Journal of Machine Learning Research 24 (2023) 1-43 Submitted 1/23; Revised 7/23; Published 7/23

Atlas: Few-shot Learning with Retrieval Augmented Language Models

Gautier Izacard[1,2,*,†] GAUTIER@INFLECTION.AI
Patrick Lewis[1,*,†] PATRICK@COHERE.COM
Maria Lomeli[1] MARIALOMELI@META.COM
Lucas Hosseini[1,†] HOSS@META.COM
Fabio Petroni[1,†] FABIOPETRONI@META.COM
Timo Schick[1,†] SCHICK@META.COM
Jane Dwivedi-Yu[1] JANEYU@META.COM
Armand Joulin[1,†] AJOULIN@META.COM
Sebastian Riedel[1,3,†] SRIEDEL@META.COM
Edouard Grave[1,†] EGRAVE@META.COM
[1] *Meta AI,* [2] *ENS, PSL University & Inria,* [3] *University College London*

Editor: Ivan Titov

Abstract

Large language models have shown impressive few-shot results on a wide range of tasks. However, when knowledge is key for such results, as is the case for tasks such as question answering and fact checking, massive parameter counts to store knowledge seem to be needed. Retrieval-augmented models are known to excel at knowledge intensive tasks without the need for as many parameters, but it is unclear whether they work in few-shot settings. In this work we present Atlas, a carefully designed and pre-trained retrieval-augmented language model able to learn knowledge intensive tasks with very few training examples. We perform evaluations on a wide range of tasks, including MMLU, KILT and Natural Questions, and study the impact of the content of the document index, showing that it can easily be updated. Notably, Atlas reaches over 42% accuracy on Natural Questions using only 64 examples, outperforming a 540B parameter model by 3% despite having 50x fewer parameters.

Keywords: retrieval augmented language models, information retrieval, language models

1. Introduction

Large language models (LLMs) are impressive few-shot learners (Brown et al., 2020; Rae et al., 2021; Hoffmann et al., 2022; Chowdhery et al., 2022). They are able to learn new tasks with very few examples or even from instructions alone. For this generalisation ability to emerge, the key ingredients are scaling both the parameter count of the model, and the size of the training data. Large language models owe this improvement to both a larger computational budget, enabling more complex reasoning, and the ability to memorize more

∗. Equal contribution
†. Work done while at Meta AI

information related to downstream tasks from the larger training data. While it is intuitive to assume that increased reasoning abilities lead to better generalisation, and hence few-shot learning, the same is not true for in-parameter memorisation. Specifically, it is unclear to what extent effective few-shot learning requires vast knowledge in the parameters of the model.

In this paper, we investigate whether few-shot learning requires models to store a large amount of information in their parameters, and if memorisation can be decoupled from generalisation. To do so, we leverage the fact that memory can be outsourced and replaced by an external non-parametric knowledge source by employing a *retrieval-augmented* architecture. These models employ a non-parametric memory, for example a neural retriever over a large, external, potentially non-static knowledge source to enhance a parametric language model. In addition to their memorisation abilities, such architectures are attractive due to a number of other established advantages in terms of adaptability, interpretability and efficiency (Guu et al., 2020; Lewis et al., 2020; Yogatama et al., 2021; Borgeaud et al., 2021, inter alia). However, retrieval-augmented models have yet to demonstrate compelling few-shot learning capabilities. In this work we address this gap, and present Atlas, a retrieval-augmented language model capable of strong few-shot learning, despite having lower parameter counts than other powerful recent few-shot learners.

Atlas retrieves relevant documents based on the current context by using a general-purpose dense retriever using a dual-encoder architecture, based on the Contriever (Izacard et al., 2022). The retrieved documents are processed, along with the current context, by a sequence-to-sequence model using the Fusion-in-Decoder architecture (Izacard and Grave, 2021a) that generates the corresponding output. We study the impact of different techniques to train Atlas on its few-shot performance on a range of downstream tasks, including question answering and fact checking. We find that jointly pre-training the components is crucial for few-shot performance, and we carefully evaluate a number of existing and novel pre-training tasks and schemes for this purpose. Atlas achieves strong downstream performance in both few-shot and resource-rich settings. For example, with only 11B parameters, Atlas achieves an accuracy of 42.4% on Natural Questions using 64 training examples (45.1% using a Wikipedia-only index), outperforming PaLM (Chowdhery et al., 2022), a 540B parameter model by almost 3 points, and 64.0% in a full data set setting with a Wikipedia index, establishing a new state of the art by 8.1 points.

