Joint Passage Ranking for Diverse Multi-Answer Retrieval

Sewon Min[1]* **Kenton Lee**[2] **Ming-Wei Chang**[2] **Kristina Toutanova**[2] **Hannaneh Hajishirzi**[1]
[1]University of Washington [2]Google Research
{sewon,hannaneh}@cs.washington.edu
{kentonl,mingweichang,kristout}@google.com

Abstract

We study *multi-answer retrieval*, an under-explored problem that requires retrieving passages to cover multiple distinct answers for a given question. This task requires joint modeling of retrieved passages, as models should not repeatedly retrieve passages containing the same answer at the cost of missing a different valid answer. In this paper, we introduce JPR, the first joint passage retrieval model for multi-answer retrieval. JPR makes use of an autoregressive reranker that selects a sequence of passages, each conditioned on previously selected passages. JPR is trained to select passages that cover new answers at each timestep and uses a tree-decoding algorithm to enable flexibility in the degree of diversity. Compared to prior approaches, JPR achieves significantly better answer coverage on three multi-answer datasets. When combined with downstream question answering, the improved retrieval enables larger answer generation models since they need to consider fewer passages, establishing a new state-of-the-art.

1 Introduction

Passage retrieval is the problem of retrieving a set of passages relevant to a natural language question from a large text corpus. Most prior work focuses on single-answer retrieval, which scores passages independently from each other according to their relevance to the given question, assuming there is a single answer (Voorhees et al., 1999; Chen et al., 2017; Lee et al., 2019). However, questions posed by humans are often open-ended and ambiguous, leading to multiple valid answers (Min et al., 2020). For example, for the question in Figure 1, "What was Eli Whitney's job?", an ideal retrieval system should provide passages covering all professions of Eli Whitney. This introduces the problem of *multi-answer retrieval*—retrieval of multiple passages with maximal coverage of all distinct answers—which is a challenging yet understudied problem.

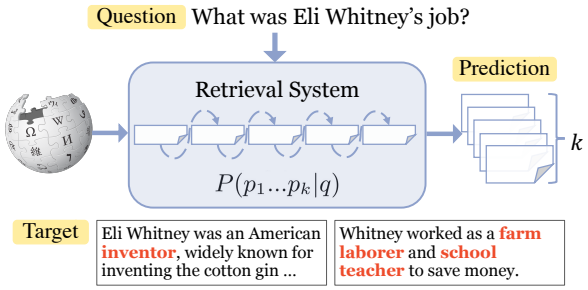

Figure 1: The problem of multi-answer retrieval. A retrieval system must retrieve a set of k passages ($k = 5$ in the figure) which has maximal coverage of diverse answers to the input question: *inventor*, *farm laborer* and *school teacher* in this example. This requires modeling the joint probability of the passages in the output set: $P(p_1...p_k|q)$. Our proposed model JPR achieves this by employing an autoregressive model.

Multi-answer retrieval poses two challenges that are not well represented in single-answer retrieval. First, the task requires scoring passages jointly to optimize for retrieving multiple relevant-yet-complementary passages. Second, the model needs to balance between two different goals: retrieving passages dissimilar to each other to increase the recall, and keeping passages relevant to the question.

In this work, we introduce Joint Passage Retrieval (JPR), a new model that addresses these challenges. To jointly score passages, JPR employs an encoder-decoder reranker and autoregressively generates passage references by modeling the probability of each passage as a function of previously retrieved passages. Since there is no ground truth ordering of passages, we employ a new training method that dynamically forms supervision to drive the model to prefer passages with answers not already covered by previously selected passages. Furthermore, we introduce a new tree-decoding algorithm to allow flexibility in the degree of diversity.

*Work done while interning at Google.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 6997–7008
November 7–11, 2021.

In a set of experiments on three multi-answer datasets—WEBQSP (Yih et al., 2016), AMBIGQA (Min et al., 2019) and TREC (Baudiš and Šedivỳ, 2015), JPR achieves significantly improved recall over both a dense retrieval baseline (Guu et al., 2020; Karpukhin et al., 2020) and a state-of-the-art reranker that independently scores each passage (Nogueira et al., 2020). Improvements are particularly significant on questions with more than one answer, outperforming dense retrieval by up to 12% absolute and an independent reranker by up to 6% absolute.

We also evaluate the impact of JPR in downstream question answering, where an answer generation model takes the retrieved passages as input and generates short answers. Improved reranking leads to improved answer accuracy because we can supply fewer, higher-quality passages to a larger answer generation model that fits on the same hardware. This practice leads to a new state-of-the-art on three multi-answer QA datasets and NQ (Kwiatkowski et al., 2019). To summarize, our contributions are as follows:

1. We study multi-answer retrieval, an underexplored problem that requires the top k passages to maximally cover the set of distinct answers to a natural language question.
2. We propose JPR, a joint passage retrieval model that integrates dependencies among selected passages, along with new training and decoding algorithms.
3. On three multi-answer QA datasets, JPR significantly outperforms a range of baselines with independent scoring of passages, both in retrieval recall and answer accuracy.

2 Background

2.1 Review: Single-Answer Retrieval

In a typical single-answer retrieval problem, a model is given a natural language question q and retrieves k passages $\{p_1...p_k\}$ from a large text corpus $\mathcal{C}$ (Voorhees et al., 1999; Ramos et al., 2003; Robertson and Zaragoza, 2009; Chen et al., 2017; Lee et al., 2019; Karpukhin et al., 2020; Luan et al., 2020). The goal is to retrieve at least one passage that contains the answer to q. During training, question-answer pairs (q, a) are given to the model.

Evaluation *Intrinsic* evaluation directly evaluates the retrieved passages. The most commonly used metric is RECALL @ k which considers retrieval successful if the answer a is included in $\{p_1...p_k\}$. *Extrinsic* evaluation uses the retrieved passages as input to an answer generation model such as the model in Izacard and Grave (2021) and evaluates final question answering performance.

