End-to-End Training of Multi-Document Reader and Retriever for Open-Domain Question Answering

Devendra Singh Sachan[1,2]**, Siva Reddy**[1,2]**, William Hamilton**[1,2]**, Chris Dyer**[3]**, Dani Yogatama**[3]

[1]Mila - Quebec AI Institute
[2]School of Computer Science, McGill University
[3]DeepMind
sachande@mila.quebec, {siva, wlh}@cs.mcgill.ca
{cdyer, dyogatama}@deepmind.com

Abstract

We present an end-to-end differentiable training method for retrieval-augmented open-domain question answering systems that combine information from multiple retrieved documents when generating answers. We model retrieval decisions as latent variables over sets of relevant documents. Since marginalizing over sets of retrieved documents is computationally hard, we approximate this using an expectation-maximization algorithm. We iteratively estimate the value of our latent variable (the set of relevant documents for a given question) and then use this estimate to update the retriever and reader parameters. We hypothesize that such end-to-end training allows training signals to flow to the reader and then to the retriever better than stage-wise training. This results in a retriever that is able to select more relevant documents for a question and a reader that is trained on more accurate documents to generate an answer. Experiments on three benchmark datasets demonstrate that our proposed method outperforms all existing approaches of comparable size by 2-3 absolute exact match points, achieving new state-of-the-art results. Our results also demonstrate the feasibility of learning to retrieve to improve answer generation without explicit supervision of retrieval decisions.

1 Introduction

Open-domain question answering (OpenQA) is a question answering task where the goal is to train a language model to produce an answer for a given question. In contrast to many question answering tasks, an OpenQA model is only provided with the question as its input without accompanying documents that contain the answer. One of the most promising approaches to OpenQA is based on augmenting the language model with an external knowledge source such as Wikipedia (often referred to as the evidence documents). In this approach, the model consists of two core components (Chen *et al.*, 2017): (i) an information retrieval system to identify useful pieces of text from the knowledge source (the retriever); and (ii) a system to produce the answer given the retrieved documents and the question (the reader).

We can view such a model as a latent variable model, where the latent variables represent retrieved documents that are used to produce answers given questions (Lee *et al.*, 2019). End-to-end (joint) training of this model is challenging since we need to learn both to generate an answer given retrieved documents and what to retrieve. Previous work considers two potential solutions (see Table 1 for a high-level summary). First, they adopt a stage-wise training, where the retriever is trained while freezing the reader and vice versa (Karpukhin *et al.*, 2020, Izacard and Grave, 2021b,a). Another

		Reader and Retriever Training				
Model	*Multi-Doc Reader*	*Retriever Adaptation*	*Disjoint*	*End-to-End*	*Multi-Step*	*Unsupervised Retriever*
REALM (Guu *et al.*, 2020)		✓		✓		✓
DPR (Karpukhin *et al.*, 2020)			✓			
RAG (Lewis *et al.*, 2020b)		✓		✓		
FiD (Izacard and Grave, 2021b)	✓		✓			
FiD-KD (Izacard and Grave, 2021a)	✓	✓			✓	
EMDR^2 (Our Approach)	✓	✓		✓		✓

Table 1: Bird's-eye view of the recent OpenQA approaches. **Multi-Doc reader** indicates whether the reader architecture uses multiple documents or a single document. **Retriever adaptation** shows whether the retriever gets feedback from the reader to update its parameters. **Disjoint** denotes that first the retriever is trained and then the reader is trained. **End-to-end** denotes that the reader and retriever are trained jointly in one cycle. **Multi-step** indicates that the reader and retriever are trained iteratively in multiple cycles. **Unsupervised retriever** indicates whether the retriever is initialized using unsupervised approaches or using supervised data.

alternative is to constraint the reader to condition on each retrieved document individually[1] (Guu *et al.*, 2020)—sometimes with extra supervision for the latent variables in the form of the relevant document for a question (Lewis *et al.*, 2020b).

In this paper, we consider a retrieval-augmented question answering model that combines information from multiple documents when generating answers. Expectation-maximization (Dempster *et al.*, 1977) offers a principled template for learning this class of latent variable models. We present EMDR^2: **E**nd-to-end training of **M**ulti-**D**ocument **R**eader and **R**etriever (§2). EMDR^2 iteratively uses feedback from the model itself as "pseudo labels" of the latent variables for optimizing the retriever and reader parameters. We use two estimates of the latent variables: (i) prior scores for updating the reader parameters and (ii) approximate posterior scores given all observed variables for the retriever parameters.

We evaluate our proposed method by experimenting on three commonly used OpenQA datasets: Natural Questions, TriviaQA, and WebQuestions (§3). EMDR^2 achieves new state-of-the-art results for models of comparable size on all datasets, outperforming recent approaches by 2-3 absolute exact match points. We also show that EMDR^2 is robust to retriever initialization. It achieves high accuracy with unsupervised initialization, suggesting that supervised training of the retriever may not be an essential component of the training process as suggested in prior work (Karpukhin *et al.*, 2020).

In summary, our contributions are as follows: (i) we present an end-to-end training method (EMDR^2) for retrieval-augmented question-answering systems; (ii) we demonstrate that EMDR^2 outperforms other existing approaches of comparable size without any kind of supervision on the latent variables; (iii) we provide ablation studies for a better understanding of the contributions of different components of our proposed method; and (iv) we release our code and checkpoints to facilitate future work and for reproducibility.[2]

EMDR^2 is a framework that can be used to train retrieval-augmented text generation models for any task. We believe that our estimation technique in EMDR^2 is also useful for learning similar latent variable models in other domains.

2 Model

Our proposed model EMDR^2 consists of two components: (i) a neural retriever and (ii) a neural reader, which we train jointly in an end-to-end setting. Figure 1 shows an illustration of our model and training procedure. We discuss each component and our training objective in detail below.

---

[1]This makes marginalization over the latent variables easier since we only need to consider one document at a time rather than multiple documents at once.

[2]Our code is available at: `https://github.com/DevSinghSachan/emdr2`

2.1 Neural Retriever: Dual Encoder

Let the collection of evidence documents be denoted by $\mathcal{D} = \{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_M\}$. Given a question $\boldsymbol{q}$, the goal of the retriever module is to select a subset of documents $\mathcal{Z} \subset \mathcal{D}$ to answer the question. We model the retriever as a dual-encoder network (Bromley *et al.*, 1994), where one encoder f_q encodes the question and another f_d encodes the evidence document (to a vector). The retrieval score is defined as the dot product between the two resulting vectors:

$$\text{score}(\boldsymbol{q}, \boldsymbol{d}_i; \Phi) = f_q(\boldsymbol{q}; \Phi_q)^\top f_d(\boldsymbol{d}_i; \Phi_d), \tag{1}$$

where $\Phi = [\Phi_q, \Phi_d]$ denotes the retriever parameters. We select top-K documents for the question $\boldsymbol{q}$ from $\mathcal{D}$ based on the retrieval scores. We denote the set of retrieved documents by $\mathcal{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_K\}$.

We use transformer encoders (Vaswani *et al.*, 2017) as our f_q and f_d. Our transformer architecture is similar to BERT with 12 layers and 768 hidden size (Devlin *et al.*, 2019). We use the final representation of the first token (i.e., the standard `[CLS]` token from BERT's tokenization) as our question (and similarly document) embedding. Initializing f_q and f_d with BERT weights has been shown to lead to a poor retrieval accuracy (Lee *et al.*, 2019, Sachan *et al.*, 2021). Therefore, we initialize the retriever with an unsupervised training procedure. We discuss our initialization technique in detail in §3.2.

2.2 Neural Reader: Fusion-in-Decoder

The reader takes as input a question $\boldsymbol{q}$ and a set of retrieved documents (to be read) $\mathcal{Z}$ to generate an answer. Our reader is based on the Fusion-in-Decoder (FiD; Izacard and Grave, 2021b) model, which is built on top of T5 (Raffel *et al.*, 2020). T5 is a pretrained sequence-to-sequence transformer that consists of an encoder g_e and a decoder g_d.

In FiD, each retrieved document $\boldsymbol{z}_k$ is first appended with its title ($\boldsymbol{t}_{\boldsymbol{z}_k}$) and the question:

$$\boldsymbol{x}_k = \texttt{[CLS]}\,\boldsymbol{q}\,\texttt{[SEP]}\,\boldsymbol{t}_{\boldsymbol{z}_k}\,\texttt{[SEP]}\,\boldsymbol{z}_k\,\texttt{[SEP]},$$

where `[CLS]` is used to indicate the start of a document and `[SEP]` is used as a separator for the different parts of the document as well as the final token.

