Hierarchical Representations in Dense Passage Retrieval for Question-Answering

Philipp Ennen, Federica Freddi, Chyi-Jiunn Lin, Po-Nien Kung
RenChu Wang, Chien-Yi Yang, Da-Shan Shiu, Alberto Bernacchia
MediaTek Research
{philipp.ennen, alberto.bernacchia}@mtkresearch.com

Abstract

An approach to improve question-answering performance is to retrieve accompanying information that contains factual evidence matching the question. These retrieved documents are then fed into a reader that generates an answer. A commonly applied retriever is dense passage retrieval. In this retriever, the output of a transformer neural network is used to query a knowledge database for matching documents. Inspired by the observation that different layers of a transformer network provide rich representations with different levels of abstraction, we hypothesize that useful queries can be generated not only at the output layer, but at every layer of a transformer network, and that the hidden representations of different layers may combine to improve the fetched documents for reader performance. Our novel approach integrates retrieval into each layer of a transformer network, exploiting the hierarchical representations of the input question. We show that our technique outperforms prior work on downstream tasks such as question answering, demonstrating the effectiveness of our approach.

1 Introduction

In open book question answering, the answer to a given question needs to be generated from a large pool of passages. Typically, this problem is tackled in two stages. Given a question, a retriever collects a set of top-k passages from the passages memory. Then, a reader generates the answer from the retrieved documents. In this setting, dense passage retrieval (DPR) is a commonly used retriever (Karpukhin et al., 2020). Therein, each passage in a document collection is represented as a vector in a high-dimensional space. These vectors are then used to compute similarity scores between passages. The most similar passages are then retrieved and used as input to a machine learning model.

However, we observe that current open-book QA systems do not adequately exploit the correlations between passages in the retrieved document collection. Typically, questions contain several word features that need to be represented in the retrieved document collection, i.e. questions in Figure 1. In order to answer such questions, the reader needs to reason about multiple word features of the question simultaneously. However, we found that many questions have a critical feature that is underrepresented in the retrieved documents (i.e. Ivy League, Ming, Baltic Republic). In order to improve the QA performance, we propose to increase the occurrence on these critical features.

When did Harvard become an Ivy League school?	
FetcHR:	*Harvard*: 300, *Ivy League*: 109
DPR:	*Harvard*: 341, *Ivy League*: 66
Who overthrew the Mongols and established the Ming Dynasty?	
FetcHR:	*Mongols*: 108, *Ming*: 112
DPR:	*Mongols*: 108, *Ming*: 87
When did the Soviet Union first gain control of parts of Poland and the Baltic Republics?	
FetcHR:	*Soviet Union*: 139, *Poland*: 93 , *Baltic Republics*: 7
DPR:	*Soviet Union*: 133, *Poland*: 214, *Baltic Republics*: 2

Figure 1: We present the occurrences of word features inside the document collection as retrieved by either DPR or FetcHR. The feature from the question with the lowest occurrence is the critical feature for QA tasks. Our retriever FetcHR outperforms DPR on critical word features in the question (underlined). Our work shows that this improved document collection increases the reader performance by up to 1.9 EM score on Natural Question and 2.1 EM score on WebQuestion.

Typically, the retrieved document collection matches the highest abstraction level of an input question. We hypothesize that a document collection addressing different, hierarchical abstraction levels of an input question may improve on the critical features. With these documents, the reader improves the performance on question-answering benchmarks.

We present a novel retriever architecture and training procedure to test this hypothesis (Figure

Proceedings of the Sixth Fact Extraction and VERification Workshop (FEVER), pages 17–28

2). Our architecture extends BERT (Devlin et al., 2019) with a neural retrieval network producing queries not only at its output layer but also at intermediate layers. This is inspired by the observation that different layers provide different level of abstraction (Rogers et al., 2020) that can be all used for downstream tasks (Evci et al., 2022). Under this setup, the retrievable documents embody a non-parametric knowledge of the transformer (Guu et al., 2020). With a separate reader function, an answer is inferred from the documents retrieved by the hierarchical retrievers (see section 3 for details of the model).

Our main contributions are:

- We introduce FetcHR, a document fetcher based on hierarchical retrieval. We equip transformer layers with a neural retrieval network allowing hidden representations to contribute to the retrieval query.
- We show that retrieval performance of all layers combined is higher than any of the individual layers, in most of our experiments. This allows improving performance of previous models, which considered retrieval from single layers only.
- When using a reader to generate the answer to the input question, we show that documents retrieved by FetcHR obtain the highest performance in all experiments, and advance the state-of-the-art on the Natural Question and WebQuestion datasets.
- All results are obtained by training only the retrieval networks. This avoids any modification of the underlying language model, making it feasible to customize a large pretrained language model with moderate training resources.