Task	**Single-answer Retrieval**	**Multi-answer Retrieval**
Train Data	(q, a)	$(q, \{a_1...a_n\})$
Inference	$q \rightarrow \{p_1...p_k\}$	$q \rightarrow \{p_1...p_k\}$
Evaluation	RECALL($a, \{p_1...p_k\}$)	MRECALL($\{a_1...a_n\}$, $\{p_1...p_k\}$)
Appropriate Model	$P(p_i\|q)$	$P(p_1...p_k\|q)$

Table 1: A comparison of single-answer and multi-answer retrieval tasks. Previous work has used independent ranking models $P(p_i|q)$ for multi-answer retrieval because the inference-time inputs and outputs are the same. We propose JPR as an instance of $P(p_1...p_k|q)$.

Reranking Much prior work (Liu, 2011; Asadi and Lin, 2013; Nogueira et al., 2020) found an effective strategy in using a two-step approach of (1) retrieving a set of candidate passages $\mathcal{B}$ from the corpus $\mathcal{C}$ ($k < |\mathcal{B}| \ll |\mathcal{C}|$) and (2) using another model to rerank the passages, obtaining a final top k. A reranker could be more expressive than the first-stage model (e.g. by using cross-attention), as it needs to process much fewer candidates. Most prior work in reranking, including the current state-of-the-art (Nogueira et al., 2020), scores each passage independently, modeling $P(p|q)$.

2.2 Multi-Answer Retrieval

We now formally define the task of multi-answer retrieval. A model is given a natural language question q and needs to find k passages $\{p_1...p_k\}$ from $\mathcal{C}$ that contain *all distinct answers* to q. Unlike in single-answer retrieval, question-and-answer-set pairs $(q, \{a_1...a_n\})$ are given during training.

Evaluation Similar to single-answer retrieval, the *intrinsic* evaluation directly evaluates a set of k passages. As the problem is underexplored, metrics for it are less studied. We propose to use MRECALL @ k, a new metric which considers retrieval to be successful if all answers or at least k answers in the answer set $\{a_1...a_n\}$ are recovered by $\{p_1...p_k\}$. Intuitively, MRECALL is an extension of RECALL that considers the completeness of the retrieval; the model must retrieve all n answers

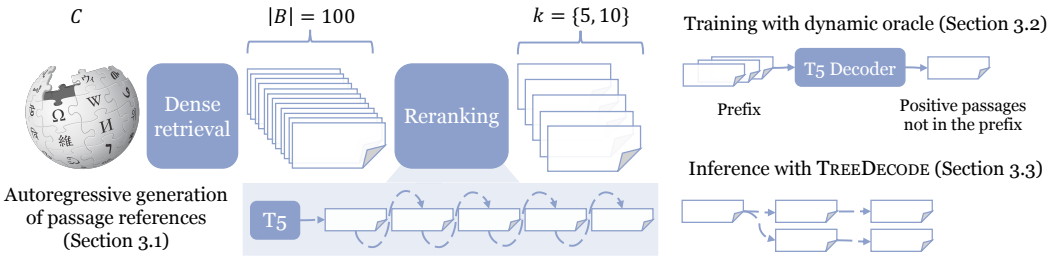

Figure 2: An overview of JPR. We focus on reranking and propose: autoregressive generation of passage references (Section 3.1), training with dynamic oracle (Section 3.2), and inferece with TREEDECODE (Section 3.3).

when $n \leq k$, or at least k answers when $n > k$.[1]

The *extrinsic* evaluation inputs the retrieved passages into an answer generation module that is designed for multiple answers, and measures multi-answer accuracy using an appropriate metric such as the one in Min et al. (2020).

Comparing to single-answer retrieval We compare single-answer retrieval and multi-answer retrieval in Table 1. Prior work makes no distinctions between these two problems, since they share the same interface during inference. However, while independently scoring each passage ($P(p_i|q)$) may be sufficient for single-answer retrieval, multi-answer retrieval inherently requires *joint* passage scoring $P(p_1...p_k|q)$. For example, models should not repeatedly retrieve the same answer at the cost of missing other valid answers, which can only be done by a joint model.

Choice of *k* for downstream QA Previous state-of-the-art models typically input a large number ($k \geq 100$) of passages to the answer generation model. For instance, Izacard and Grave (2021) claim the importance of using a larger value of k to improve QA accuracy. In this paper, we argue that with reranking, using a smaller value of k (5 or 10) and instead employing a larger answer generation model is advantageous given a fixed hardware budget.[2] We show in Section 5 that, as retrieval performance improves, memory is better spent on larger answer generators rather than on more passages, ultimately leading to higher QA accuracy.

3 JPR: Joint Passage Retrieval

We propose JPR (Joint Passage Retrieval), which models the joint probability $P(p_1...p_k|q)$ for multi-answer retrieval. JPR uses an approach consisting of first-stage retrieval followed by reranking: the first-stage retrieval obtains candidate passages $\mathcal{B}$ from the corpus $\mathcal{C}$, and a reranker processes $\mathcal{B}$ to output $\{p_1...p_k\} \subset \mathcal{B}$. We refer to Section 4.2 for the first-stage retrieval, and focus on the reranking component of the model, which allows (1) efficiently modeling the joint probability $P(p_1...p_k|q)$, and (2) processing candidate passages with a more expressive model.

The overview of JPR is illustrated in Figure 2. The reranker of JPR leverages the encoder-decoder architecture for an autoregressive generation of passage references (Section 3.1). Unlike typical use cases of the encoder-decoder, (1) the ordering of passages to retrieve is not given as supervision, and (2) it is important to balance between exploring passages about new answers and finding passages that may cover previously selected answers. To this end, we introduce a new training method (Section 3.2) and a tree-based decoding algorithm (Section 3.3).

3.1 Autoregressive generation of passage references

JPR makes use of the encoder-decoder architecture, where the encoder processes candidate passages and the decoder autoregressively generates a sequence of k passage references (*indexes*). Intuitively, dependencies between passages can be modeled by the autoregressive architecture.