In summary we make the following contributions:

- A thorough study on how to design and train retrieval-augmented language models, with a focus on downstream few-shot learning and sample efficiency.
- The findings of this study lead to a retrieval-augmented language model, called Atlas, that exhibits few-shot abilities that emerge at lower scale than standard LLM.
- We provide an exploration of fine-tuning strategies to efficiently adapt both the retriever and the language model to the task at hand.
- Thorough downstream experiments in few-shot settings, demonstrating state-of-the-art results on few-shot Natural Questions (+2.8%), TriviaQA (+3.3%), FEVER (+5.1%), and results on par with models with 15× more parameters on MMLU.

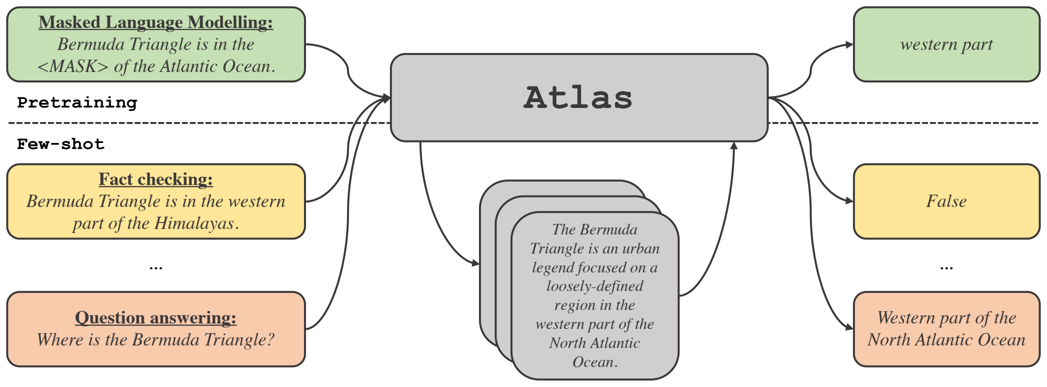
Masked Language Modelling:
Bermuda Triangle is in the <MASK> of the Atlantic Ocean.
Pretraining
Few-shot
Fact checking:
Bermuda Triangle is in the western part of the Himalayas.
...
Question answering:
Where is the Bermuda Triangle?
Atlas
The Bermuda Triangle is an urban legend focused on a loosely-defined region in the western part of the North Atlantic Ocean.
western part
False
...
Western part of the North Atlantic Ocean

Figure 1: We introduce Atlas, a retrieval-augmented language model that exhibits strong few-shot performance on knowledge tasks, and uses retrieval during both pre-training and fine-tuning.

- Experiments investigating full data set finetuning, setting new state-of-the-art results in Natural Questions (+8.1%), TriviaQA (+9.3%) and 5 KILT Tasks.
- Experiments demonstrating the updateability and interpretability characteristics of Atlas.
- Experiments demonstrating that a compressed index using product quantisation achieves comparable performance as an uncompressed index while resulting in a 5x memory reduction.

Our code, pre-trained Atlas checkpoints, and various supporting data are available at `https://github.com/facebookresearch/atlas`

2. Method

Our approach follows the *text-to-text* framework (Raffel et al., 2019). This means that all the tasks are framed as follows: the system gets a *text query* as input, and generates a *text output*. For example, in the case of question answering, the query corresponds to the question and the model needs to generate the answer. In the case of classification tasks, the query corresponds to the textual input, and the model generates the lexicalized class label, that is the word corresponding to the label. We give more examples of downstream tasks, from the KILT benchmark in Figure 2. As many natural language processing tasks require *knowledge*, our goal is to enhance standard text-to-text models with retrieval, which, as we hypothesise in the introduction, may be crucial to endow models with few-shot capabilities.

2.1 Architecture

Our model is based on two sub-models: the *retriever* and the *language model*. When performing a task, from question answering to generating Wikipedia articles, our model starts by retrieving the top-k relevant documents from a *large corpus* of text with the retriever.