2 Related Work

Retrieval Question-answering tasks are usually tackled introducing retrieval components in order to efficiently select a subset of relevant documents (Voorhees et al., 1999). In the past, Q&A tasks would be generally attempted using sparse vector space models such as BM25 and TF-IDF (Chen et al., 2017; Yang et al., 2019; Nie et al., 2019; Wolfson et al., 2020; Min et al., 2019). In the past few years, these were replaced by transformer-based models mapping a model input to a dense vector representation. There are mainly two approaches of neural network-based retrievers based on single or multiple embedding vectors (Singh et al., 2021). Dual encoders belong to the single embedding approaches. Such retrievers use one encoder for the documents and another one for the query (Yih et al., 2011; Lee et al., 2019). Dense Passage Retrieval (DPR) (Karpukhin et al., 2020) uses two BERT-style models to learn a similarity metric between document and query. In case of multi-vector retrievers instead, multiple embeddings are generated for each document, such as in (Khattab and Zaharia, 2020; Zhang et al., 2022; Luan et al., 2021). However this approach is computationally limiting in large-scale retrieval since it requires to retrieve in many search spaces (up to the document token length) leading to increased memory needs and search time. Instead we propose to perform retrieval in each layer of a BERT-based transformer network limiting the retrieval runs to 12. This approach borrows ideas from early work in information retrieval on multi-layer matching (Nie et al., 2018a,b), however retrieval is performed based on an aggregated score over all layers. Instead, our approach performs retrieval in each layer, while using a more modern transformer architecture, scales to large-scale retrieval and is evaluated on end-to-end QA.

Training of Dual Encoders Past work has shown (Qu et al., 2021; Guu et al., 2020; Lewis et al., 2020; Singh et al., 2021) that the performance of dual encoders can be improved by (i) carefully selecting negative documents for the contrastive loss and (ii) by an end-to-end training with a reader function. Cross-batch negatives, denoised hard negatives and data augmentation are options mentioned in (Qu et al., 2021) for negative documents. End-to-end training approaches tune the retrieval function to match the distribution of data for the reader (Lewis et al., 2020; Guu et al., 2020; Singh et al., 2021). These approaches are orthogonal to our work and might be considered as an extension for future work.

3 FetcHR Model

3.1 Retrieval Score

The purpose of a retrieval system is to choose a selection of documents from a collection of documents (referred to as the "Document Memory") that contains pertinent information to answer a given

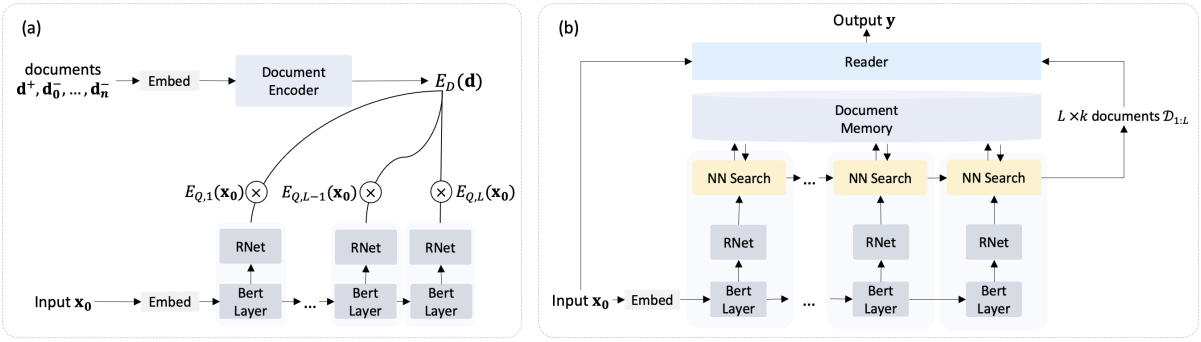

Figure 2: FetcHR is a stack of BERT layers with an additional Retrieval Network (RNet) at each layer. (**a**) Training: The inner product between the encoded input question at each layer $E_{Q,\ell}(\mathbf{x}_0)$ and the encoded document $E_D(\mathbf{d})$ is maximised for positive documents $\mathbf{d}^+$ and minimized for negative documents $\mathbf{d}^-$, using a contrastive loss. (**b**) Inference: Using nearest neighbour search, FetcHR retrieves k documents per layer from the document memory, for a total $L \times k$ documents. The reader outputs the answer $\mathbf{y}$ given the hierarchy of retrieved documents combined with the model input $\mathbf{x}_0$