We extend the architecture from Izacard and Grave (2021); we reuse the encoder but modify the decoder. Each candidate passage p_i is concatenated with the question q and the number i (namely *index*). It is fed into the encoder to be transformed to $\mathbf{p}_i \in \mathbb{R}^{L \times h}$, where L is the length of the input text and h is a hidden size. Next, $\mathbf{p}_1...\mathbf{p}_{|\mathcal{B}|}$ are concatenated to form $\bar{\mathbf{p}} \in \mathbb{R}^{L|\mathcal{B}| \times h}$, and then fed into the decoder. The decoder is trained to autoregressively output a sequence of indexes $i_1...i_k$,

[1] This is to handle the cases where n is very large (e.g. over 100) and the model covers a reasonable number of answers given the limit of k passages, therefore deserves credit.

[2] We care about a fixed type of hardware since it is the hardest constraint and usually a bottleneck for performance. We do not control for running time in this comparison.

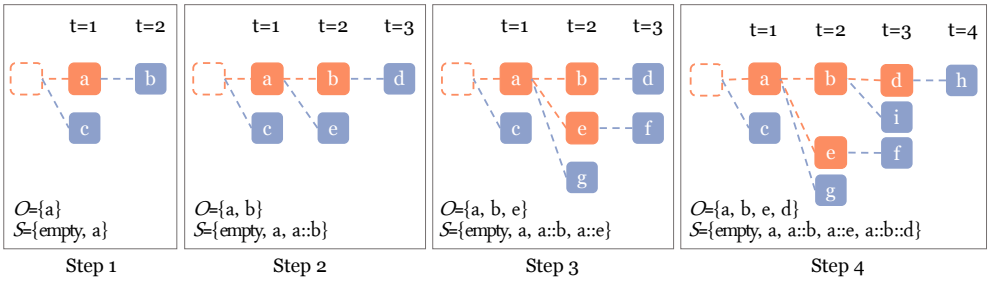

Figure 3: An illustration of TREEDECODE, where passages that are chosen and passages that are being considered are indicated in orange and blue, respectively. See Section 3.3 and Algorithm 1 for details.

representing a sequence of passages $p_1 ... p_k$. As the generation at step t ($1 \leq t \leq k$) is dependent on the generation at step $1 \ldots t-1$, it can naturally capture dependencies between selected passages. As each index occupies one token, the length of the decoded sequence is k.

3.2 Training with Dynamic Oracle

A standard way of training the encoder-decoder is teacher forcing which assumes a single correct sequence. However, in our task, a set of answers can be retrieved through many possible sequences of passages, and it is unknown which sequence is the best. To this end, we dynamically form the supervision data which pushes the model to assign high probability to a *dynamic oracle*—any positive passage covering a correct answer that is not in the prefix, i.e., previously selected passages.

We first precompute a set of positive passages $\tilde{\mathcal{O}}$ and a prefix $\tilde{p}_1 ... \tilde{p}_k$. A set of positive passages $\tilde{\mathcal{O}}$ includes up to k passages with maximal coverage of the distinct answers.[3] A prefix $\tilde{p}_1 ... \tilde{p}_k$ is a simulated prediction of the model, consisting of $\tilde{\mathcal{O}}$ and $k - |\tilde{\mathcal{O}}|$ sampled negatives. At each step t ($1 \leq t \leq k$) , given a set of positive passages $\tilde{\mathcal{O}}$ and a prefix $\tilde{p}_1 ... \tilde{p}_t$ (denoted as $\mathcal{P}_t$), JPR is trained to assign high probabilities to the dynamic oracle $\tilde{\mathcal{O}} - \mathcal{P}_t$. The objective is defined as follows:

$$\sum_{1 \leq t \leq k} \sum_{o \in \tilde{\mathcal{O}} - \mathcal{P}_{t-1}} -\log P(o|q, \mathcal{B}, \mathcal{P}_{t-1}).$$

3.3 Inference with TREEDECODE

A typical autoregressive decoder makes the top 1 prediction at each step to decode a sequence of k (SEQDECODE in Algorithm 1),[4] which, based on our training scheme, asks the decoder to find a new answer at every step. However, when k is larger than the number of correct answers, it would be counter-productive to ask for k passages that each covers a distinct answer. Instead, we want the flexibility of decoding fewer timesteps and take multiple predictions from each timestep.

Algorithm 1 Decoding algorithms for JPR.

procedure SEQDECODE($k, P(p|p_1 ... p_n)$)
  $O \leftarrow []$ *// a list of selected passages*
  while $i = 1, \ldots, k$ **do**
    $\hat{p} \leftarrow \text{argmax}_{p \in \mathcal{B} - O.\text{toSet}()} \log P(p|O)$
    $O \leftarrow O :: \hat{p}$
  return $\text{Set}(O)$
procedure TREEDECODE($k, P(p|p_1 ... p_n), l$)
  $\mathcal{O} \leftarrow \emptyset$ *// a set of selected passages*
  $\mathcal{S} \leftarrow [\texttt{Empty}]$ *// a tree*
  while $|\mathcal{O}| < k$ **do**
    $P'(p|s) \leftarrow P(p|s)\mathbb{I}[s :: p \notin \mathcal{S}]$
    $(\hat{s}, \hat{p}) \leftarrow \text{argmax}_{p \in \mathcal{B}, s \in \mathcal{S}} l(|s|+1) \log P'(p|s)$
    $\mathcal{O} \leftarrow \mathcal{O} \cup \{\hat{p}\}, \mathcal{S} \leftarrow \mathcal{S}.\text{append}(\hat{s} :: \hat{p})$
  return $\mathcal{O}$

In this context, we introduce a new decoding algorithm TREEDECODE, which decodes a *tree* from an autoregressive model. TREEDECODE iteratively chooses between the depth-wise and the width-wise expansion of the tree—going forward to the next step and taking the next best passage within the same step, respectively—until it reaches k passages (Figure 3). Intuitively, if the model believes that there are many distinct answers covered by different passages, it will choose to take the next step, being closer to SEQDECODE. On the other hand, if the model believes that there are very few distinct answers, it will choose to take more predictions from the same step, resulting in behavior closer to independent scoring.

