer recasters used an encoder length of 240 and a decoder length of 40; they were first pre-trained on NQ for 10 epochs and then finetuned on WebQSP/AMBIGQA for 80/20 epochs with early stopping. Batch size was set to 320. We trained the recasters to decode gold answers covered by a given positive passage (following the order they appear in the passage) and output the “irrelevant” token given a negative passage. Our best system adopts the recaster trained with \( \alpha_{neg}=0.1 \) because compared with \( \alpha_{neg}=0 \), the recaster trained with \( \alpha_{neg}=0.1 \) shrinks down nearly half of answer candidates without a significant drop in recall.

Our answer verifiers used an encoder length of 280; they were first pre-trained on NQ for 3 epochs and then finetuned on WebQSP/AMBIGQA for 30/10 epochs with early stopping. Batch size was set to 320 for \( k=1 \) and set to 64 for \( k \in \{5, 10\} \). The number of invalid answers used for training was set to 10 times the number of valid answers. The best threshold \( \tau \) was chosen from \{0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\} based on F1 scores on the dev set.

Though our framework focuses on multi-answer questions, we also experimented on NQ to demonstrate that our framework is applicable to single-answer scenario without suffering from low precision. Specifically, for each question, we only output the candidate with the highest validity score. As shown by Table 8, we slightly outperform previous state-of-the-art rerank-then-read systems.

B Experiments

B.1 Single-Answer QA Result

<table>
<thead>
<tr>
<th>System</th>
<th>( k )</th>
<th>T5</th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Izacard and Grave (2021a)</td>
<td>100 large</td>
<td>51.9</td>
<td>53.7</td>
<td></td>
</tr>
<tr>
<td>JPR</td>
<td>10</td>
<td>3b</td>
<td>50.4</td>
<td>54.5</td>
</tr>
<tr>
<td>Ours</td>
<td>10</td>
<td>3b</td>
<td><strong>52.8</strong></td>
<td><strong>54.8</strong></td>
</tr>
</tbody>
</table>

Table 8: Exact match scores of different systems on the single-answer dataset NQ. The column T5 shows the size of the readers of rerank-then-read systems and the size of the verifier of our recall-then-verify system. Izacard and Grave (2021a) adopted the rerank-then-read framework and used significantly more memory resources for training than JPR and our system.
Recaller vs. Reranker The time complexity of answer recalling is \(O(|B| \cdot (L_p^2 + L_n \cdot L_p + L_a^2))\), while that of passage reranking is \(O(|B| \cdot L_p^2 + k \cdot |B| \cdot L_p + k^2)\). As encoding dominates computation cost (whose time complexity is \(O(|B| \cdot L_p^2))\), given the same model size and \(|B|\), the time complexity of answer recalling and passage reranking is at the same level.

Verifier vs. Reader The time complexity of answer verification is \(O(|A| \cdot (k \cdot L_a^2 + k \cdot L_n))\), while that of the reader is \(O(k \cdot L_a^2 + L_n \cdot k \cdot L_p + L_a^2)\). As the reader decodes a sequence of length \(L_a\) in an autoregressive way, while the decoding length of the verifier is only 1, the ratio between the inference time of the verifier and that of the reader should be much less than \(|A|\).

Evidence Aggregator Evidence aggregation is significantly faster than answer recalling and verification, as representations of Wikipedia passages are pre-computed. The time complexity is \(O(|A| \cdot (L_p^2 + |B| \cdot \log k))\) where \(L_p^2\) comes from encoding a question-candidate pair with the retriever’s question encoder, and \(|B| \cdot \log k\) comes from selecting the top-\(k\) relevant passages for a candidate.

One can adjust the computation cost of a recall-then-verify system, depending on how much inference efficiency is valued over precision and recall, by (1) choosing a recaller model of proper time complexity\(^9\), (2) tuning \(\alpha_{neg}\) to adjust the expected number of candidates \(|A|\) needed for verification, (3) or tuning the number of passages \(k\) used for verification.

Table 9 shows QA performance and inference efficiency of our systems with different configurations on the dev set of AMBIGQA. \(Sec/Q\) denotes seconds per question when using a single V100 GPU. Answer verifiers were all initialized with T5-3b.

### Table 9: QA performance and inference efficiency of our systems with different configurations on the dev set

| T5 \(\alpha_{neg}\) \(Sec/Q\) | \(|A|\) Recall | Precision | \(\tau\) | \(Sec/Q\) | Overall F1 |
|---|---|---|---|---|---|
| 3b 0.1 | 4.88 28.7 72.6/60.5 6.3/7.5 | 0.02 | 0.5 | 4.83 | 9.73 52.1/41.6 |
| 3b 0.1 | 4.88 28.7 72.6/60.5 6.3/7.5 | 0.02 | 0.5 | 2.41 | 3.71 51.9/40.5 |
| base 0.1 | 0.85 48.4 70.95/79.7 3.3/4.1 | 0.03 | 0.6 | 8.56 | 9.44 50.9/40.8 |
| base 0.7 | 0.54 16.0 63.5/50.1 9.5/10.3 | 0.01 | 0.5 | 1.33 | 1.88 50.7/38.2 |

Recaller vs. Reranker

By also increasing \(\alpha_{neg}\) and choosing a smaller \(k\), as shown by the last row of Table 9, the overall time needed to answer a question on the dev set of AMBIGQA can be reduced to 1.88 sec on a single V100 GPU while also obtaining state-of-the-art F1 scores (50.7/38.2). By contrast, the rerank-then-read system from Min et al. (2021) using a T5-base JPR (\(k=10\)) and a T5-3b reader is estimated to take 1.51 sec per question\(^{10}\) with F1 scores of 48.5/37.6.