Each $\boldsymbol{x}_k$ is then independently given as an input to the T5 encoder g_e. The output representations corresponding to all of the retrieved documents are concatenated as:

$$\mathbf{X}_{\mathcal{Z}} = [g_e(\boldsymbol{x}_1); \ldots; g_e(\boldsymbol{x}_K)] \in \mathbb{R}^{(N \times K) \times H},$$

where N is the number of tokens in each $\boldsymbol{x}_k$[3] and H is the hidden size of the T5 encoder g_e. In this work, we use the T5-*base* configuration with $N = 512$ and $H = 768$.

$\mathbf{X}_{\mathcal{Z}}$ is then given as an input to the T5 decoder g_d. When generating an answer token, the decoder attends to both previously generated tokens (i.e., causal attention) as well as the tokens encoded in $\mathbf{X}_{\mathcal{Z}}$ (i.e., cross attention). Since $\mathbf{X}_{\mathcal{Z}}$ contains information from multiple documents, the decoder has the ability to aggregate useful signals contained in multiple documents and jointly reason over them. We define the probability of the answer as:

$$p(\boldsymbol{a} \mid \boldsymbol{q}, \mathcal{Z}; \Theta) = \prod_{t=1}^{T} p\left(a_t \mid \boldsymbol{a}_{<t}, \boldsymbol{q}, \mathcal{Z}; \Theta\right), \tag{2}$$

where Θ denotes the reader parameters (i.e., T5 encoder and decoder) and T is the number of answer tokens. We keep generating answer tokens until the decoder outputs a special `EOS` token or a pre-specified maximum answer length is reached.

2.3 End-to-End Training of Reader and Retriever

In contrast to previous work on generative question answering, we train both the reader and the retriever jointly in an end-to-end differentiable fashion.

Denote our latent variable which represents a set of retrieved documents by Z and let $\mathcal{Z}$ be a possible value of Z. The marginal likelihood of an answer (marginalizing over all the possible values of Z)

---

[3] We truncate and pad as necessary such that every $\boldsymbol{x}_k$ has the same length N. See §3.2 for details.
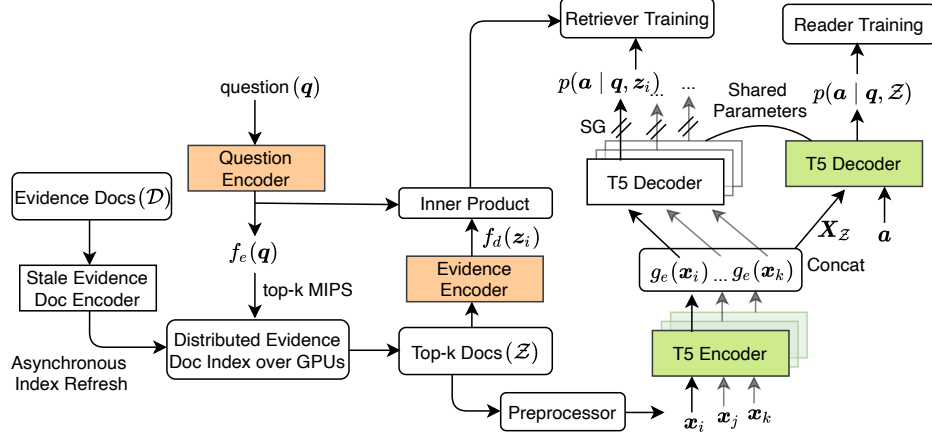
Figure 1: An illustration of the different components of EMDR². Colored blocks indicate components which contain trainable parameters.

is: $p(\boldsymbol{a} \mid \boldsymbol{q}; \Theta, \Phi) = \sum_{Z=\mathcal{Z}} p(\boldsymbol{a} \mid \boldsymbol{q}, \mathcal{Z}; \Theta)p(\mathcal{Z} \mid \boldsymbol{q}; \Phi)$. The goal of our training procedure is to find Φ and Θ that would maximize the above objective. Exactly optimizing Eq. 3 is intractable as it is combinatorial in nature.[4] For one particular value $\mathcal{Z}$, the log-likelihood is simpler to compute: $\log p(\boldsymbol{a} \mid \boldsymbol{q}, \mathcal{Z}; \Theta)p(\mathcal{Z} \mid \boldsymbol{q}; \Phi) = \log p(\boldsymbol{a} \mid \boldsymbol{q}, \mathcal{Z}; \Theta) + \log p(\mathcal{Z} \mid \boldsymbol{q}; \Phi)$.

Expectation-maximization (EM) algorithm (Dempster *et al.*, 1977) offers a solution to learning this latent variable model. In classical EM, we iteratively compute the posterior of Z given all observed variables and use it to update Θ and Φ.

We propose using two estimates of Z—$\mathcal{Z}_{\text{reader}}$ and $\mathcal{Z}_{\text{retriever}}$—for updating the two components of the model (reader parameters Θ and retriever parameters Φ):

$$\log \underbrace{p(\boldsymbol{a} \mid \boldsymbol{q}, \mathcal{Z}_{\text{reader}}; \Theta)}_{\text{reader}} + \log \underbrace{p(\mathcal{Z}_{\text{retriever}} \mid \boldsymbol{q}; \Phi)}_{\text{retriever}}. \tag{3}$$

In the first term, we set the value of the latent variable $Z = \mathcal{Z}_{\text{reader}}$ based on the prior scores. In the second term, we seek to maximize an approximate posterior of $Z = \mathcal{Z}_{\text{retriever}}$. We discuss them in more detail below.

Reader parameters Θ. For updating Θ (the first term of Eq. 3), we use the top-K documents with the highest individual scores (as computed by Eq. 1 based on the current value of Φ) to construct $\mathcal{Z}_{\text{reader}}$. This is equivalent to relying on the prior $p(Z \mid \boldsymbol{q}; \Phi)$ to estimate $\mathcal{Z}_{\text{reader}}$ (without using information from the answer $\boldsymbol{a}$). We choose to use the prior to train reader parameters since the prior scores are also used at evaluation time to obtain the top-K documents. As a result, there is no mismatch between training and test computations when computing $p(\boldsymbol{a} \mid \boldsymbol{q}, \mathcal{Z}; \Theta)$ (i.e., $\mathcal{Z}$ that is used at test time is obtained in exactly the same way as $\mathcal{Z}_{\text{reader}} = \mathcal{Z}_{\text{top-}K}$).

Retriever parameters Φ. For updating Φ (the second term of Eq. 3), we propose to use the posterior estimate. In other words, we use additional information from $\boldsymbol{a}$ when evaluating $\mathcal{Z}_{\text{retriever}}$ to train Φ. Using the posterior allows our retriever to learn from richer training signals as opposed to relying only on the prior.

We need to be able to compute $p(\mathcal{Z}_{\text{retriever}} \mid \boldsymbol{q}, \boldsymbol{a}; \Theta, \Phi)$ to maximize the retriever parameters. However, computing this quantity is difficult since it is a probability of a set.[5] Consider a set of K documents (e.g., $\mathcal{Z}_{\text{top-}K}$), where $\boldsymbol{z}_k$ denotes a document in the set. We approximate the maximization of the probability of the set by assuming that its probability is maximized if the sum of the probability of

---

[4]Contrast our objective with REALM (Guu *et al.*, 2020), where the reader only conditions on one retrieved document $\boldsymbol{z}_k$ when generating an answer. In this case, the latent variable represents a document assignment instead of a set of retrieved documents.

[5]This is true whether we choose to use the posterior probability or the prior probability.

each document in the set is maximized.[6] With this approximation, we arrive at a simpler quantity: $\sum_{k=1}^{K} p(z_k \mid q, a; \Theta, \Phi)$. Note that using Bayes rule, we can rewrite:[7]

$$p(z_k \mid q, a; \Theta, \Phi) \propto p(a \mid q, z_k; \Theta) p(z_k \mid q; \Phi). \tag{4}$$

The reader now only conditions on one document when computing the probability of an answer $p(a \mid q, z_k; \Theta)$. This simpler reader uses the same parameters as the more sophisticated one Θ, but it only uses one document z_k instead of a set of documents.

To compute Eq. 4, we first obtain K documents with the highest scores as computed by Eq. 1 based on the current value of Φ. We compute the probability of document $z_k \in \mathcal{Z}_{\text{top-}K}$ as:

$$p(z_k \mid q, \mathcal{Z}_{\text{top-}K}; \Phi) \approx \frac{\exp(\text{score}(q, z_k)/\tau; \Phi)}{\sum_{j=1}^{K} \exp(\text{score}(q, z_j)/\tau; \Phi)}, \tag{5}$$

where τ is a temperature hyperparameter and the approximation assumes that documents beyond the top-K contributes very small scores so we do not need to sum over all evidence documents M in the denominator (which is in the order of tens of millions in our experiments). We then compute $p(a \mid q, z_k; \Theta)$ similarly to Eq. 2.