The formal algorithm is as follows. We represent a tree $\mathcal{S}$ as a list of ordered lists $[s_1 ... s_T]$ where s_1 is an empty list and s_i is one element appended to any of $s_1 ... s_{i-1}$. The corresponding set $\mathcal{O}$ is

[3] $|\tilde{\mathcal{O}}| < k$ if fewer than k passages are sufficient to cover all distinct answers; $|\tilde{\mathcal{O}}| = k$ otherwise.

[4] We explored beam search decoding but it gives results that are the same as or marginally different from SEQDECODE.

Dataset	# questions %			# answers	
	Train	Dev	Test	Avg.	Median
WEBQSP	2,756	241	1,582	12.4	2.0
AMBIGQA	10,036	2,002	2,004	2.2	2.0
TREC	1,250	119	654	4.1	2.0

Table 2: Number of questions and an average & median number of the answers on the development data. Data we use for TREC is a subset of the data from Baudiš and Šedivỳ (2015) as described in Appendix B.1.

$\cup_{s \in \mathcal{S}} \text{Set}(s)$. We define a score of a tree $\mathcal{S}$ as

$$f(\mathcal{S}) = \sum_{p_1 \ldots p_{t_i} \in \mathcal{S}} \log P(p_{t_i} | p_1 \ldots p_{t_{i-1}}).$$

We form $\mathcal{S}$ and $\mathcal{O}$ through an iterative process by (1) starting from $\mathcal{O} = \emptyset$ and $\mathcal{S} = [\texttt{null}]$, and (2) updating $\mathcal{O}$ and $\mathcal{S}$ by finding the best addition of an element that maximizes the gain in $f(\mathcal{S})$, until $|\mathcal{O}| = k$, as described in Algorithm 1.

4 Experimental Setup

We compare JPR with multiple baselines in a range of multi-answer QA datasets. We first present an intrinsic evaluation of passage retrieval by reporting MRECALL based on answer coverage in the retrieved passages (Section 5.1). We then present an extrinsic evaluation through experiments in downstream question answering (Section 5.2).

4.1 Datasets

We train and evaluate on three datasets that provide a set of distinct answers for each question. Statistics of each dataset are provided in Table 2.

WEBQSP (Yih et al., 2016) consists of questions from Google Suggest API, originally from Berant et al. (2013). The answer is a set of distinct entities in Freebase; we recast this problem as textual question answering based on Wikipedia.

AMBIGQA (Min et al., 2020) consists of questions mined from Google search queries, originally from NQ (Kwiatkowski et al., 2019). Each question is paired with an annotated set of distinct answers that are equally valid based on Wikipedia.

TREC (Baudiš and Šedivỳ, 2015) contains questions curated from TREC QA tracks, along with regular expressions as answers. Prior work uses this data as a task of finding a single answer (where retrieving any of the correct answers is sufficient), but we recast the problem as a task of finding all answers, and approximate a set of distinct answers. Details are described in Appendix B.1.

4.2 First-stage Retrieval

JPR can obtain candidate passages $\mathcal{B}$ from any first-stage retrieval model. In this paper, we use DPR$^+$, our own improved version of DPR (Karpukhin et al., 2020) combined with REALM (Guu et al., 2020). DPR and REALM are dual encoders with a supervised objective and an unsupervised, language modeling objective, respectively. We initialize the dual encoder with REALM and train on supervised datasets using the objective from DPR. More details are provided in Appendix A.

4.3 Baselines

We compare JPR with three baselines, all of which are published models or enhanced versions of them. All baselines independently score each passage.

DPR$^+$ only uses DPR$^+$ without a reranker.

DPR$^+$+Nogueira et al. (2020) uses DPR$^+$ followed by Nogueira et al. (2020), the state-of-the-art document ranker. It processes each passage p_i in $\mathcal{B}$ independently and is trained to output `yes` if p_i contains any valid answer to q, otherwise `no`. At inference, the probability for each p_i is computed by taking a softmax over the logit of `yes` and `no`. The top k passages are chosen based on the probabilities assigned to `yes`.

INDEPPR is our own baseline that is a strict non-autoregressive version of JPR in which prediction of a passage is independent from other passages in the retrieved set. It obtains candidate passages $\mathcal{B}$ through DPR$^+$ and the encoder of the reranker processes q and $\mathcal{B}$, as JPR does. Different from JPR, the decoder is trained to output a single token i ($1 \leq i \leq |\mathcal{B}|$) rather than a sequence. The objective is the sum of $-\log P(p|q, \mathcal{B})$ of the passages including any valid answer to q. At inference, INDEPPR outputs the top k passages based the logit values of the passage indices. We compare mainly to INDEPPR because it is the strict non-autoregressive version of JPR, and is empirically better than or comparable to Nogueira et al. (2020) (Section 5.1).

4.4 Implementation Details

We use the English Wikipedia from 12/20/2018 as the retrieval corpus $\mathcal{C}$, where each article is split into passages with up to 288 wordpieces, All rerankers are based on T5 (Raffel et al., 2020), a pretrained encoder-decoder model; T5-base is used unless otherwise specified. We use $|\mathcal{B}| = 100, k = \{5, 10\}$. Models are first trained on NQ (Kwiatkowski et al., 2019) and then finetuned

k	Models	WEBQSP		AMBIGQA	TREC	
		Dev	Test	Dev	Dev	Test
5	DPR$^+$ only	56.4/37.8	57.0/38.9	55.2/36.3	53.8/**29.9**	57.8/36.6
	DPR$^+$+Nogueira et al. (2020)	60.2/40.9	60.2/39.9	63.4/43.1	53.8/28.4	61.0/39.5
	INDEPPR	60.6/40.2	62.9/45.2	63.7/43.7	53.8/28.4	**62.4/41.1**
	JPR	**68.5/56.7**	**64.9/50.6**	**64.8/45.2**	**55.5/29.9**	**62.4/41.1**
10	DPR$^+$ only	61.4/42.5	59.0/38.6	59.3/39.6	55.5/28.4	60.1/38.4
	DPR$^+$+Nogueira et al. (2020)	64.7/45.7	62.9/41.5	65.8/46.4	55.5/28.4	**64.8**/43.0
	INDEPPR	65.6/47.2	63.3/43.1	65.5/46.2	53.8/26.9	63.8/42.2
	JPR	**68.9/55.1**	**65.7/48.9**	**67.1/48.2**	**56.3/29.9**	64.5/**43.3**