Task	**Query**	**Output**
Fact Checking	*Bermuda Triangle is in the western part of the Himalayas.*	*False*
Question Answering	*who is playing the halftime show at super bowl 2016*	*Coldplay*
Entity Linking	*NTFS-3G is an open source <E>cross-platform</E> implementation of the Microsoft Windows NTFS file system with read-write support.*	*Cross-platform software*

Figure 2: Examples of query and output pairs for different tasks from KILT.

Then, these documents are fed to the language model, along with the query, which in turns generates the output. Both the retriever and the language model are based on pre-trained transformer networks, which we describe in more detail below.

2.1.1 Retriever

Our retriever module is based on the Contriever (Izacard et al., 2022), an information retrieval technique based on continuous dense embeddings. The Contriever uses a dual-encoder architecture, where the query and documents are embedded independently by a transformer encoder (Huang et al., 2013; Karpukhin et al., 2020). Average pooling is applied over the outputs of the last layer to obtain one vector representation per query or document. A similarity score between the query and each document is then obtained by computing the dot product between their corresponding embeddings. The Contriever model is pre-trained using the MoCo contrastive loss (He et al., 2020), and uses unsupervised data only. As shown in the following section, an advantage of dense retrievers is that both query and document encoders can be trained without document annotation, using standard techniques such as gradient descent and distillation.

2.1.2 Language Model

For the language model, we rely on the T5 sequence-to-sequence architecture (Raffel et al., 2019). We rely on the Fusion-in-Decoder modification of sequence-to-sequence models, and process each document independently in the encoder (Izacard and Grave, 2021a). We then concatenate the outputs of the encoder corresponding to the different documents, and perform cross-attention over this single sequence in the decoder. Following Izacard and Grave (2021a), we concatenate the query to each document in the encoder. Another way to process the retrieved documents in the language model would be to concatenate the query and all the documents, and to use this long sequence as input of the model. Unfortunately, this approach does not scale with the number of documents, since the self-attention in the encoder results in a quadratic complexity with respect to the number of documents.

2.2 Training Objectives for the Retriever

In this section, we discuss four different loss functions to train the retriever jointly with the language model. We consider loss functions that leverage the language model to provide *supervisory signal* to train the retriever. In other words, if the language model finds a document useful when generating the output, the retriever objective should encourage the retriever to rank said document higher. This allows us to train models using only query and output pairs from the task of interest, without relying on document annotations. For example, in the case of fact checking, a model only requires pairs of claims and corresponding verdicts but no documents containing the evidence to back up the verdict. In practice, we can apply this approach on any task, including self-supervised pre-training. As shown in the experimental section, pre-training is critical for obtaining models that exhibit few-shot learning abilities.

2.2.1 Attention Distillation (ADist)

The first loss that we consider is based on the attention scores of the language model, and is heavily inspired by Izacard and Grave (2021b). The main idea is that the cross-attention scores between the input documents and the generation can be used as a proxy of the importance of each input document when generating the output. In particular, Izacard and Grave (2021b) showed that these scores can be aggregated across attention heads, layers and tokens for a given document to obtain a single score for each document. Then, these scores are distilled into the retriever by minimizing the KL-divergence with the probability distribution p_{RETR} over the top-K documents $\{\mathbf{d}_k\}_{1,...,K}$ obtained from the retriever:

$$p_{\text{RETR}}(\mathbf{d} \mid \mathbf{q}) = \frac{\exp(s(\mathbf{d}, \mathbf{q})/\theta)}{\sum_{k=1}^{K} \exp(s(\mathbf{d}_k, \mathbf{q})/\theta)}, \tag{1}$$

where s is the dot-product between the embedding vectors of the query and documents and θ is a temperature hyperparameter.

In the original paper, to obtain a relevance score per document it was proposed to use the pre-softmax scores from the decoder cross-attentions, and average across heads, layers and tokens. Here, we use the pre-softmax score multiplied by the norm of the values, an alternative which gives slightly stronger results. First, let us briefly review the Fusion-in-Decoder model (FiD, Izacard and Grave, 2021a). The underlying architecture is a sequence-to-sequence model, composed of an encoder and a decoder. The encoder independently processes K different text inputs $(\text{INPUT}(\mathbf{d}_k))_{1 \leq k \leq K}$, where $\text{INPUT}(\mathbf{d})$ is the concatenation of the input query and the retrieved document $\mathbf{d}$. The output representations of the encoder are then concatenated to form a global representation $\mathbf{X}$ of dimension $(\sum_k \ell_k) \times d$, where ℓ_k is the length of $\text{INPUT}(\mathbf{d}_k)$ and d is the dimension of the hidden representations of the model. Then, the decoder processes this representation as a regular autoregressive model, alternating self-attention, cross-attention and feed-forward modules.