Overall training objective of EMDR^2. Combining the above derivations, our end-to-end training objective that we seek to maximize for a particular example becomes:

$$\mathcal{L} = \underbrace{\log p(a \mid q, \mathcal{Z}_{\text{top-}K}; \Theta)}_{\text{reader}} + \underbrace{\log \sum_{k=1}^{K} \mathbb{SG}\left(p(a \mid q, z_k; \Theta)\right) p(z_k \mid q, \mathcal{Z}_{\text{top-}K}; \Phi)}_{\text{retriever}}, \tag{6}$$

where $\mathbb{SG}$ is the stop-gradient operator so that the reader parameters Θ are not updated to also perform well given a single document z_k. The stop-gradient operator in the second term of EMDR^2 has several benefits. First, the FiD reader is trained from the first term of the EMDR^2 objective in which its likelihood is conditioned on all the retrieved documents, similar to how the reader is used at test time. Second, it also makes training faster since the backward pass which is computationally more expensive than the forward pass is not needed, which in turn reduces the usage of GPU RAM as intermediate activations need not be saved.

Given a training example, we update Θ and Φ by taking gradients of Eq. 6 with respect to Θ and Φ in an end-to-end fashion. Intuitively, we train the reader to generate the correct answer given K highest scoring documents $\mathcal{Z}_{\text{top-}K}$. For the retriever, we train it to select K documents which *collectively* has a high score of generating an answer (since the sum over K is inside the log in the second term) while taking into account feedback from the reader. Algorithm 1 summarizes our training algorithm.

Algorithm 1: End-to-end training of multi-document reader and retriever.

Input: Model parameters Θ and Φ, evidence documents $\mathcal{D}$.
while *not converged* **do**
- Compute $\mathcal{Z}_{\text{top-}K}$ using the current retriever parameters Φ.   // E-step
- Compute $p(a \mid q, z_k)$ for each z_k using the current reader parameters Θ.   // E-step
- Update model parameters Θ and Φ to maximize the log-likelihood in Eq. 6. // M-step

end

---

3 Experiments

3.1 Datasets

We experiment with three commonly used open-domain question answering datasets:

---

[6]The intuition is that each element of the set contributes independently, which greatly simplifies the computation to find the maximum of the set.

[7]We choose not to normalize with $p(a \mid q; \Theta, \Phi)$ since computing this quantity would require summing over all evidence documents M. While this makes the resulting objective that we optimize not correspond to a proper probability distribution anymore, we observe that our training method still behaves well in practice.

- **Natural Questions (NQ; Kwiatkowski *et al.*, 2019).** NQ contains questions asked by users of the Google search engine. Similar to Lee *et al.* (2019), we use the short answer subset.
- **TriviaQA (Joshi *et al.*, 2017).** TriviaQA is a collection of trivia question-answer pairs that were collected from multiple sources on the web.
- **WebQuestions (WebQ; Berant *et al.*, 2013).** WebQ questions were collected using Google Suggest API and the answers were annotated using Mechanical Turk. We use the version from Chen *et al.* (2017) where Freebase IDs in the answers are replaced by entity names.

Evidence documents $\mathcal{D}$**.** We use the preprocessed English Wikipedia dump from December 2018 released by Karpukhin *et al.* (2020) as our evidence documents. Each Wikipedia article is split into non-overlapping 100 words long segments. Each segment corresponds to a document in our case. There are a total of 21,015,324 documents in total.

We provide descriptive statistics and other preprocessing details in Appendix A.

3.2 Implementation Details

Hardware and library. We run all of our experiments on a machine with 96 CPUs, 1.3TB physical memory, and 16 A100 GPUs. We use PyTorch (Paszke *et al.*, 2019) to implement our proposed model and relevant baselines.

Model configurations. For both the retriever and reader, we use the *base* configuration that consists of 12 layers, 768 dimensional hidden size, and 12 attention heads. In all experiments, we retrieve 50 documents, unless stated otherwise. We only use the base configuration in our experiments due to GPU memory constraints. However, we believe that our results would generalize to larger configurations as well.

Retrieval. To support fast retrieval, we pre-compute evidence document embeddings and store them in a distributed fashion over all the GPUs. We refer to these document embeddings as the document index. For each question, we retrieve documents in an online (on-the-fly) manner by performing exact maximum inner product search (MIPS), implemented using asynchronous distributed matrix multiplication over the document index. These documents are converted to subwords using BERT's tokenization and are given as input to the T5 reader. If a tokenized document is shorter than 512 tokens, it is padded using the tokens from the neighboring documents until the maximum token limit is reached. Such padding additionally helps to provide an extended context for answer generation.

Initialization and training details. We initialize the parameters of the model with unsupervised pre-training before performing supervised training using the question-answer training examples. Unsupervised pre-training is essential as it helps to warm-start the retriever so that it outputs relevant documents for a given question.

We first pre-train the retriever parameters with unsupervised Inverse Cloze Task training (Lee *et al.*, 2019) for 100,000 steps. We then extract sentences containing named entities from the evidence documents. Next, we replace 15% of the named entity tokens with masked tokens, which are often referred to as masked salient spans (MSS; Guu *et al.*, 2020). The masked sentence can be considered as the question and its salient spans (i.e, named entities) can be considered as the answer to train the model with Eq. 6. We train the model on these question-answer (masked sentence-named entities) pairs for 82,000 steps with a batch size of 64 using Adam (Kingma and Ba, 2015). We refer to this initialization method as *unsupervised pre-training with masked salient spans*. We provide further description in Appendix C.

After MSS training, we finetune the model on the dataset-specific question-answer training examples with EMDR^2. We perform training for 10 epochs on NQ and TriviaQA with a batch size of 64, and for 20 epochs on WebQ with a batch size of 16. During training, we save a checkpoint every 500 steps and select the best checkpoint based on its performance on the development set.

During end-to-end training, since the parameters of the document encoder (f_d) are also updated at every step, the pre-computed document embeddings become stale as training progresses. We use the most recent document encoder checkpoint to compute fresh document embeddings asynchronously with which the document index is updated after every 500 training steps to prevent staleness.

Model	top-K	NQ		TriviaQA		WebQ		# of
		dev	test	dev	test	dev	test	*params*
Closed-Book QA Models								
T5-*base* (Roberts *et al.*, 2020)	0	-	25.7	-	24.2	-	28.2	220M
T5-*large* (Roberts *et al.*, 2020)	0	-	27.3	-	28.5	-	29.5	770M
T5-*XXL* (Roberts *et al.*, 2020)	0	-	32.8	-	42.9	-	35.6	11B
GPT-3 (Brown *et al.*, 2020)	0	-	29.9	-	-	-	41.5	175B
Open-Book QA Models								
BM25 + BERT (Lee *et al.*, 2019)	5	24.8	26.5	47.2	47.1	27.1	21.3	220M
ORQA (Lee *et al.*, 2019)	5	31.3	33.3	45.1	45.0	36.8	30.1	330M
REALM (Guu *et al.*, 2020)	5	38.2	40.4	-	-	-	40.7	330M
DPR (Karpukhin *et al.*, 2020)	25	-	41.5	-	56.8	-	34.6	330M
RECONSIDER (Iyer *et al.*, 2021)†	30	-	43.1	-	59.3	-	44.4	440M
RAG-Sequence (Lewis *et al.*, 2020b)†	50	44.0	44.5	55.8	56.8	44.9	45.2	626M
Individual Top-K (Sachan *et al.*, 2021)	-	-	45.9	-	56.3	-	-	440M
Joint Top-K (Sachan *et al.*, 2021)	50	-	49.2	-	64.8	-	-	440M
FiD (Izacard and Grave, 2021b)	100	-	48.2	-	65.0	-	-	440M
FiD-KD (Izacard and Grave, 2021a)	100	48.0	49.6	68.6	68.8	-	-	440M
Our Implementation (Base Configuration)								
FiD / T5-*base*	0	26.0	25.1	26.7	27.8	31.0	32.4	220M
FiD (DPR retriever, T5 reader)	1	37.3	38.4	50.8	50.4	40.2	38.3	440M
FiD (DPR retriever, T5 reader)	50	47.3	48.3	65.5	66.3	46.0	45.2	440M
FiD (MSS + DPR retriever, T5 reader)	50	48.8	50.4	68.0	68.8	43.5	46.8	440M
FiD (MSS retriever, MSS reader)	50	38.5	40.1	60.0	59.8	39.1	40.2	440M
EMDR^2 (MSS retriever, MSS reader)	50	**50.4**	**52.5**	**71.1**	**71.4**	**49.9**	**48.7**	440M

Table 2: Exact match scores on three evaluation datasets. Top-K denotes the number of retrieved documents that are used by the reader to produce an answer. To provide a fair comparison with our reimplementations, we show results from other papers with the base configuration, except for RAG-Sequence that uses BART-*large* (Lewis *et al.*, 2020a). † indicates that their results on WebQ use NQ training data to pretrain the model.