Table 3: Results on passage retrieval in MRECALL. The two numbers in each cell indicate performance on all questions and on questions with more than one answer, respectively. Test-set metrics on AMBIGQA are not available as its test set is hidden, but we report the test results on question answering in Section 5.2.
Note: it is possible to have higher MRECALL @ 5 than MRECALL @ 10 based on our definition of MRECALL (Section 2.2).

Training method	MRECALL
Dynamic oracle	**67.6/56.7**
Dynamic oracle w/o negatives	65.1/52.0
Teacher forcing	66.4/51.2

Table 4: Ablations in training methods for JPR. Results on WEBQSP ($k = 5$). All rows use SEQDECODE (instead of TREEDECODE).

k	Decoding	WEBQSP		AMBIGQA	
		d	MRECALL	d	MRECALL
5	SEQDECODE	5.0	67.6/**56.7**	5.0	63.1/42.5
	TREEDECODE	3.0	**68.5/56.7**	2.1	**64.8/45.2**
10	SEQDECODE	10.0	68.0/54.3	10.0	65.0/45.9
	TREEDECODE	5.4	**68.9/55.1**	2.9	**67.1/48.2**

Table 5: Ablations in decoding methods for JPR. d refers to the average depth of the tree ($\max_{s \in \mathcal{S}} |s|$ in Algorithm 1).

on multi-answer datasets, which we find helpful since all multi-answer datasets are relatively small. During dynamic oracle training, $k - |\tilde{\mathcal{O}}|$ negatives are sampled from $\mathcal{B} - \tilde{\mathcal{O}}$ based on $s(p_i) + \gamma g_i$, where $s(p_i)$ is a prior logit value from INDEPPR, $g_i \sim \text{Gumbel}(0, 1)$ and γ is a hyperparameter. In TREEDECODE, to control the trade-off between the depth and the width of the tree, we use a length penalty function $l(y) = \left(\frac{5+y}{5+1}\right)^{\beta}$, where β is a hyperparameter, following Wu et al. (2016). More details are in Appendix B.2.

5 Experimental Results

5.1 Retrieval Experiments

Table 3 reports MRECALL on all questions and on questions with more than one answer.

No reranking vs. reranking Models with reranking (DPR$^+$+Nogueira et al. (2020), INDEPPR or JPR) are always better than DPR$^+$ only, demonstrating the importance of reranking.

Independent vs. joint ranking JPR consistently outperforms both DPR$^+$+Nogueira et al. (2020) and INDEPPR on all datasets and all values of k. Gains are especially significant on questions with more than one answer, outperforming two reranking baselines by up to 11% absolute and up to 6% absolute, respectively. WEBQSP sees the largest gains out of the three datasets, likely because the average number of answers is the largest.

5.1.1 Ablations & Analysis

Training methods Table 4 compares dynamic oracle training with alternatives. 'Dynamic oracle w/o negatives' is the same as dynamic oracle training except the prefix only has positive passages. 'Teacher forcing' is a standard method in training an autoregressive model: given a target sequence $o_1 ... o_k$, the model is trained to maximize $\Pi_{1 \le t \le k} P(o_t | o_1 ... o_{t-1})$. We form a target sequence using a set of positive passages $\tilde{\mathcal{O}}$, where the order is determined by following the ranking from INDEPPR. Table 4 shows that our dynamic oracle training, which uses both positives and negatives, significantly outperforms the other methods.

Impact of TREEDECODE Table 5 compares JPR with SEQDECODE and with TREEDECODE. We find that TREEDECODE consistently improves the performance on both WEBQSP and AMBIGQA, with both $k = 5$ and 10. Gains are especially significant on AMBIGQA, since the choice of whether to increase diversity is more challenging on AMBIGQA where questions are more specific and have fewer distinct answers, which TREEDE-

Q: Who play Mark on the TV show Roseanne?	
INDEPPR	JPR
#1 Glenn Quinn ... He was best known for his portrayal of Mark Healy on the popular '90s family sitcom Roseanne.	**#1** Glenn Quinn ... He was best known for his portrayal of Mark Healy on the popular '90s family sitcom Roseanne.
#2 Glenn Quinn, who played Becky's husband, Mark, died in December 2002 of a heroin overdose at the age of 32 ...	**#2** Becky begins dating Mark Healy (Glenn Quinn) ...
#3 Becky begins dating Mark Healy (Glenn Quinn) ...	**#3** Glenn Quinn, who played Becky's husband, Mark, died in December 2002 of a heroin overdose at the age of 32 ...
#4 Johnny Galecki ... on the hit ABC sitcom Roseanne as the younger brother of Mark Healy (Glenn Quinn) ...	**#4** Roseanne (season 10) ... In September 2017, Ames McNamara was announced to be cast as Mark Conner-Healy.

Table 6: An example prediction from INDEPPR and JPR; answers to the input question highlighted. While INDEPPR repeatedly retrieves passages supporting the same answer *Glenn Quinn* and fails to cover other answers, JPR successfully retrieves a passage covering a novel answer *Ames McNamara*.

CODE better handles compared to SEQDECODE. The average depth of the tree is larger on WEBQSP, likely because its average number of distinct answers is larger and thus requires more diversity.