tokenized input question $\mathbf{x_0}$. Our novel retrieval system *FetcHR* gathers documents for different hierarchical representations of the model input. For this, we employ a multi-layer encoder architecture. Each layer ℓ has its own encoder function $E_{Q,\ell}$, which is used to query the document memory. This allows us to define a retrieval score as the inner product between the vector pairs $E_{Q,\ell}$ and E_D for all layers. Given a question $\mathbf{x_0}$ and document $\mathbf{d}$, the retrieval score for layer ℓ can be computed via the inner product between the vector pairs $E_{Q,\ell}$ and E_D

$$\text{score}_\ell = E_{Q,\ell}(\mathbf{x}_0; \boldsymbol{\theta}_{1:\ell}, \boldsymbol{\phi}_\ell) \cdot E_D(\mathbf{d}; \boldsymbol{\omega}) \quad (1)$$

The parameters $\boldsymbol{\theta}_{1:\ell}$ are shared by the first ℓ layers, while $\boldsymbol{\phi}_\ell$ corresponds to the parameters specific to layer ℓ, and $\boldsymbol{\omega}$ is the parameter vector of the document encoder. Consequently, the retrieval score in layer ℓ depends on a layer-specific question encoder $E_{Q,\ell}$ and a single, shared document encoder E_D for all layers.

3.2 Contrastive Training

During training, we present data points to the model. Each one is a tuple of an input question $\mathbf{x}_0$, a positive document $\mathbf{d}^+$ containing the correct answer to the question and n negative (randomly chosen) documents $\mathbf{d}_1^-, ..., \mathbf{d}_n^-$. The training goal is to improve the retrieval score of all layers at the same time. Following (Karpukhin et al., 2020), we adopt the contrastive loss function which, for a single data point $\{\mathbf{x}_0, \mathbf{d}^+, \mathbf{d}_1^-, ..., \mathbf{d}_n^-\}$, is equal to

$$\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{\omega}) = \frac{1}{L} \sum_{\ell=1}^{L} -\log \frac{e^{\text{score}_\ell(\mathbf{x}_0, \mathbf{d}^+)}}{e^{\text{score}_\ell(\mathbf{x}_0, \mathbf{d}^+)} + \sum_{i=1}^{n} e^{\text{score}_\ell(\mathbf{x}_0, \mathbf{d}_i^-)}}$$

This loss is minimized by adjusting the parameters $\boldsymbol{\phi}$ while holding $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$ constant to the pre-trained values from (Karpukhin et al., 2020). The loss is calculated and averaged over a batch of data points at each iteration (see Section 4.2 for details).

3.3 Multi-layer encoder

FetcHR is build on top of BERT transformer layers (Devlin et al., 2019). We use a stack of these BERT layers and define for each layer ℓ a retrieval network *RNet* to generate single query vectors $\mathbf{q}_\ell = E_{Q,\ell}(\mathbf{x})$. Formally, the encoder function $E_{Q,\ell}$ is obtained according to

$$\begin{aligned} \mathbf{x}_\ell &= \mathit{BertLayer}(\mathbf{x}_{\ell-1}, \boldsymbol{\theta}_\ell) \qquad \text{for } \ell = 1, \ldots, L \\ E_{Q,\ell} &= \mathit{RNet}(\mathbf{x}_\ell; \boldsymbol{\phi}_\ell) \end{aligned} \quad (2)$$

where the parameters $\boldsymbol{\phi}_\ell$ corresponds to the retrieval network and $\boldsymbol{\theta}_\ell$ to BERT layer ℓ. Note that the encoder function $E_{Q,\ell}$ depends on the input question $\mathbf{x}_0$ and the parameters $\boldsymbol{\theta}_{1:\ell}$ of all upstream BERT layers through $\mathbf{x}_\ell$.

The output of the retrieval network *RNet* is the embedding at the CLS position of the output of a stack of two transformer layers. Each layer is a BERT transformer layer, with the skip connection placed outside of the layer normalization:

$$\begin{aligned} &\mathit{LayerNorm}(\mathbf{W}_2(\sigma_1(\mathbf{W}_1(\mathit{AttnLayer}(\cdot))))) \\ &\qquad + \mathit{AttnLayer}(\cdot) \end{aligned} \quad (3)$$

where W and σ are the weights and activation function, respectively, of a two-layer MLP as in standard self-attention. Placing the skip connection outside of layer normalization is advantageous when *BertLayer* is initialized with pre-trained weights such that $\mathbf{x}_\ell$ already captures a meaningful abstraction of the input question. In all of our experiments, we use a combination of the *BertLayer* and the *RNet*, which we refer to as the *FetcHR layer*. We use $L = 12$ layers in total. All of the embeddings are 768-dimensional, as in the original BERT model. For the document encoder $E_D(\mathbf{d}; \boldsymbol{\omega})$, we use a BERT model in base configuration. The pre-trained parameters $\boldsymbol{\omega}$ come from (Karpukhin et al., 2020).