Inference. We use greedy decoding for answer generation at inference time.

3.3 Baselines

We compare our model to other approaches for OpenQA that can be categorized under the following two classes:

- **Closed-book QA models.** Large-scale language models capture a lot of world knowledge in their parameters derived from the corpus they have been trained on (Petroni *et al.*, 2019). We compare with the work of Roberts *et al.* (2020) who show that larger T5 models—when finetuned with question-answer pairs—can perform remarkably well. We also compare with the few-shot results of GPT-3 (Brown *et al.*, 2020).[8]

- **Open-book QA models.** Similar to this work, these models consist of retriever and reader components and adopt the retrieve then predict approach for answering questions given a collection of evidence documents. These models mainly differ in how the retriever is initialized (ORQA; Lee *et al.*, 2019, DPR; Karpukhin *et al.*, 2020), whether the reader processes a single document (ORQA, DPR, RAG; Lewis *et al.*, 2020b) or multiple documents (FiD; Izacard and Grave, 2021b), or whether the reader and retriever are trained jointly or in a multistage process (REALM; Guu *et al.*, 2020, FiD-KD; Izacard and Grave, 2021a).

3.4 Results

We follow standard conventions and report exact match (EM) scores using the reference answers included in each dataset. Table 2 shows our main results. We divide the table into three main sections: closed-book QA models, open-book QA models, and our implementation. The first two sections contain results from other papers, which we include for comparisons. The last section includes results from our proposed model, as well as our reimplementation of relevant baselines to control for our experimental setup.

Our reimplementation of T5-base provides strong baselines when the number of retrieved documents is set to 0 (no retrieval) and 1. From Table 2, we see that the setting of top-1 vastly improves performance over the setting with no retrieved documents, signifying the importance of retrieval for OpenQA tasks. When further increasing the top-k documents to 50, the performance of the FiD models substantially improves over the top-1 retrieval, verifying the observation from (Izacard and Grave, 2021b) about the *importance of modeling the retrieved documents as a set*.

Comparing EMDR^2 with our reimplementation of FiD illustrates the benefit of our end-to-end training approach. The underlying model is similar in both cases, but the training method is different. FiD adopts a two-stage approach to first train the retriever and then the reader. We have three variants of FiD: (i) the reader and retriever are initialized with MSS training, (ii) the retriever is initialized with DPR training, which is the setting used in the original paper (Izacard and Grave, 2021b), and (iii) the retriever is initialized with MSS + DPR training from (Sachan *et al.*, 2021), as it further improves DPR recall. EMDR^2 outperforms all the variants by large margins on all the datasets.

The current best approach for training multi-document reader and retriever is FiD-KD (Izacard and Grave, 2021a). FiD-KD is a complex training procedure that requires multiple training stages and performs knowledge distillation with inter-attention scores. We take the results from the original paper when comparing our model with FiD-KD. EMDR^2 outperforms the reported numbers of FiD-KD by more than 2.5 points on NQ and TriviaQA to obtain new state-of-the-art results on these benchmarks.

In addition to better performance, EMDR^2 also has three other advantages compared to FiD-KD: (i) EMDR^2 is more efficient since it only uses 50 evidence documents, whereas FiD-KD leverages 100 documents; (ii) FiD-KD is based on a distillation approach which requires multiple cycles of retriever and reader training, while EMDR^2 only requires one cycle of end-to-end training; and (iii) FiD-KD relies on supervised initialization of the retriever to achieve its best performance. EMDR^2 is more robust to the retriever initialization, as demonstrated by state-of-the-art results even with unsupervised initialization of the retriever.

For the WebQ dataset, the training set size is much smaller compared to the other datasets (Table 5). Previous approaches such as RAG rely on supervised transfer (i.e., they finetune a model pre-trained on NQ) to obtain good results. In contrast, EMDR^2 improves over the results from this RAG model by 3.5 points *without the supervised transfer step*. This result demonstrates the applicability of our approach to the low-resource setting where we only have a limited number of training examples.

We also perform qualitative analysis of the model outputs, which is included in Appendix E.

3.5 Ablations

Number of retrieved documents. We investigate the performance of EMDR^2 and FiD as we vary the number of retrieved documents K in Figure 2. We observe that when the number of retrieved documents is increased, both EMDR^2 and FiD improve in performance. When K is small, the gap between EMDR^2 and FiD is larger. This indicates the efficacy of EMDR^2 in a more constrained setting where we can only retrieve a small number of documents (e.g., due to memory limitations).

Retriever initialization. We explore the effect of different parameter initialization strategies when training with EMDR^2: (i) unsupervised MSS pre-training, (ii) supervised retriever training (DPR), and (iii) MSS pre-training followed by supervised retriever training (MSS + DPR; Sachan *et al.* (2021)). Table 3 shows our results. We can see that on NQ, MSS pre-training being unsupervised leads to a lower initial retriever recall than DPR. After EMDR^2 training, the recall improves by 20% (highlighted in yellow cells). Training with DPR initialization leads to the same final recall as obtained by MSS

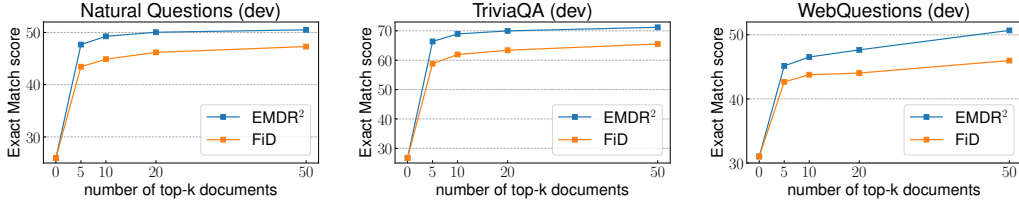[8] We note that GPT-3 is not trained on the full training examples that we use, so the results are not directly comparable.

Figure 2: Performance on NQ, TriviaQA, and WebQ as we vary the number of retrieved documents.

		NQ (dev)			TriviaQA (dev)			WebQ (dev)		
Retriever	**Reader**	**R@50**		**EM**	**R@50**		**EM**	**R@50**		**EM**
Initialization	**Initialization**	B.T.	A.T.		B.T.	A.T.		B.T.	A.T.	
MSS pre-training	MSS pre-training	66.4	86.3	50.4	74.8	86.2	71.1	59.8	88.6	49.9
MSS pre-training	T5	66.4	86.3	50.3	74.8	86.3	70.9	59.8	88.6	47.7
DPR training	T5	82.3	86.3	50.0	83.2	86.2	70.5	84.2	88.6	49.0
MSS + DPR	MSS pre-training	84.5	86.3	50.5	85.3	86.3	71.2	85.0	88.6	49.9

Table 3: R@50 denotes the retrieval recall from the top-50 retrieved documents. B.T. and A.T. indicates R@50 score Before Training and After Training the model, respectively.

pre-training, *suggesting that DPR initialization of the retriever may not be an essential component to obtain good performance in OpenQA tasks.* Similar trends are also observed on TriviaQA and WebQ. Similarly, MSS + DPR initialization has a better initial recall but leads to a marginal or no improvements in answer extraction performance over MSS pre-training. Finally, we also observe that MSS pre-training also provides an improvement of 2 points in answer extraction on WebQ when compared to the T5 reader (shown in orange cells), highlighting its importance in the low-resource OpenQA tasks.

3.6 Alternative End-to-End Training Objectives

We compare EMDR^2 objective (Eq. 6) to two alternative formulations for end-to-end training.

In the first alternative formulation, when training the retriever parameters Φ, we simply factorize $p(\mathcal{Z} \mid \boldsymbol{q}; \Phi) = \prod_{k=1}^{K} p(\boldsymbol{z}_k \mid \boldsymbol{q}; \Phi)$ to arrive at the following objective:

$$\mathcal{L}_{\text{alt-1}} = \log p(\boldsymbol{a} \mid \boldsymbol{q}, \mathcal{Z}; \Theta) + \sum_{k=1}^{K} \log p(\boldsymbol{z}_k \mid \boldsymbol{q}, \mathcal{Z}; \Phi).$$

Method	top-k	NQ	TriviaQA	WebQ
FiD	50	47.3	65.5	46.0
EMDR^2	50	**50.4**	**71.1**	**49.9**
$\mathcal{L}_{\text{alt-1}}$	50	14.1	11.9	28.0
$\mathcal{L}_{\text{alt-2}}$	50	49.9	69.6	28.8

Table 4: EM scores on the development set for alternative training objectives.

The second term in this objective is maximised by a uniform retrieval, in other words, by *removing* any discrimination between documents in the retriever. We include it to show the impact of an adversarial objective.