An example prediction Table 6 shows predictions from INDEPPR and JPR given an example question from AMBIGQA, "Who plays Mark on the TV show Roseanne?" One answer *Glenn Quinn* is easy to retrieve because there are many passages in Wikipedia providing evidence, while the other answer *Ames McNamara* is harder to find. While INDEPPR repeatedly retrieves passages that mention *Glenn Quinn* and fails to cover *Ames McNamara*, JPR successfully retrieves both answers.

More analysis can be found in Appendix C.

5.2 QA Experiments

This section discusses experiments on downstream question answering: given a question and a set of passages from retrieval, the model outputs all valid answers to the question. We aim to answer two research questions: (1) whether the improvements in passage retrieval are transferred to improvements in downstream question answering, and (2) whether using a smaller number of passages through reranking is better than using the largest possible number of passages given fixed hardware memory.

We use an answer generation model based on Izacard and Grave (2021) which we train to generate a sequence of answers, separated by a `[SEP]` token, given a set of retrieved passages. Our main model uses JPR to obtain passages fed into the answer generation model. The baselines obtain passages from either **DPR$^+$ only** or **INDEPPR**, described in Section 4.3.

We compare different models that fit on the same hardware by varying the sizes of T5 (base, large, 3B) and use the maximum number of passages (k).[5] This results in three settings: $\{k = 140, \text{base}\}$, $\{k = 40, \text{large}\}$ and $\{k = 10, \text{3B}\}$.

5.2.1 Main Result

Table 7 reports the performance on three multi-answer datasets in F1, following Min et al. (2020).

Impact of reranking With $\{k = 10, \text{3B}\}$, JPR outperforms both baselines, indicating that the improvements in retrieval are successfully transferred to improvements in QA performance. We however find that our sequence-to-sequence answer generation model tends to undergenerate answers, presumably due to high variance in the length of the output sequence. This indicates the model is not fully benefiting from retrieval of many answers, and we expect more improvements when combined with an answer generation model that is capable of generating many answers.

More passages vs. bigger model With fixed memory during training, using fewer passages equipped with a larger answer generation model outperforms using more passages. This is only true when reranking is used; otherwise, using more passages is often better or comparable. This demonstrates that, as retrieval improves, memory is better spent on larger answer generators rather than more passages, leading to the best performance.

Finally, JPR establishes a new state-of-the-art, outperforming the previous state-of-the-art on AMBIGQA (Gao et al., 2021) with extensive reranking and the answer generation model trained using x3 more resources than ours.[6]

5.2.2 Single-answer QA result

While our main contributions are in multi-answer retrieval, we experiment on NQ to demonstrate

[5] The memory requirement is $O(k \times \text{T5 size})$.

[6] Gao et al. (2021) reranks 1000 passages through independent scoring as in Nogueira et al. (2020); it is not a directly comparable baseline and serves as a point of reference.

Retrieval	QA Model	k	Mem	WEBQSP		AMBIGQA		TREC	
				Dev	Test	Dev	Test	Dev	Test
DPR$^+$ only	T5-3B	10	x1	50.7/45.3	51.9/45.0	43.5/34.6	39.6/31.4	46.2/32.2	44.7/32.1
INDEPPR	T5-3B	10	x1	51.8/46.9	51.8/45.0	47.6/36.2	42.3/32.0	44.6/32.8	45.9/31.8
JPR	T5-3B	10	x1	**53.6/49.5**	**53.1/47.2**	**48.5/37.6**	**43.5/34.2**	**48.6/32.8**	**46.8/33.3**
DPR$^+$ only	T5-large	40	x1	51.4/47.0	52.4/45.8	45.5/34.9	41.1/30.9	40.1/32.8	42.5/32.2
Gao et al. (2021)	BART-large	100	x3	-	-	48.3/37.3	42.1/33.3	-	-

Table 7: Question Answering results on multi-answer datasets. The two values in each cell indicate F1 on all questions and F1 on questions with multiple answers only, respectively. *Mem* compares the required hardware memory during training. Note that Gao et al. (2021) reranks 1000 passages instead of 100, and trains an answer generation model using x3 more memory than ours. **Better retrieval enables using larger answer generation models on fewer retrieved passages.**

Model	T5	k	dev	test
DPR$^+$ only	base	140	46.4	-
DPR$^+$ only	large	40	47.3	-
DPR$^+$ only	3B	10	46.5	-
INDEPPR	large	40	49.4	-
INDEPPR	3B	10	50.4	**54.5**
Izacard and Grave (2021)			-	51.4

Table 8: Question Answering results on NQ. We report Exact Match (EM) accuracy. The first five rows are from our own experiments, which all use the same hardware resources for training. The last row is the previous state-of-the-art which requires x5 more resources than ours to train the model.

that the value of good reranking extends to the single-answer scenario. Table 8 indicates two observations consistent to the findings from multi-answer retrieval: (1) when compared within the same setting (same T5 and k), INDEPPR always outperforms DPR$^+$ only, and (2) with reranking, $\{k = 10, \text{3B}\}$ outperforms $\{k = 40, \text{large}\}$. Finally, our best model outperforms the previous state-of-the-art (Izacard and Grave, 2021) which uses x5 more training resources. Altogether, this result (1) justifies our choice of focusing on reranking, and (2) shows that INDEPPR is very competitive and thus our JPR results in multi-answer retrieval are very strong.

6 Related Work

We refer to Section 2 for related work focusing on single-answer retrieval.

Diverse retrieval Studies on diverse retrieval in the context of information retrieval (IR) requires finding documents covering many different subtopics to a query topic (Zhai et al., 2003; Clarke et al., 2008). Questions are typically underspecified, and many documents (e.g. up to 56 in Zhai et al. (2003)) are considered relevant. In their problem space, effective models post-hoc increase the distances between output passages during inference (Zhai et al., 2003; Abdool et al., 2020).

Our problem is closely related to diverse retrieval in IR, with two important differences. First, since questions represent more specific information needs, controlling the trade-off between relevance and diversity is harder, and simply increasing the distances between retrieved passages does not help.[7] Second, multi-answer retrieval uses a clear notion of "answers"; "sub-topics" in diverse IR are more subjective and hard to enumerate fully.