3.4 Inference

After training, retrieval on test data is implemented by matching the FetcHR encodings of each layer with document encodings in the document memory. For each layer ℓ, a set $\mathcal{D}_\ell$ containing k documents is retrieved by nearest neighbour search. During this search, we do not allow single documents to be retrieved more than once at multiple layers. More formally, given a question $\mathbf{x}_0$ during testing time, the optimal parameters $\boldsymbol{\phi}^\star$ obtained by training and the query vector $\mathbf{q}_\ell = E_{Q,\ell}(\mathbf{x}_0; \boldsymbol{\theta}_{1:\ell}, \boldsymbol{\phi}^\star_\ell)$, the set of retrieved documents at layer ℓ is equal to

$$\mathcal{D}_\ell = \mathit{NNSearch}(\mathbf{q}_\ell, \mathcal{D}_{1:\ell-1}, k) \tag{4}$$

where *NNSearch* returns the k nearest neighbours of $\mathbf{q}_\ell$ that are not included in the document sets $\mathcal{D}_{1:\ell-1}$ retrieved in previous layers. We use the Faiss-library for the implementation of *NNSearch* (Johnson et al., 2019). We investigated the performance on both exhaustive search on a flat index and the compressed IVF index (see section 4.2 for details).

With L layers, a total of $L \times k$ documents are retrieved. These retrieved documents are fed to the reader, together with the question $\mathbf{x}_0$, to obtain the answer. For the reader, we implement the state-of-the-art Fusion-in-Decoder (FiD) of (Izacard and Grave, 2021). Compared to other reader such as DPR-reader (Karpukhin et al., 2020) and REALM-reader (Guu et al., 2020) the FiD reader takes the retrieved document collection as an input simultaneously which allows to exploit the correlation between documents.

In a subset of experiments (e.g. Figure 3), we isolate retrieval in individual layers. In this case, each experiment retrieves from a single layer, without excluding any document.

4 Results

In the following, we evaluate how FetcHR influences the performance of modern readers and present isolated retrieval results of FetcHR.

4.1 Datasets

We test FetcHR on two commonly used open-domain question answering datasets:

- **WebQuestions** (Berant et al., 2013): This dataset includes questions collected using the Google Suggest API, with answers being entities from Freebase annotated by Mechanical Turk. Since only pairs of questions and answers are provided and no positive document, we follow (Karpukhin et al., 2020) by using the highest-ranked document from BM25 containing the answer span as positive document $\mathbf{d}^+$.
- **Natural Question** (Kwiatkowski et al., 2019): This dataset contains questions asked by users of Google-Search, with answers given as spans of text within Wikipedia articles. For each question, the positive document is the Wikipedia article containing the span with the answer.

The pre-processed English Wikipedia dump from December 2018 is used as the document memory as provided by (Karpukhin et al., 2020). This Wikipedia dump has been divided into non-overlapping chunks of 100 words following (Chen et al., 2017) and (Wang et al., 2019). Each chunk corresponds to a document. In total, there are $21,015,324$ documents.

4.2 Implementation Details

Hardware and libraries We use 32 Nvidia RTX 3090 GPUs with a total memory of 768 GB for training and testing of FetcHR. The distributed training is implemented in PyTorch with NCCL backend (Paszke et al., 2019). Our model implementation is is based on the Huggingface Transformers library (Wolf et al., 2020). The training time of the FetcHR retriever is $30-50$ hours for all our experiments. For training the FiD-large reader, we use a single Nvidia RTX A6000 GPU with 48 GB of RAM and gradient accumulation over 32 steps. This training takes about 100 hours.

Dataset details We follow the train/test splits from (Karpukhin et al., 2020) and we discard datapoints when the gold documents don't match the applied Wikipedia dump. This filtering process leaves us with a train set of 122,892 data points, which come from Natural Question, TriviaQA, WebQuestions, and CuratedTREC.

Training details We initialize the model parameters using the multiset checkpoint that was trained and provided by DPR. For the contrastive loss function, we use in-batch negatives. Our total training time is 30 epochs. The learning rate is $2 \cdot 10^{-5}$, and we use Adam optimizer (Kingma and Ba, 2014), linear scheduling with warm-up, and a dropout rate of 0.1. Our batch size is 256. We evaluate two checkpoints after training: the one with the lowest validation loss and the last checkpoint. The best performing checkpoint is reported in this paper. For distributing the training over multiple GPUs, we compute the scores on each GPU first and gather these scores to compute the loss. Then, we reduce the loss back to each GPU and compute local gradients. The final gradient update is averaged over all local gradients.