In the second formulation, for each retrieved document, we approximate its posterior under the assumption that we have a uniform prior over the set of retrieved documents: $\tilde{p}(\boldsymbol{z}_k \mid \boldsymbol{q}, \boldsymbol{a}, \mathcal{Z}_{\text{top-}K}; \Theta) \propto p(\boldsymbol{a} \mid \boldsymbol{q}, \boldsymbol{z}_k; \Theta) \times \frac{1}{K}$. We use this to train reader and retriever parameters as follows:

$$\mathcal{L}_{\text{alt-2}} = \log p(\boldsymbol{a} \mid \boldsymbol{q}, \mathcal{Z}; \Theta) + \mathbb{KL}(\mathbb{SG}\left(\tilde{p}(\boldsymbol{z}_k \mid \boldsymbol{q}, \boldsymbol{a}, \mathcal{Z}_{\text{top-}K}; \Theta)\right) \mid\mid p(\boldsymbol{z}_k \mid \boldsymbol{q}, \mathcal{Z}; \Phi)).$$

Intuitively, we try to match the probability of retrieving a document $\boldsymbol{z}_k$ with the "contribution" of that document to the generated answer $\boldsymbol{a}$, regardless of whether the retriever is relatively more or less likely to retrieve the document *a priori*.

Table 4 shows our results on the development set of NQ. We observe that training with the adversarial $\mathcal{L}_{\text{alt-1}}$ objective diverges, leading to poor performance, as expected. This shows that harming the retriever during training can significantly harm performance of the QA system. In contrast, although it disregards the estimated prior, the $\mathcal{L}_{\text{alt-2}}$ objective still improves over the FiD baseline for NQ and

TriviaQA. However, it still lags behind EMDR^2. On WebQ, the $\mathcal{L}_{\text{alt-2}}$ objective diverges and leads to a poor performance. We leave further analysis on the convergence of $\mathcal{L}_{\text{alt-2}}$ objective as a part of future work.

4 Related Work

Our work is based on end-to-end training of neural readers and retrievers, which we discuss in §1, §2, and §3. Here we instead focus on discussing previous work related to standalone neural retrievers, neural readers, and their application in other natural language processing tasks.

Neural retrievers. There are two broad classes of neural retrievers based on the number of embeddings computed for a document: dual encoders (Yih *et al.*, 2011, Lee *et al.*, 2019) and multivector encoders (Khattab and Zaharia, 2020, Luan *et al.*, 2021). Dual encoders store one embedding for each evidence document. Multivector encoders require multiple embeddings, which can be computationally expensive for large-scale retrieval. Due to the large size of the evidence document collection in OpenQA, our work uses the more efficient dual-encoder. Sachan *et al.* (2021) show that the performance of supervised dual encoders in OpenQA can be improved when pre-training with the Inverse Cloze Task for the high-resource setting or masked salient spans for the low-resource setting.

Neural readers. Neural readers output an answer given retrieved documents as its input. There are also two broad classes of neural readers: extractive and generative. Extractive readers (Clark and Gardner, 2018, de Masson d'Autume *et al.*, 2019, Wang *et al.*, 2019, Guu *et al.*, 2020, Karpukhin *et al.*, 2020) extract a span from a retrieved document to produce an answer. Generative readers (Izacard and Grave, 2021b), on the other hand, generates an answer conditioned on the retrieved documents.

Other application areas. In addition to question answering, retrieval-augmented methods have been successfully applied to other natural language processing tasks. In left-to-right language modeling, retrieving similar words from an external memory has been shown to improve perplexity (Khandelwal *et al.*, 2020, Yogatama *et al.*, 2021). In machine translation, retrieving domain-specific target language tokens has improved performance in domain adaptation (Khandelwal *et al.*, 2021). Finally, in dialog modeling, retrieving knowledge-informed text has helped improve factual correctness in the generated conversations (Fan *et al.*, 2021).

We provide a detailed comparison of EMDR^2 with some of the previous work in Appendix C and D.

5 Discussion

Summary of contributions. We presented EMDR^2, an end-to-end training method for retrieval-augmented question answering systems. We showed how to arrive at our training objective using the expectation-maximization algorithm. We demonstrated that EMDR^2 achieves state-of-the-art performance on three benchmark OpenQA datasets.

Technical limitations. EMDR^2 shares a few limitations with other retrieval-augmented question answering models. In particular, as evidence documents are stored in an uncompressed format, maintaining them and searching for relevant documents can be expensive (both in terms of compute and memory consumption). In our experiments, we only focused on open-domain question answering. It would be interesting to see how EMDR^2 performs for other text generation models as well. We also note that training is relatively resource-heavy (requiring 16 GPUs), potentially having environmental concerns.

Potential negative societal impacts. While EMDR^2 has the potential to improve language models in the low-resource setting (as demonstrated by our results on WebQ in §3.4), it could exhibit typical biases that are associated with large language models. For example, our model does not have an explicit mechanism to generate answers that are calibrated for fairness across all spectra. As a retrieval-augmented method, it also could be more prone to generating fake answers if an attacker manages to have access and modify information in the collection of evidence documents.

Acknowledgements

Funding Statement

References

Berant, J., Chou, A., Frostig, R., and Liang, P. (2013). Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1994). Signature verification using a "siamese" time delay neural network. In *Advances in Neural Information Processing Systems*.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems*.

Chen, D., Fisch, A., Weston, J., and Bordes, A. (2017). Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Clark, C. and Gardner, M. (2018). Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

de Masson d'Autume, C., Ruder, S., Kong, L., and Yogatama, D. (2019). Episodic memory in lifelong language learning. In *Advances in Neural Information Processing Systems*.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, **1**(39), 1–38.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Fan, A., Gardent, C., Braud, C., and Bordes, A. (2021). Augmenting Transformers with KNN-Based Composite Memory for Dialog. *Transactions of the Association for Computational Linguistics*, **9**.

Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M. (2020). Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*.

Iyer, S., Min, S., Mehdad, Y., and Yih, W. (2021). Reconsider: Re-ranking using span-focused cross-attention for open domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Izacard, G. and Grave, E. (2021a). Distilling knowledge from reader to retriever for question answering. In *International Conference on Learning Representations*.

Izacard, G. and Grave, E. (2021b). Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*.

Joshi, M., Choi, E., Weld, D., and Zettlemoyer, L. (2017). TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Karpukhin, V., Oğuz, B., Min, S., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. (2020). Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L., and Lewis, M. (2020). Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*.

Khandelwal, U., Fan, A., Jurafsky, D., Zettlemoyer, L., and Lewis, M. (2021). Nearest neighbor machine translation. In *International Conference on Learning Representations*.

Khattab, O. and Zaharia, M. (2020). Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *The 2015 International Conference for Learning Representations*.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Kelcey, M., Devlin, J., Lee, K., Toutanova, K. N., Jones, L., Chang, M.-W., Dai, A., Uszkoreit, J., Le, Q., and Petrov, S. (2019). Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

Lee, K., Chang, M.-W., and Toutanova, K. (2019). Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020a). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. (2020b). Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*.

Luan, Y., Eisenstein, J., Toutanova, K., and Collins, M. (2021). Sparse, Dense, and Attentional Representations for Text Retrieval. *Transactions of the Association for Computational Linguistics*, **9**.

Min, S., Chen, D., Hajishirzi, H., and Zettlemoyer, L. (2019). A discrete hard EM approach for weakly supervised question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China. Association for Computational Linguistics.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*.

Petroni, F., Rocktäschel, T., Riedel, S., Lewis, P., Bakhtin, A., Wu, Y., and Miller, A. (2019). Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Qi, P., Zhang, Y., Zhang, Y., Bolton, J., and Manning, C. D. (2020). Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, **21**(140), 1–67.

Roberts, A., Raffel, C., and Shazeer, N. (2020). How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*.

Sachan, D. S., Patwary, M., Shoeybi, M., Kant, N., Ping, W., Hamilton, W. L., and Catanzaro, B. (2021). End-to-end training of neural retrievers for open-domain question answering. In *Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B. (2019). Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*.

Wang, S., Yu, M., Guo, X., Wang, Z., Klinger, T., Zhang, W., Chang, S., Tesauro, G., Zhou, B., and Jiang, J. (2018). R3: Reinforced ranker-reader for open-domain question answering. In *AAAI*.

Wang, Z., Ng, P., Ma, X., Nallapati, R., and Xiang, B. (2019). Multi-passage BERT: A globally normalized BERT model for open-domain question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Yih, W.-t., Toutanova, K., Platt, J. C., and Meek, C. (2011). Learning discriminative projections for text similarity measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*.

Yogatama, D., de Masson d'Autume, C., and Kong, L. (2021). Adaptive Semiparametric Language Models. *Transactions of the Association for Computational Linguistics*, **9**, 362–373.

Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Please see the model (§2) and result (§3) sections that solidify the claims made in the abstract and introduction sections.

   (b) Did you describe the limitations of your work? [Yes] Please see limitations in §5.

   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Please see negative societal impact in §5.

   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...

   (a) Did you state the full set of assumptions of all theoretical results? [N/A]

   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We include the code, data, and instructions in the supplemental material and §3.2.

   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We specify these details in the appendix included in the supplementary material.

   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Our experiments are compute expensive and it is not feasible to perform multiple runs of the same experiment with different seeds. All our training runs use the same seed value (1234). As an alternative to running multiple seeds, we perform a number of ablation studies (§3.5).

   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please see §3.2 under hardware and library.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

   (a) If your work uses existing assets, did you cite the creators? [Yes] Please see §3.1 for the details.

   (b) Did you mention the license of the assets? [Yes] Our work is based on open-source data and framework. When applicable, we describe the license information in the appendix.

   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include our code in the supplementary material.

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Dataset	Train	Filtered Train	Dev	Test
WebQuestions (WebQ)	3,417	2,474	361	2,032
Natural Questions (NQ)	79,168	58,880	8,757	3,610
TriviaQA	78,785	60,413	8,837	11,313

Table 5: OpenQA dataset statistics. The training set is used for end-to-end training of the QA models whereas the filtered training set is used for supervised training of the retriever (i.e., for DPR experiments). The filtered set ignores those question-answer pairs where the evidence (Wikipedia) document retrieved using BM25 (Robertson and Zaragoza, 2009) does not align with the provided gold context documents. We leverage the filtered training set as provided by (Karpukhin et al., 2020).

A Dataset Details

Dataset statistics. For validation, we randomly select approximately 10% examples from the training set. For all the datasets, we use the dataset splits from (Lee et al., 2019). We provide the size of the training, development, and test sets in Table 5.

Pre-processing. For TriviaQA experiments, following (Izacard and Grave, 2021a), we select human-annotated answers for training the QA model. We also filter out those questions whose answer length is more than 5 words. Overall, this filters out 2,362 examples from the training set.

Dataset license and URLs. All the datasets are open-source and widely used by the community. Below, we provide the URLs of the actual dataset source and their preprocessed version which is used in this work.

- NQ: *dataset*: `https://ai.google.com/research/NaturalQuestions/download`, *license*: `https://github.com/google-research-datasets/natural-questions/blob/master/LICENSE`
- TriviaQA: *dataset*: `http://nlp.cs.washington.edu/triviaqa/`, *license*: `https://github.com/mandarjoshi90/triviaqa/blob/master/LICENSE`
- WebQ: *dataset*: `https://github.com/google-research/language/tree/master/language/orqa#getting-the-data`, *license*: `https://nlp.stanford.edu/software/sempre/`
- Preprocessed version: We make use of NQ, TriviaQA, and evidence datasets as open-sourced by Karpukhin et al. (2020) here: `https://github.com/facebookresearch/DPR/blob/master/data/download_data.py`.

B Additional Training Details

In addition to the details provided in §3.2, here, we provide further training details for reproducibility.

BERT and Inverse Cloze Task (ICT). We derive the implementations of BERT (Devlin et al., 2019) and ICT (Lee et al., 2019) from the open-source Megatron-LM toolkit.[9] For ICT, the dual-encoder retriever is initialized with BERT weights and then we train the model according to Lee et al. (2019). For training, we use Wikipedia paragraphs where we truncate the maximum length of a paragraph to 256 tokens. We list the settings and hyperparameters used for training BERT and ICT in Table 6.

T5. We derive the implementation of T5 (Raffel et al., 2020) language model from the open-source Megatron-LM toolkit (Shoeybi et al., 2019). We list the hyperparameters used for training T5 in Table 6. For consistency, we train T5 for the same number of steps and batch size as was done in the original paper. Additionally, we use BERT lowercase tokenization for both T5 and BERT.

---

[9] `https://github.com/NVIDIA/Megatron-LM`

Hyperparameter	BERT	ICT	T5	MSS
Dataset	Wikipedia, BookCorpus	Wikipedia	C4, Wikipedia, OpenWebText	Wikipedia
Num. Parameters	110M	220M	220M	440M
Hidden Size	768	768	768	768
Attention heads	12	12	12	12
Dropout	0.1	0.1	0.1	0.1
Optimizer	Adam	Adam	Adam	Adam
Batch Size	256	4096	2048	64
Training Steps	1M	100K	1M	82K
Warmup Ratio	0.01	0.01	0.01	0.05
Max. Learning Rate	1e-4	1e-4	1e-4	2e-5
Weight Decay	1e-2	1e-2	1e-2	1e-1
Learning Rate Decay	Linear	Linear	Linear	Linear
Gradient Clipping	1.0	1.0	1.0	1.0

Table 6: Hyperparameters for training BERT, ICT, T5, and MSS models.

Hyperparameter	NQ	TriviaQA	WebQ
Num. Parameters	440M	440M	440M
Hidden Size	768	768	768
Attention heads	12	12	12
Dropout	0.1	0.1	0.1
Optimizer	Adam	Adam	Adam
Batch Size	64	64	16
Epochs	10	10	20
Warmup Ratio	0.01	0.01	0.01
Max. Learning Rate	2e-5	2e-5	2e-5
Weight Decay	1e-1	1e-1	1e-1
Learning Rate Decay	Linear	Linear	Linear
Gradient Clipping	1.0	1.0	1.0
Temperate (τ)	27.7	27.7	27.7

Table 7: Hyperparameters for finetuning on NQ, TriviaQA, and WebQ datasets.

Unsupervised pre-training with masked salient spans (MSS). For MSS training, we initialize the retriever of our model from the ICT weights and the reader from the T5 weights. We make use of the Stanza toolkit (Qi *et al.*, 2020) to segment evidence documents into sentences. We then extract named entities from these sentences using the NER model trained on the OntoNotes-5.0 dataset as provided by Stanza. These names entities are replaced by mask tokens. As the masked tokens correspond to special named entities, they are referred to as salient spans. The masked sentence is considered as the question to retrieve evidence documents and the reader is trained to generate the named entities corresponding to the masked salient spans with the help of retrieved documents. During retrieval, we ignore the evidence document from which the masked sentence was derived. We list the hyperparameters of MSS training in Table 6.

Supervised training using the question-answer pairs. We provide the training details in §3.2. We list the hyperparameters in Table 7. Apart from the number of epochs and batch size in WebQ, we use the same hyperparameters for all the experiments. For the temperature parameter (τ) in Eq. 5, we follow Sachan *et al.* (2021) and set it as the square root of the hidden size.

Training Time. We run all of our experiments on a machine with 96 CPUs, 1.3TB physical memory, and 16 A100 GPUs. We use PyTorch (Paszke *et al.*, 2019) to implement our proposed model. With this hardware setup, our experiments on NQ and TriviaQA took approximately 25 hours to complete, while experiments on WebQ took roughly 8 hours to complete. Before supervised training, we also perform a one-time unsupervised MSS pre-training for 82,000 steps that took roughly 1 week.

Method	R@5 after ICT	R@5 after MSS
REALM (Guu *et al.*, 2020)	13.9	38.5
EMDR^2	28.0	38.6

Table 8: Retrieval recall on the NQ development set after ICT and MSS pre-training.

Method	Evidence Size	Evidence Dimension	GPU Memory (in FP16)
REALM (Guu *et al.*, 2020)	13M	128	3 GB
EMDR^2	21M	768	30 GB

Table 9: Comparison of evidence embeddings storage for retrieval.

C Unsupervised Pre-training and Comparisons with REALM

We make use of a couple of training techniques introduced in the REALM paper (Guu *et al.*, 2020): masked salient spans (MSS) pre-training and asynchronous evidence embedding update. There are similarities and differences in the way in which we apply these ideas to EMDR^2 training.

C.1 ICT and MSS Pre-training

Both ICT and MSS are unsupervised techniques used to bootstrap the retriever so that it has a good initial recall.

We first initialize the retriever with ICT pre-training. For ICT, similar to REALM, we follow the settings in the ORQA paper (Lee *et al.*, 2019). We observe our Recall@5 to be much higher than that reported in the REALM paper (see Table 8). We believe that our choice of 768 dimensional embedding of each evidence document leads to better results when compared to the 128 dimensional embedding used in REALM.

We further pre-train with MSS once the retriever weights are initialized with ICT. We use a batch size of 64 and train for 82K steps using the EMDR^2 objective. In comparison, REALM uses a batch size of 512 and trains the model for 200K steps. Even with a much smaller batch size and training steps, EMDR^2 achieves similar Recall@5 after MSS training (Table 8). We hypothesize that with a large batch size and longer training, EMDR^2 would be able to further improve its recall. Another implementation detail is that EMDR^2 does not require the additional null document which was used in REALM.