Multi-hop passage retrieval Recent work studies multi-hop passage retrieval, where a passage containing the answer is the destination of a chain of multiple hops (Asai et al., 2020; Xiong et al., 2021; Khattab et al., 2021). This is a difficult problem as passages in a chain are dissimilar to each other, but existing datasets often suffer from annotation artifacts (Chen and Durrett, 2019; Min et al., 2019), resulting in strong lexical cues for each hop. We study an orthogonal problem of finding multiple answers, where the challenge is in controlling the trade-off between relevance and diversity.

7 Conclusion

We introduce JPR, an autoregressive passage reranker designed to address the multi-answer retrieval problem. On three multi-answer datasets, JPR significantly outperforms a range of baselines

[7] In our preliminary experiment, we tried increasing diversity based on Maximal Marginal Relevance (Carbonell and Goldstein, 1998) following Zhai et al. (2003); Abdool et al. (2020); it improves diversity but significantly hurts the relevance to the input question, dropping the overall performance.

in both retrieval recall and downstream QA accuracy, establishing a new state-of-the-art. Future work could extend the scope of the problem to other tasks that exhibit specific information need while requiring diversity.

Acknowledgements

We thank the Google AI Language members, the UW NLP members, and the anonymous reviewers for their valuable feedback. This work was supported in part by ONR N00014-18-1-2826 and DARPA N66001-19-2-403.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems.

Mustafa Abdool, Malay Haldar, Prashant Ramanathan, Tyler Sax, Lanbo Zhang, Aamir Mansawala, Shulin Yang, and Thomas Legrand. 2020. Managing diversity in airbnb search. In *ACM SIGKDD*.

Nima Asadi and Jimmy Lin. 2013. Effectiveness/efficiency tradeoffs for candidate generation in multi-stage retrieval architectures. In *SIGIR*.

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *ICLR*.

Petr Baudiš and Jan Šedivỳ. 2015. Modeling of the question answering task in the yodaqa system. In *International Conference of the Cross-Language Evaluation Forum for European Languages*.

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

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*.

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

Jifan Chen and Greg Durrett. 2019. Understanding dataset design choices for multi-hop reasoning. In *NAACL*.

Charles LA Clarke, Maheedhar Kolla, Gordon V Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. 2008. Novelty and diversity in information retrieval evaluation. In *SIGIR*.

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

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-augmented language model pre-training. In *ICML*.

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

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *TOIS*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP*.

Omar Khattab, Christopher Potts, and Matei Zaharia. 2021. Baleen: Robust multi-hop reasoning at scale via condensed retrieval. *arXiv preprint arXiv:2101.00436*.

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

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *ACL*.

Tie-Yan Liu. 2011. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2020. Sparse, dense, and attentional representations for text retrieval. *TACL*.

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

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. Compositional questions do not necessitate multi-hop reasoning. In *ACL*.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pre-trained sequence-to-sequence model. In *Findings of EMNLP*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*.

Tetsuya Sakai and Zhaohao Zeng. 2019. Which diversity evaluation measures are" good"? In *SIGIR*.

Noam M. Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, H. Lee, Mingsheng Hong, C. Young, Ryan Sepassi, and Blake A. Hechtman. 2018. Mesh-tensorflow: Deep learning for supercomputers. In *NeurIPS*.

Ellen M Voorhees et al. 1999. The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82. Citeseer.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

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

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*.

Cheng Zhai, William W Cohen, and John Lafferty. 2003. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *SIGIR*.

A Details of DPR$^+$

We use a pretrained dual encoder model from REALM (Guu et al., 2020) and further finetune it on the QA datasets using the objective from DPR (Karpukhin et al., 2020):

$$\mathcal{L} = -\log \frac{f_q(q)^T f_p(p^+)}{\sum_{p \in \{p^+\} \cup \mathcal{B}^-} f_q(q)^T f_p(p)},$$

where f_q and f_p are trainable encoders for the questions and passages, respectively, p^+ is a positive passage (i.e., a passage containing the answer), and $\mathcal{B}^-$ is a set of negative passages (i.e., passages without the answer). As shown in Karpukhin et al. (2020), a choice of $\mathcal{B}^-$ is significant for the performance. We explore two methods:

Distant negatives follows DPR (Karpukhin et al., 2020) in using distantly obtained negative passages as $\mathcal{B}^-$. We obtain two distant negative passages per question: one hard negative, a top prediction from REALM without finetuning, and one random negative, drawn from a uniform distribution, both not containing the answer.

Full negatives considers all passages in Wikipedia expect p^+ as $\mathcal{B}^-$, and instead freezes the passage encoder f_p and only finetunes the question encoder f_q. This is appealing because (a) the number and the quality of the negatives, which both are the significant factors for training, are the strict maximum, and (b) f_p from REALM is already good, producing high quality passage representations without finetuning. Implementation of this method is feasible by exploiting extensive model parallelism.

We use distant negatives for multi-answer datasets and full negatives for NQ as this combination gave the best result.

B Experiment Details

B.1 Data processing for TREC

TREC from Baudiš and Šedivỳ (2015) contains regular expressions as the answers. We approximate a set of semantically distinct answers as follows. We first run regular expressions over Wikipedia to detect valid answer text. If there is no valid answer found from Wikipedia, or there are more than 100 valid answers[8], we discard the question. We then only keep the answers with up to five tokens, following the notion of short answers from Lee et al. (2019). Finally, we group the answers that are the same after normalization and white space removal. We find that this gives a reasonable approximation of a set of semantically distinct answers. Note that the data we use is the subset of the original data because we discarded a few questions. Statistics are reported in Section 4.1.

Here is an example: a regular expression from the original data is `Long Island|New\s?York|Roosevelt Field`. All matching answers over Wikipedia include `roosevelt field`, `new york`, `new\xa0york`, `new\nyork`, `newyork`, `long island`. Once the grouping is done, we have three semantically distinct answers: (1) `roosevelt field`, (2) `new york|new\xa0york|new\nyork|newyork`, and (3) `long island`.