Retrieval and search In the nearest neighbor search function "NNSearch", documents that have already been retrieved are excluded by iteratively increasing $k \rightarrow k'$ until k new documents are retrieved. The underlying search algorithm uses the Faiss library. However, due to computational limitations, in most experiments, we use an IVF search index with 131072 clusters and 128 inference probes. We end-to-end ran experiments using both the IVF index and a flat index for exhaustive search. The IVF index was built on four Nvidia RTX 3090 GPUs that sums up to 96 GB VRAM. During inference, we ran IVF and flat index on CPU with access to 2 TB of physical memory. The total search time required for FetcHR is about 0.5 s on IVF index and 150 s on the flat index for a single inference.

4.3 Baselines

We compare FetcHR with DPR (Karpukhin et al., 2020) by testing it on the state-of-the-art Fusion-in-Decoder reader (FiD, (Izacard and Grave, 2021)). DPR is the best performing retriever in the original FiD publication and serves as a strong baseline.

The authors of DPR published multiple checkpoints of their work, some of which are trained on a single dataset (Natural Question) while others are trained on multiple datasets (Natural Question, TriviaQA, WebQuestions, CuratedTrec). We compare with the checkpoint trained on multiple dataset, which also serves as initialization of our model. For a fair comparison, we re-evaluate the DPR checkpoint on the same search index (IVF and flat), with identical retrieval budget $L \times k$, identical tokenization and entity normalization. Retrieved documents are fed to the FiD reader with pre-trained weights, as provided by the authors. We use the Fusion-in-Decoder in either base or large configuration. Fusion-in-decoder was trained on the Natural Question, TriviaQA and SQuAD v1.1 on 100 retrieved documents. We finetuned FiD-large to read from 240 documents from the Natural Question train set for 1 epoch. For the WebQuestion dataset, we performed an additional finetuning of the previously obtained Natural Question checkpoint to compensate for the small size of the WebQuestion dataset. We stopped this finetuning after 15 epochs. We denote the finetuned checkpoints as *FiD-large trained* in our tables.

4.4 Performance Metrics

Since the critical occurrence is a qualitatively metric which cannot be measured automatically, we follow the standard convention of retrieval accuracy and exact match from related work to measure the performance of our model as follows:

Retrieval Accuracy The retrieval accuracy is the probability that the correct answer span is included in one of the documents that are retrieved. Normalizations are performed, such as lower casing as well as removing punctuation and articles. This accuracy is commonly used to evaluate retrieval systems but is not capturing the occurrence of critical features in the retrieved document collection (see Figure 1).

Exact Match The performance of a reader is measured by the exact match (EM) score. This score is calculated by the percentage of exact matches between the reader's output and the correct answer. This score provides an end-to-end QA score capturing also the occurrence of critical features in the document collection.

4.5 Main Results

Figure 3 illustrates the retrieval accuracy of single FetcHR layer. We consider retrieval budgets of $L \times k = 10$ and $L \times k = 120$ and compare

Retriever	NQ		WebQ	
	top-120	top-240	top-120	top-240
DPR-IVFIdx	81.7	83.8	81.6	83.4
FetcHR-IVFIdx	**83.9**	**85.6**	**82.5**	**84.4**
DPR-flat	**86.5**	**88.2**	**85.2**	**87.3**
FetcHR-flat	**86.5**	**88.2**	84.8	86.8

Table 1: Top-120 and -240 accuracy's for different retriever on a flat and a compressed search space (IVFIdx).

Retriever	FiD-base		FiD-large				FiD-large trained	
	top-120 NQ	top-240 NQ	top-120 NQ	top-240 NQ	top-120 WebQ	top-240 WebQ	top-240 NQ	top-240 WebQ
DPR-IVFIdx	41.0	41.5	45.7	46.4	24.9	26.2	46.6	36.9
FetcHR-IVFIdx	**43.7**	**43.7**	**48.1**	**49.3**	**26.1**	**27.2**	**48.7**	**39.9**
DPR-flat	46.0	46.0	50.5	51.3	27.0	28.1	51.6	40.7
FetcHR-flat	**46.7**	**46.6**	**50.8**	**52.0**	**27.1**	**28.2**	**53.3**	**48.0**

Table 2: Exact-match scores with different readers on Natural Question and WebQuestion test sets.

the results to DPR which always retrieves at the final output layer 12. In contrast to the rest of this work, where FetcHR retrieves from all/multiple layers simultaneously ($L > 1$), in this experiment we retrieve from individual layers separately and independently ($L = 1$). Figure 3 reveals that the retrieval accuracy improves as the layer becomes deeper. The final layer achieves the best performance, while none of the FetchHR-layers outperforms DPR on its own, despite the last FetchHR-layer reaches nearly the same accuracy as DPR.