### B.3 Ablation Study on WebQSP

#### B.3.1 Answer Recalling

Table 10: Performance of recaller on WebQSP, trained with different models and \(\alpha_{neg}\). The recaller was paired with a verifier only in the last two rows which show end-to-end QA results.

By increasing \(\alpha_{neg}\) and choosing a smaller \(k\), as shown by the last row of Table 9, the overall time needed to answer a question on the dev set of AMBIGQA can be reduced to 1.88 sec on a single V100 GPU while also obtaining state-of-the-art F1 scores (50.7/38.2). By contrast, the rerank-then-read system from Min et al. (2021) using a T5-base JPR (\(k=10\)) and a T5-3b reader is estimated to take 1.51 sec per question\(^{10}\) with F1 scores of 48.5/37.6.

Table 10 shows the results of recaller on WebQSP, which were trained with different models and \(\alpha_{neg}\). In summary, a weak model suffices to recall answers with high coverage. Using a large and strong model for the recaller benefits precision, but it is still difficult for the recaller alone to answer open-domain multi-answer questions, which necessitates answer verification based on more associated evidence. However, an answer recaller can help reduce the burden on the answer verifier by conservatively filtering out negative candidates.

Though a recall-then-verify system with a recaller based on T5-base significantly outperforms JPR on WebQSP, it lags behind the system with a

---

\(^9\)As shown in Table 5, a smaller and faster answer recaller is capable of recalling answers with high coverage.

\(^{10}\)The average inference time of JPR from Min et al. (2021) is independent of its parameters given a fixed number of encoded tokens and a fixed decoder length, which can be estimated with a randomly initialized JPR. The average inference time of JPR’s reader was estimated with OPR’s reader.
T5-3b recaller on F1 (all) on the test set. We conjecture that this is because with the same $\alpha_{neg}=0.1$, a T5-base recaller obtains lower recall (69.5/61.4) than a T5-3b recaller (72.1/63.3); a T5-base recaller may need an even lower value of $\alpha_{neg}$ to obtain a higher coverage of gold answers.

B.3.2 Answer Verification

As shown by Figure 6, F1 scores on WEBQSP are insensitive to a wide range of $\tau$, while a lower $\tau$ is helpful to predict more gold answers.

C Error Analysis

Table 11 reports our error analysis on the dev set of AMBIGQA.
Missed Gold Answers > Evidence is wrong (24%)

**Question:** Who does brooke davis have a baby with?

**Gold Answers:** Julian Baker

**Missed Gold Answer:** Julian Baker

**Evidence:** Brooke Davis is happier than ever; preparing to marry Julian Baker ... The Scott family are expecting their second child and Haley feels the baby will be a girl ...

**Explanation:** Evidence is insufficient to infer Brooke Davis has a baby with Julian Baker.

Missed Gold Answers > Evidence is right and straightforward (76%)

**Question:** What’s the most points scored in an nba game?

**Gold Answers:** 370; 153; 162; 100; 186

**Missed Gold Answer:** 162

**Evidence:** The 1971-72 team holds franchise records in wins (69), most points scored, and largest margin of victory; both of the latter came in the team’s 63 point win versus Golden State (162-99).

Wrong Predictions > Predictions are true negatives (20%)

**Question:** When did the song lost boy come out?

**Gold Answers:** February 12, 2015; January 2015; 4 December 2015; May 9, 2016; 2015; 2017; November 17, 2017

**Prediction:** 20 December 2011

**Evidence:** “The Lost Boy” was written by Holden in 2011 ... Holden recorded it and released as a charity single on 20 December 2011 ...

**Explanation:** “Lost Boy” and “The Lost Boy” are different songs.

Wrong Predictions > Predictions are superficially-different false negatives (52%)

**Question:** How much sports are there in the winter olympics?

**Gold Answers:** fifteen; 86; 98; seven; 102

**Prediction:** 15

**Evidence:** ... the Winter Olympics programme features 15 sports.

Wrong Predictions > Predictions are unannotated false negatives (28%)

**Question:** How much did it cost rio to host the olympics?

**Gold Answers:** US$11.6 billion; US$13,100,000,000

**Prediction:** USD 4.6 billion

**Evidence:** Indirect capital costs were “not” included, such as for road ... Rio Olympics’ cost of USD 4.6 billion compares with costs of USD 40-44 billion for Beijing 2008 ...

Table 11: Analysis of predictions from our answer verifier. We display all annotated forms of gold answers, which are separated with semicolons.