[8] In most of such cases, the regular expressions are extremely permissive.

	k	B	# train steps	γ	β
WEBQSP	5	256	10k	1.5	3.0
	10	224		1.5	1.5
AMBIGQA	5	256	6k	1.0	2.5
	10	224		1.0	2.0
TREC	5	64	3k	1.5	1.5
	10	56		1.5	2.0

Table 9: Full hyperparamters for training JPR.

B.2 Details of reranker training

All implementations are based on Tensorflow (Abadi et al., 2015) and Mesh Tensorflow (Shazeer et al., 2018). All experiments are done in Google Cloud TPU. We use batch size that is the maximum that fits one instance of TPU v3-32 (for WEBQSP and AMBIGQA) or TPU v3-8 (TREC). We use the same batch size for INDEPPR; for Nogueira et al. (2020), we use the batch size of 1024. We use the encoder length of 360 and the decoder length of k (JPR) or 1 (all others). We use $k = \{5, 10\}$ for all experiments. We train JPR with $\gamma = \{0, 0.5, 1.0, 1.5\}$ and choose the one with the best accuracy on the development data. We use a flat learning rate of 1×10^{-3} with warm-up for the first 500 steps. Full hyperparameters are reported in Table 9.

For training INDEPPR and JPR, instead of using all of $|\mathcal{B}|$ passages, we use $|\mathcal{B}|/4$ passages by sampling k positive passages and $|\mathcal{B}|/4 - k$ negative passages. We find that this trick allows larger batch size when using the same hardware, ultimately leading to substantial performance gains. We also find

k	Models	WEBQSP		AMBIGQA	TREC	
		Dev	Test	Dev	Dev	Test
5	INDEPPR	62.4/59.0	65.1/60.9	73.6/69.5	**70.7**/61.1	**74.9**/66.4
	JPR	**69.5/67.9**	**69.1/65.8**	**73.7/70.0**	69.8/**61.4**	74.7/**66.8**
10	INDEPPR	60.1/57.2	61.0/57.4	73.6/**69.5**	66.4/60.3	68.9/61.5
	JPR	**70.3/67.2**	**68.9/65.4**	**73.7**/69.4	**70.1/62.6**	**74.3/66.2**

Table 10: Results on passage retrieval in α-NDCG.

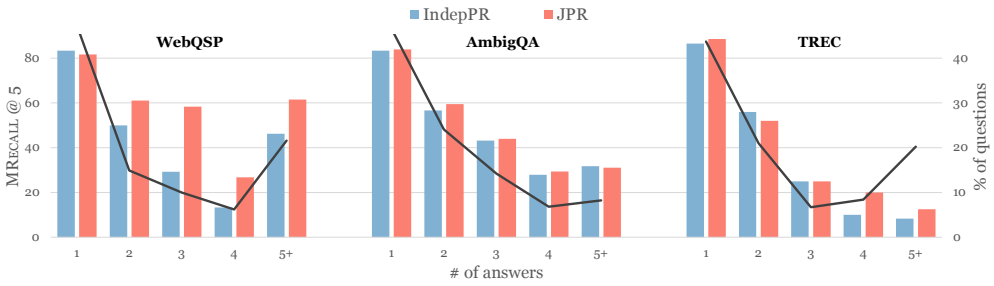

Figure 4: INDEPPR vs. JPR on the development data of three datasets. MRECALL @ 5 is reported. Lines indicate % of questions in the data. **JPR benefits more on questions with 2+ distinct answers.**

Algorithm 2 An algorithm to obtain $\tilde{\mathcal{O}}$ from the answer set and $\mathcal{B}$.

```
procedure PREPROC(k, {a_1...a_n}, B)
    Õ ← // a set of positive passages
    A_left ← {a_1...a_n}
    for b in B do
        if b covers any of A_left then
            Õ ← Õ.add(b)
            A_left ← A_left − answers in b
        if |Õ| == k then
            break
    return O.toSet()
```

that assigning indexes of the passages based on a prior, e.g., ranking from dense retrieval, leads to significant bias, e.g., in 50% of the cases, the top-1 passage from dense retrieval contains a correct answer. We therefore randomly assign the indexes, and find this gives significantly better performance.

Algorithm 2 describes how a set of positive passages $\tilde{\mathcal{O}}$ used in Section 3.2 is computed during preprocessing.

B.3 Details of answer generation training

We train the models using a batch size of 32. We use a decoder length of 20 and 40 for NQ and multi-answer datasets, respectively. We decode answers only when they appear in the retrieved passages, as we want the generated answers to be grounded by Wikipedia passages. Answers in the output sequence follow the order they appear in the passages, except on WEBQSP, where shuffling the order of the answers improves the accuracy. All other training details are the same as details of reranker training.

C Additional Results

We additionally report retrieval performance in **α-NDCG @ k**, one of the metrics for diverse retrieval in IR (Clarke et al., 2008; Sakai and Zeng, 2019). It is a variant of NDCG (Järvelin and Kekäläinen, 2002), but penalizes retrieval of the same answer. We refer to Clarke et al. (2008) for a complete definition. We use $\alpha = 0.9$.

Results are reported in Table 10. JPR consistently outperforms INDEPPR across all datasets, although the gains are less significant than the gains in MRECALL. We note that we report α-NDCG following IR literatures, but we think of MRECALL as a priority, because α-NDCG does not use an explicit notion of *completeness* of retrieval of all answers. It is also a less strict measure than recall because it gives partial credits to retrieving a subset of the answers.

Gains with respect to the number of answers Figure 4 shows gains over INDEPPR on three datasets with respect to the number of answers. Overall, gains are larger when the number of answers is larger, especially for WEBQSP and TREC. For AMBIGQA, the largest gains are when the number of answers is 2, which is responsible for over half of multi-answer questions.