Only the cross-attention module explicitly takes as input the global output representation $\mathbf{X}$ of the encoder. If $\mathbf{H} \in \mathbb{R}^d$ denotes the output of the previous self-attention layer of the decoder, the cross-attention operation consists in the following operations. First, queries $\mathbf{Q}$, keys $\mathbf{K}$ and values $\mathbf{V}$ are computed by applying linear transformations:

$$\mathbf{Q} = \mathbf{W}_Q\mathbf{H}, \quad \mathbf{K} = \mathbf{W}_K\mathbf{X}, \quad \mathbf{V} = \mathbf{W}_V\mathbf{X}.$$

Then a similarity score between the query at position i, $\mathbf{Q}_i$, and the key at position j, $\mathbf{K}_j$, is obtained by computing the dot-product between these two elements, and normalized over the dimension:

$$\alpha_{i,j} = \mathbf{Q}_i^T \mathbf{K}_j, \qquad \tilde{\alpha}_{i,j} = \frac{\exp(\alpha_{i,j})}{\sum_m \exp(\alpha_{i,m})}.$$

A new representation is obtained as a sum of the values, weighted by the attention probabilities, before going through a final linear transformation $\mathbf{W}_o$:

$$\mathbf{O}_i = \mathbf{W}_O \sum_j \tilde{\alpha}_{i,j} \mathbf{V}_{i,j}.$$

This describes the single-head attention case, in the case of multi-head attention with n_h heads, the output of the cross-attention layer can be written as:

$$\mathbf{O}_i = \sum_{h=1}^{n_h} \mathbf{W}_{O,h} \sum_j \tilde{\alpha}_{h,i,j} \mathbf{V}_{j,h}.$$

For the layer l and the head h, we use the quantity $\tilde{\alpha}_{l,h,i,j} \|\mathbf{V}_{l,h,j}\|_2$ as the measure of relevance for the input token at position j relatively to the generated token at position i. We average these scores over all attention heads, layers, tokens of the generation and tokens of the input segment $\textsc{input}(\mathbf{d})$ to obtain an attention score $\textsc{score}_{\textsc{attn}}(\mathbf{d})$ for each document $\mathbf{d}$:

$$\textsc{score}_{\textsc{attn}}(\mathbf{d}) = \operatorname*{mean}_{h,l,i,j \in \textsc{input}_k} \alpha_{l,h,i,j} \|\mathbf{V}_{l,h,j}\|_2.$$

We apply the Softmax operator over the resulting scores, to obtain a distribution $p_{\textsc{attn}}(\mathbf{d})$ over the top-K retrieved documents:

$$p_{\textsc{attn}}(\mathbf{d}) = \frac{\exp\left(\textsc{score}_{\textsc{attn}}(\mathbf{d})\right)}{\sum_k \exp\left(\textsc{score}_{\textsc{attn}}(\mathbf{d}_k)\right)}.$$

We then minimize the KL-divergence between $p_{\textsc{attn}}$, and the distribution $p_{\textsc{retr}}$ from the retriever defined in Equation 1:

$$\mathrm{KL}(p_{\textsc{attn}} \parallel p_{\textsc{retr}}) = \sum_{k=1}^{K} p_{\textsc{attn}}(\mathbf{d}_k) \log\left(\frac{p_{\textsc{attn}}(\mathbf{d}_k)}{p_{\textsc{retr}}(\mathbf{d}_k)}\right).$$

Here, this loss is only used to optimize the parameters of the retriever, and not the language model. When using recent deep learning frameworks, this is achieved by applying a StopGradient operator on $p_{\textsc{attn}}$.