In the second experiment, we show that retrieving from all layers simultaneously achieves higher performance than the best individual layer, for an equal total number of retrieved documents. With $L = 12$ layers and $k = \{10, 20\}$, we consider total budgets of $L \times k = 120$ and $L \times k = 240$, respectively. We combine all retrieved documents as described in Section 3.4. The results are shown in Table 1. FetcHR's accuracy using all layers is higher than DPR when the IVF index is used. However, it is equal or slightly worse when the flat index is used. This might be a consequence of FetcHR being able to explore a compressed search space efficiently. While retrieval accuracy of FetcHR is not better than DPR for the flat index, in the next experiment we find that FetcHR performance is always higher when integrated into a QA system containing a reader to generate the answer.

We apply the Fusion-in-Decoder, a state-of-the-art reader, in the third experiment. This reader takes the retrieved documents combined with the input question as input and generates the answer. The exact match score is shown in Table 2 for question-answering tasks from the Natural Question and Web Question datasets. The documents provided by FetcHR always enable the reader to score higher than with the documents provided by DPR for all datasets and documents budgets. This is especially significant for the IVF index and the finetuned FiD reader, but it holds consistently also for the other scenarios, despite the lower retrieval performance shown in Table 1. These results confirm that the FetcHR document collection is superior compared to the DPR documents for QA tasks. We conclude that this improvement is due to a higher occurrence of critical features of the input question in the retrieved document collection. Figure 1 showcase three example questions with their corresponding occurrence of features in the retrieved document collection. We observe FetcHR to particular improve on the critical feature.

Table 3 provides a broader comparison of the performance of FetcHR to prior work on QA tasks. We consider prior work where retriever and reader are trained separately from the same or a strong overlapping dataset as with FetcHR. We find that FetcHR outperforms prior work when the FiD reader is finetuned on the FetcHR output distribution by 1.9 EM score on Natural Question and 2.1 EM score on Web Question.

Closed-book QA Models	NQ	WebQ
T5-base (Roberts et al., 2020)	25.7	28.2
T5-large (Roberts et al., 2020)	27.3	29.5
T5-XXL (Roberts et al., 2020)	32.8	35.6
GPT-3 (Brown et al., 2020)	29.9	41.5
Open-book QA Models	**NQ**	**WebQ**
BM25+BERT (Lee et al., 2019)	26.5	21.3
QRQA (Lee et al., 2019)	33.3	30.1
DPR (Karpukhin et al., 2020)	41.5	42.4
ReConsider-base (Iyer et al., 2020)	43.1	44.4
ReConsider-large (Iyer et al., 2020)	44.5	45.9
RETRO 7.5B w. DPR (Borgeaud et al., 2021)	45.5	-
FiD-base (Izacard and Grave, 2021)	48.2	-
FiD-large (Izacard and Grave, 2021)	51.4	-
FetcHR-flat / FiD-large trained	**53.3**	**48.0**

Table 3: EM scores of related work compared to our results on the Natural Question and WebQuestion test sets.

4.6 Ablations

Importance of FetcHR layers We investigate the importance of individual FetchHR layers when retrieving simultaneously from all layers. If some layers retrieve better documents than others, then we may consider the opportunity of unbalancing the contribution of different layers, i.e. letting those layers retrieve more documents than the others. We analyse individual layer performance by measuring the averaged amount of documents containing the answer span each layer retrieves additional to previous layers – we call this support in the following. Note that this support is different to Figure 3, where retrieval is performed in single layers while here retrieval is done in all layers. The results are illustrated in Figure 4. In this case, $k = 10$ and the total budget is $L \times k = 120$. The majority of correct documents is retrieved in the first layer but we observe that each layer has a significant contribution to the final retrieval performance showing the benefit of retrieving in multiple layers. We also observe that the middle layers have the lowest support. This low support could be a consequence of having a strong overlap of retrieved documents to previous layers sides while the first and last layer retrieve very different documents due to the largest difference in the hidden state representation of the input question.

Distribution of retrieval budget over multiple layers Given the results of Figure 4, we investigate alternative ways of dividing the total budget of documents $L \times k$ among the FetcHR layers, different from distributing the budget uniformly. Table 4 provides an overview of the retrieval accuracy in different configurations ranging from retrieval in the last layer only, the first and last layer, up to all layers. Note that the total amount of retrieved documents is 120 for all experiments. From these results, we conclude that the best configuration is when retrieval is distributed over all layers equally (i.e. $L = 12, k = 10$).