For low-resource datasets such as WebQ, MSS pre-training also improves the performance of the FiD reader. As Table 3 illustrates, on WebQ, MSS pre-trained reader obtains a gain of more than 1 EM point over the T5 reader (shaded in orange color).

C.2 Asynchronous Evidence Embedding Updates

The asynchronous evidence embedding updates are performed after every 500 steps of training and is similar to REALM with a couple of differences. In our work, asynchronous embedding updates is done both during MSS pre-training and supervised training, while in REALM it is performed only during MSS pre-training. The second difference, although a minor one, we needed to compute the embeddings of 21M evidence documents while REALM had to do this for 13M documents. We do this by having two process groups during training, one group trains the model on 8 GPUs while the other group performs evidence embedding computation on 8 GPUs in an asynchronous manner.

C.3 Pre-computed Evidence Embeddings Storage for Retrieval

In Table 9, we provide some comparisons between REALM and EMDR^2 to showcase that the retrieval task is more challenging in our setting. Firstly, the size of evidence in REALM is 13M because each Wikipedia article is split into 288 wordpieces while the size of evidence in EMDR^2 is 21M as each Wikipedia article is split into 100 linguistic words. Second, the embedding dimension of each

evidence document in REALM is 128 while the embedding dimension of each evidence document in EMDR^2 is 768. Due to these factors, the memory required by REALM to store evidence embeddings (in FP16) is approximately 3 GB, while the memory required by EMDR^2 to store evidence embeddings (in FP16) is 30 GB. As the GPU RAM is constrained by its capacity (40 GB maximum in A100 GPUs), it was not possible to store the entire 30 GB embeddings in each GPU. Therefore, for online retrieval, we store the evidence embeddings in a distributed fashion over 16 GPUs and perform distributed asynchronous MIPS for fast retrieval.

D Comparison with Previous Work

Here we provide a discussion of how EMDR^2 is different from some of the previous work.

D.1 Comparison with Hard EM and Reinforced Reader-Ranker Models

There are some similarities between EMDR^2 and $\mathcal{L}_{\text{alt-2}}$ to Hard EM (Min *et al.*, 2019) and Reinforced Reader-Ranker (R^3; Wang *et al.* (2018)), at the conceptual level even though they are not equivalent. Training with REINFORCE involves sampling from a policy network (i.e., the retriever in our case). We take a deterministic approach and take the top-K documents in both EMDR^2 and $\mathcal{L}_{\text{alt-2}}$. Compared to Hard EM, $\mathcal{L}_{\text{alt-2}}$ directly minimizes the KL divergence of the probability of a retrieved document with the probability of an answer given that document.

At the implementation level, there are many other differences between $\mathcal{L}_{\text{alt-2}}$ (and EMDR^2) with models in (Min *et al.*, 2019) and (Wang *et al.*, 2018). First, we would like to note that both these methods use TF-IDF and BM25 as their retrieval approach which are not trainable. In contrast, our work uses a dense retriever which is trained in an end-to-end manner. We list other differences in more detail below.

Differences with Hard EM. Min *et al.* (2019) propose a hard EM approach to train an extractive reader model for QA tasks. The context document is assumed to contain multiple mentions of the correct answer. They propose an objective to train the reader. Specifically, during the training step, the model is trained using maximum marginal likelihood for the first τ steps and subsequently with their proposed logmax objective. In their open-domain QA experiments on TriviaQA and NQ, the retriever part is based on TF-IDF and BM25 and is non-trainable. Overall, their model is applicable to extractive readers without retriever training. In comparison, in EMDR^2, we train both the reader and retriever. As such, the hard EM approach is not directly applicable to our case.

Differences with R^3. This paper involves three pipelined components: retriever, ranker, and reader. The retriever is BM25 based and is non-trainable. They jointly train the ranker and the reader. The ranker takes 100 documents from the retriever and selects one document to give as input to the reader (contrast this with our work that selects a set of documents). As this selection operation is non-differentiable, their model leverages policy gradient to train the ranker. They also propose a custom reward function based on the overlap of text between the extracted answer and the correct answer. The reader takes a single document as input. In contrast, our approach does not involve a ranker component, both the FiD reader and retriever are trainable, and our proposed objective function EMDR^2 is end-to-end differentiable.

D.2 Comparison with Individual Top-K and Joint Top-K Models

Comparison with Individual Top-K (Sachan *et al.*, 2021). Individual Top-K is another approach for end-to-end training but the difference is that it applies a single-document reader while EMDR^2 consists of a multi-document reader. Similar to previous methods like REALM and RAG, Individual Top-K objective function is also defined over multiple retrieved documents but is better optimized than them. As the performance of EMDR^2 is much better than Individual Top-K, EMDR^2 is a better modeling approach.

Comparison with Joint Top-K (Sachan *et al.*, 2021). While both EMDR^2 and Joint Top-K are end-to-end training approaches for open-domain QA based on the FiD model, they are different in many ways. (i) *Different Objective Functions*: These approaches optimize different training objectives. To

achieve retriever training, Joint Top-K adds the retrieval probability score of the top-K documents to the unnormalized inter-attention scores. In this way, the reader pays more importance to those top-K documents with a higher retriever score. There is no explicit feedback from the reader to the retriever. In contrast, the second term in the training objective of EMDR^2 explicitly encourages the retriever to improve its predictions based on the agreement with the reader's answer-generation likelihood of a particular top-K document. (ii) *Task Performance*: EMDR^2 objective leads to a much improved end-to-end training algorithm. This is reflected by the performance gains over the FiD baseline. On NQ and TriviaQA, while EMDR^2 leads to 4.3 and 6.4 EM points improvements respectively, Joint Top-K obtains a much lower gain of 1 point improvement on NQ and no improvements on TriviaQA. This demonstrates that EMDR2 training leads to substantially better retrieval, that in turn leads to higher gains in answer generation. These results also illustrate that EMDR^2 is a much better end-to-end or joint training algorithm than Joint Top-K for the multi-document reader retriever approaches.

E Qualitative Analysis

In Table 10, we present some representative examples of the retriever output with both MSS pre-training and when the MSS pre-trained model is finetuned on NQ. We observe that after MSS pre-training, the top-1 outputs are related to the question but are not relevant enough to answer them. However, when the MSS pre-trained model is finetuned on NQ with EMDR^2, the retrieval accuracy improves with the top-1 documents being much more relevant to answer the question. The retriever's confidence score of the top-1 document also improves.

Comparing retriever initializations.   We analyze the reader's training loss when the retriever is either initialized with unsupervised MSS training or with first MSS pre-training followed by supervised DPR training (MSS + DPR). As indicated in Table 3, MSS pre-training being unsupervised has a lower accuracy while MSS + DPR retriever has a higher accuracy. However, as is also evident from the plots in Figure 4, retriever initialization has a marginal effect on the answer generation performance. We see that for NQ, for the first 1200 steps, the higher accuracy MSS + DPR retriever leads to a smaller training loss compared with the MSS retriever, after which the difference between the two training losses diminishes as the end-to-end training improves the accuracy of the MSS retriever. Similar trends are also observed for TriviaQA and WebQ but to a lesser extent.

Visualizing reader and retriever losses.   In Figure 3, we show the trajectories of the reader and retriever training losses when the model is initialized with MSS pre-training.

Questions from NQ test	Answer	MSS Pre-training	EMDR[2] finetuned on NQ
what type of reaction occurs to form a dipeptide	peptide bond	probability=0.39 ...Bornyl diphosphate synthase In enzymology, bornyl diphosphate synthase (BPPS) () is an enzyme that catalyzes the chemical reaction Bornyl diphosphate synthase is involved in the biosynthesis of the cyclic monoterpenoid bornyl diphosphate. As seen from the reaction above, BPPS takes geranyl diphosphate as its only substrate and isomerizes into the product, (+)- bornyl diphosphate. This reaction comes from a general class of enzymes called terpene synthases that ...	probability=0.78 ...Subsequent to this coupling reaction, the amine protecting group P and the ester are converted to the free amine and carboxylic acid, respectively. For many amino acids, the ancillary functional groups are protected. The condensation of the amine and the carboxylic acid to form the peptide bond generally employs coupling agents to activate the carboxylic acid. The Bergmann azlactone peptide synthesis is a classic organic synthesis for the preparation of dipeptides. ...
when was the japanese videogame company nintendo founded	23 September 1889	probability=0.37 ...contributed to the development of the following games. Creatures (company) Ape, Inc. was founded in March 1989 and Shigesato Itoi became its chief executive officer. Nintendo president Hiroshi Yamauchi had wanted to support new talent in game design. Liking Itoiś work, he proposed the idea of the company to Itoi and invested in it. Apeś staff included Tsunekazu Ishihara, who later became the Pokémon Companyś CEO, and Ashura Benimaru Itoh, a renowned illustrator. They began work on "Mother", which released in July. Its music was composed by Hip Tanaka, who later became the second CEO of Creatures ...	probability=0.61 ...Nintendo Co., Ltd. is a Japanese multinational consumer electronics and video game company headquartered in Kyoto. Nintendo is one of the world's largest video game companies by market capitalisation, creating some of the best-known and top-selling video game franchises, such as "Mario", "The Legend of Zelda", and "Pokémon". Founded on 23 September 1889 by Fusajiro Yamauchi, it originally produced handmade hanafuda playing cards. By 1963, the company had tried several small niche businesses, such as cab services and love hotels. Abandoning previous ventures in favour of toys in the 1960s ...