5 Conclusion

5.1 Summary

We presented retrieval-augmented transformers, a multi-encoder retrieval system exploiting different hierarchical abstractions of a model input. Our experiments show a competitive retrieval performance and a superior reader performance for two benchmark tasks on the FiD reader.

Since FetcHR is a retrieval system, it does not generate answers by itself and it requires a reader that can process the retrieved documents. We found that the FiD reader, in large configuration, is able to process retrieved documents efficiently – without additional training. Additional training of the FiD reader on the output distribution of FetcHR improved the performance even further and outperforms related work.

5.2 Discussion and Future Work

Our work shows that in the scenario of an end-to-end question answering task, a high retrieval

FetcHR layers	k per layer	accuracy
12	120	81.3
1, 12	60	82.6
1, 6, 12	40	83.1
1, 2, 3, 10, 11, 12	20	83.4
2, 4, 6, 8, 10, 12	20	83.5
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12	10	83.9

Table 4: Retrieval accuracy on Natural Question with varying distributions of L and k for a retrieval budget. Note that the total budget $L \times k = 120$ is kept constant across different rows of the table.

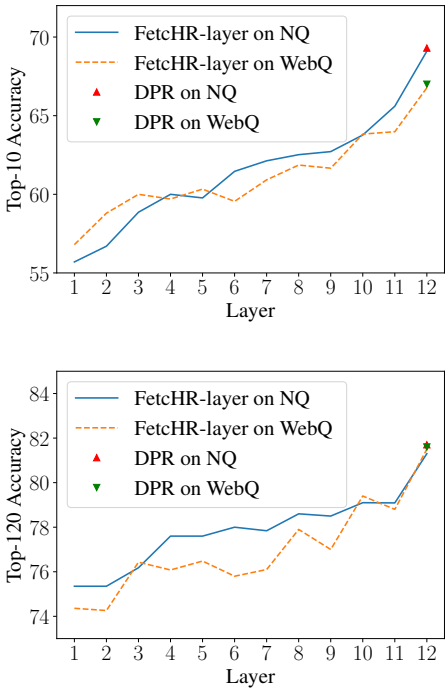

Figure 3: Top-10 (top) and top-120 (bottom) retrieval accuracy when all documents are retrieved in a single FetcHR layer compared to DPR. We evaluated this performance on the test set of Natural Questions. The accuracy is measured as percentage of top-k retrieved documents containing the correct answer span.

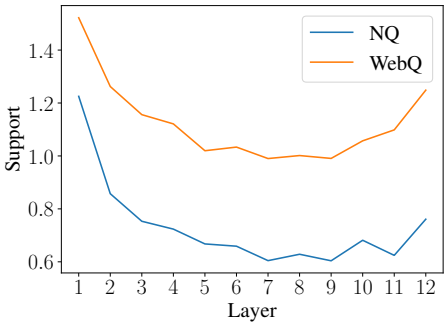

Figure 4: Support of each FetcHR layer to the final retrieval performance.

accuracy does not always translate to a high EM score of the reader output. We observe a better EM score of the reader despite an equal/slightly worse retrieval accuracy of the retriever. This appears to be contra-intuitive. However, a generative reader such as FiD performs inference over all retrieved documents at the same time. Typically, the answer to a question appears multiple times within the retrieved document collection. We believe this document distribution to be better, when the critical features of questions occur more often (see Figure 1). This might lead to future work aiming to develop automated metrics for retrieval systems when end-to-end question answering is the goal.

Another interesting observation is that FetcHR obtains the largest improvement in many scenarios on a compressed IVF index. We believe this is influenced by a wider exploration of the compressed search space, in addition to the hierarchical retrieval. This wider exploration might be a consequence of different queries from the individual FetcHR layers.

FetcHR demonstrated that a well performing retriever can be obtained with a query encoder build from just a few transformer-layers. With just a single transformer-layer and an attached retrieval network, we obtained decent retrieval performance. In the future, this may allow more hardware-efficient inference, shifting computational needs from the transformer network to the nearest neighbour search method. In future work, end-to-end training methods of FetcHR with FiD might lead to a fusion language model with both parametric and non-parametric knowledge over multiple layers. We believe this will have a significant impact on knowledge-intense tasks.

6 Limitations

Despite being trainable and usable for most datasets and document memories, there are some limitations to consider. The first one is related to the retrieval index. The discussed flat retrieval index scales poorly to large document memories. Despite being commonly used in research publications, the practical application of flat indices is limited due to long inference times on large document memories. Due to the poor scaling of the flat index, we also presented results on the much fast and more scalable IVFIndex. Another limitation is related to the DPR retriever as initial checkpoint for the retriever. We found very good retrieval results when FetcHR is trained on one of the pretraining datasets for DPR, however we observed a performance drop when FetcHR is used on a novel dataset. As this drop is expected for most models when train and test data distribution are not matching, a solution to this is an additional training of DPR following the DPR approach on the new dataset first. Afterwards, the presented FetcHR training can be conducted.

Ethics Statement

FetcHR shares the same ethical considerations and societal impact as prior work on language models and retrieval systems. Even though FetcHR improves performance on knowledge-intense tasks, it inherits the bias given by the training data and the collection of documents in the memory. This bias might lead to unfair or misleading model outputs. Since FetcHR does not have an explicit mechanism to detect and prevent a manipulated document memory, it could get prone to retrieve documents containing fake knowledge.

References

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*, pages 222–228. Springer.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2021. Improving language models by retrieving from trillions of tokens. *arXiv preprint arXiv:2112.04426*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*, pages 1870–1879. Association for Computational Linguistics (ACL).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Utku Evci, Vincent Dumoulin, Hugo Larochelle, and Michael C Mozer. 2022. Head2toe: Utilizing intermediate representations for better transfer learning. In *International Conference on Machine Learning*, pages 6009–6033. PMLR.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, pages 3929–3938. PMLR.

Srinivasan Iyer, Sewon Min, Yashar Mehdad, and Wentau Yih. 2020. Reconsider: Re-ranking using span-focused cross-attention for open domain question answering. *arXiv preprint arXiv:2010.10757*.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *EACL 2021-16th Conference of the European Chapter of the Association for Computational Linguistics*, pages 874–880. Association for Computational Linguistics.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781.

Omar Khattab and Matei Zaharia. 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*, pages 39–48.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345.

Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. A discrete hard em approach for weakly supervised question answering. *arXiv preprint arXiv:1909.04849*.

Yifan Nie, Yanling Li, and Jian-Yun Nie. 2018a. Empirical study of multi-level convolution models for ir based on representations and interactions. *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval*.

Yifan Nie, Alessandro Sordoni, and Jian-Yun Nie. 2018b. Multi-level abstraction convolutional model with weak supervision for information retrieval. *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*.

Yixin Nie, Songhe Wang, and Mohit Bansal. 2019. Revealing the importance of semantic retrieval for machine reading at scale. 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)*, pages 2553–2566.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5835–5847.

Adam Roberts, Colin Raffel, and Noam Shazeer. 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)*, pages 5418–5426.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Devendra Singh, Siva Reddy, Will Hamilton, Chris Dyer, and Dani Yogatama. 2021. End-to-end training of multi-document reader and retriever for open-domain question answering. *Advances in Neural Information Processing Systems*, 34.

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

Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. 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)*, pages 5878–5882.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *Transactions of the Association for Computational Linguistics*, 8:183–198.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with bertserini. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 72–77.

Wen-tau Yih, Kristina Toutanova, John C Platt, and Christopher Meek. 2011. Learning discriminative projections for text similarity measures. In *Proceedings of the fifteenth conference on computational natural language learning*, pages 247–256.

Shun Zhang, Yaobo Liang, Ming Gong, Daxin Jiang, and Nan Duan. 2022. Multi-view document representation learning for open-domain dense retrieval. In *ACL*.

A Datasets

The datasets used in this work are open-source and widely used in the community. We make use of a preprocessed and published version by (Karpukhin et al., 2020) which can be downloaded from here: https://github.com/facebookresearch/DPR. In Table 5 the statistics of these datasets can be found.

Natural Question (Kwiatkowski et al., 2019)
URL: https://ai.google.com/research/NaturalQuestions/download
License: https://github.com/google-research-datasets/natural-questions/blob/master/LICENSE

WebQuestions (Berant et al., 2013)
URL: https://github.com/google-research/language/tree/master/language/orqa#getting-the-data
License: https://nlp.stanford.edu/software/sempre/

TriviaQA (Joshi et al., 2017)
URL: http://nlp.cs.washington.edu/triviaqa/
License: https://github.com/mandarjoshi90/triviaqa/blob/master/LICENSE

CuratedTrec (Baudiš and Šedivỳ, 2015)
URL: https://github.com/brmson/dataset-factoid-curated
License: https://github.com/brmson/dataset-factoid-curated/tree/master/trec

Dataset	Filtered Train	Development	Test
Natural Questions	58,880	8,758	3,610
WebQuestions	2,474	8,837	2,032
TriviaQA	60,413	361	11,313
CuratedTREC	1,125	133	694

Table 5: Datasets used in this work.