Table 10: Examples of top-1 retrieved documents from the NQ test when the model is pre-trained with Masked Salient Spans (MSS) or finetuned on NQ data. If the answer exists in the document it is highlighted in blue color, and the probability of the document (Eq. 5) is indicated in orange color.
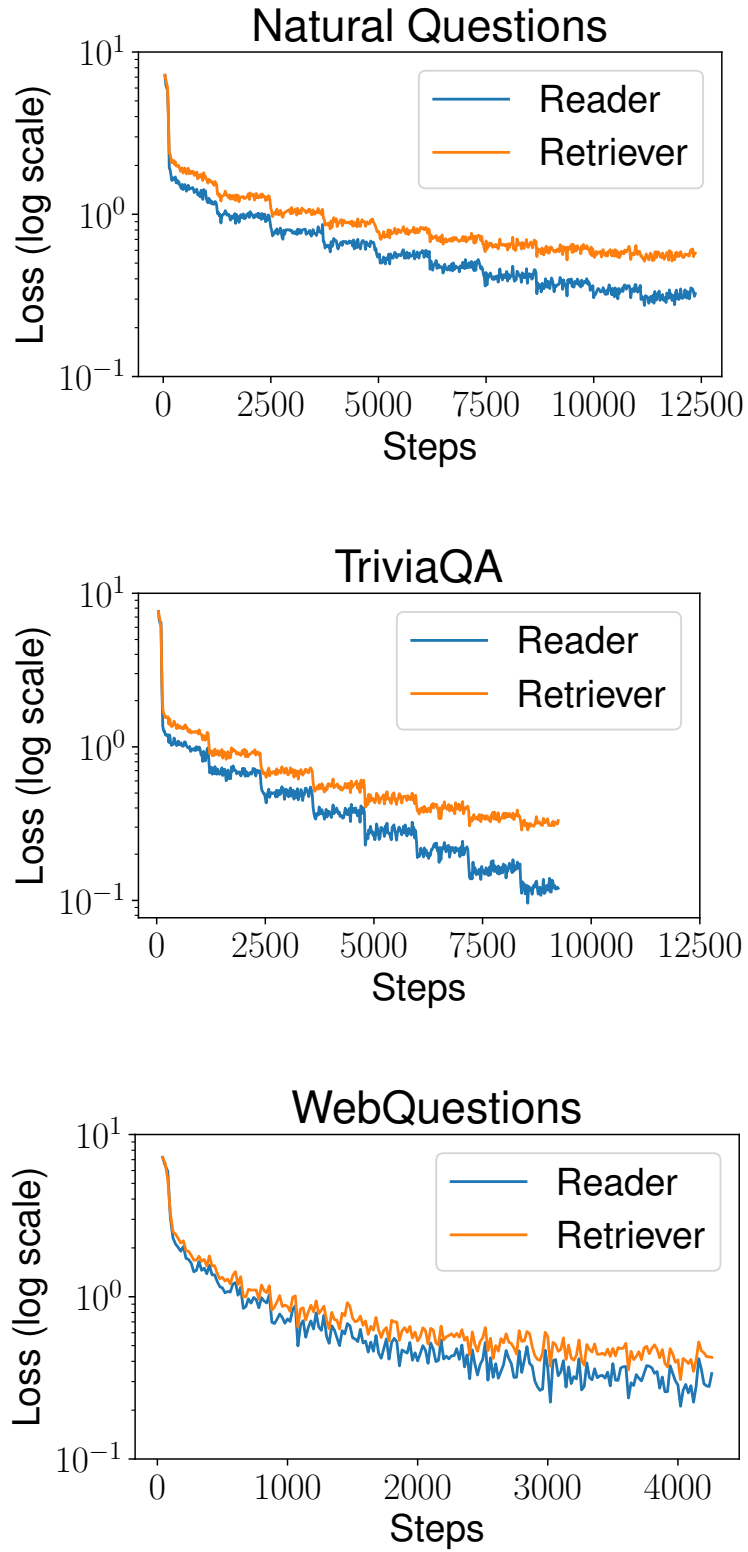
Figure 3: Reader and retriever training losses when the model is initialized with MSS pre-training.
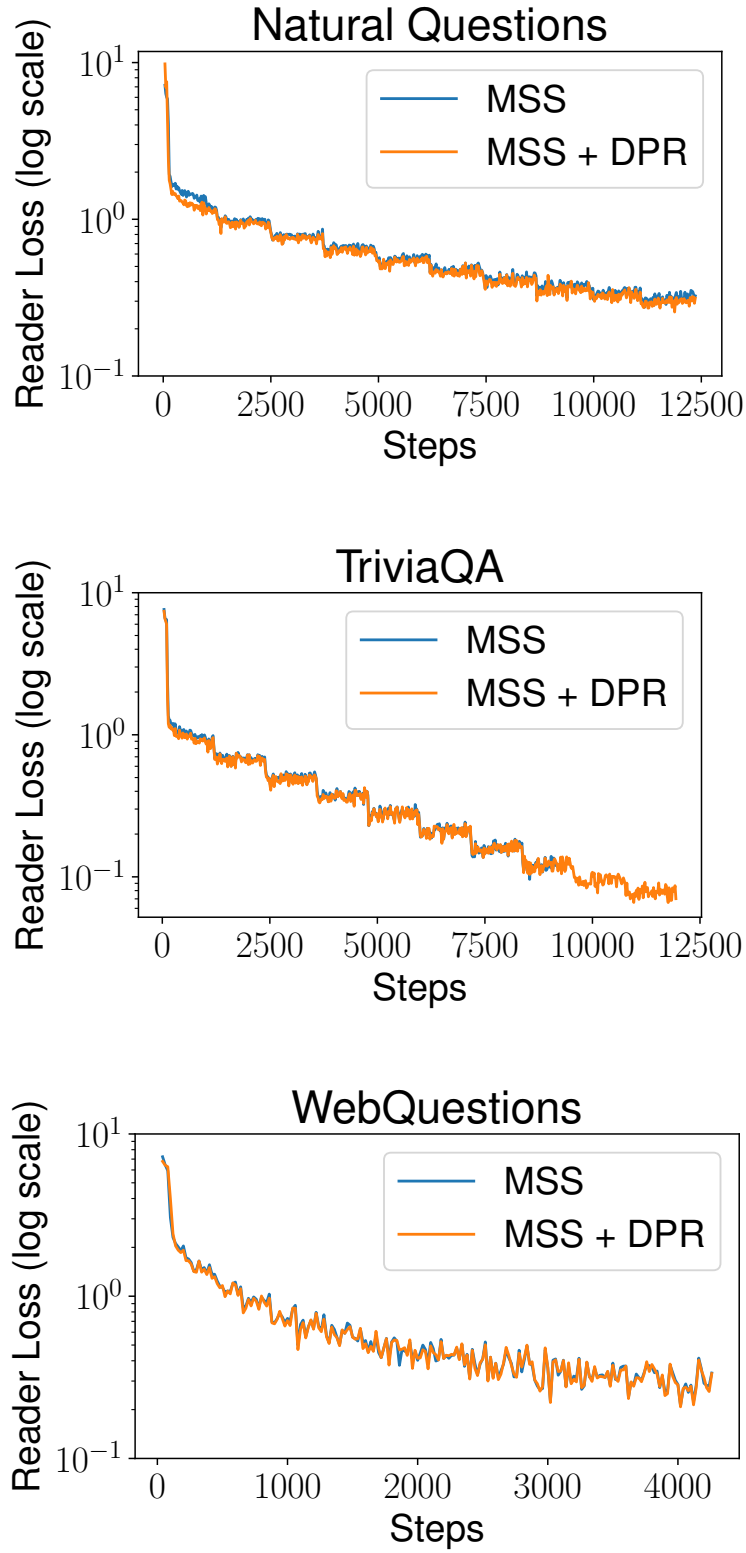
Figure 4: Reader training loss vs steps for NQ, TriviaQA, and WebQ when the retriever is either initialized by MSS pre-training or by MSS followed by supervised DPR training (MSS + DPR).