2.2.2 End-to-end Training of Multi-Document Reader and Retriever (EMDR^2)

Next, we consider the method introduced by Sachan et al. (2021), which is inspired by the expectation-maximization algorithm, treating retrieved documents as latent variables. Given a query $\mathbf{q}$, the corresponding output $\mathbf{a}$ and the set $\mathcal{D}_K$ of top-K retrieved documents with the current retriever, the EMDR^2 loss to train the retriever is

$$-\log\left[\sum_{k=1}^{K} p_{\text{LM}}(\mathbf{a} \mid \mathbf{q}, \mathbf{d}_k) p_{\text{RETR}}(\mathbf{d}_k \mid \mathbf{q})\right],$$

where p_{RETR} is again the probability over the top-K documents obtained with the retriever, as defined by Equation 1. Again, only the parameters of the retriever are updated by applying a StopGradient operator around p_{LM}. One should note that the probability distribution over documents that minimizes this loss function is an indicator of the document corresponding to the highest probability of the output according to the language model. Finally, in practice, the EMDR^2 loss function is applied at the token level, and not at the sequence level.

2.2.3 Likelihood Distillation (LDist)

Third, we discuss a simpler loss function which is inspired by the objectives from the attention distillation and EMDR^2 methods (Izacard and Grave, 2021b; Sachan et al., 2021). More precisely, we want to train the retriever to predict how much each document would improve the ability of the language model to predict the output, given the query. To this end, we minimize the KL-divergence between the documents distribution of the retriever (Eqn. 1), and the documents posterior distribution according to the language model conditioned on a single document and using a uniform prior:

$$p_{\text{LDist}}(\mathbf{d}_k) \propto p_{LM}(\mathbf{a} \mid \mathbf{d}_k, \mathbf{q}).$$

Using the Softmax operator, we have that

$$p_{\text{LDist}}(\mathbf{d}_k) = \frac{\exp(\log p_{LM}(\mathbf{a} \mid \mathbf{d}_k, \mathbf{q}))}{\sum_{i=1}^{K} \exp(\log p_{LM}(\mathbf{a} \mid \mathbf{d}_i, \mathbf{q}))}.$$

2.2.4 Leave-one-out Likelihood Distillation (LOOL)

Finally, we propose an objective based on how much *worse* the prediction of the language model gets when *removing* one of the top-k retrieved documents. To do so, we compute the log probability of the output for each subset of k-1 documents, and use the negative value as relevance score for each document. Following the previous loss function, we use the softmax operator to obtain a probability distribution over documents:

$$p_{\text{LOOL}}(\mathbf{d}_k) = \frac{\exp(-\log p_{LM}(\mathbf{a} \mid \mathcal{D}_K \setminus \{\mathbf{d}_k\}, \mathbf{q}))}{\sum_{i=1}^{K} \exp(-\log p_{LM}(\mathbf{a} \mid \mathcal{D}_K \setminus \{\mathbf{d}_i\}, \mathbf{q}))}.$$

As before, we then minimize the KL-divergence between this distribution, and the one obtained with retriever. This loss is more expensive to compute than LDist and EMDR, but, like ADist, employs the language model more closely to the way it is trained: the LM is trained to be conditioned on a set of K documents. For LOOL, the language model is conditioned on $(K-1)$ documents, rather than a single document as in EMDR^2 and LDist.

For all losses, we can also use a temperature hyperparameter when computing the target or retriever distributions to control the distribution's peakiness of, which might be important for some tasks or losses. Indeed, for LDist and LOOL, the likelihood of the output may not vary much when conditioning on different documents, especially in the case of long outputs.

2.3 Pretext Tasks

In this section, we describe pretext tasks that can be used to jointly pre-train the retriever and the language model using only unsupervised data.

2.3.1 Prefix Language Modeling

First, we consider a standard language modeling task as a potential pre-training objective. To cast language modeling in the text-to-text framework, we consider a chunk of N words, and split this chunk in two sub-sequences of equal length $N/2$. Then, the first sub-sequence is used as the query, and the second corresponds to the output. We thus retrieve relevant documents by using the first sub-sequence of $N/2$ tokens, to generate the output.

2.3.2 Masked Language Modeling

Second, we consider masked language modeling, as formulated by Raffel et al. (2019). Again, starting from a chunk of N words, we sample k spans of average length 3 tokens, leading to a masking ratio of 15%. We then replace each span by a different special token. The model is then trained to generate the masked spans, each span beginning with the special sentinel mask token that was inserted in the input sequence. We retrieve documents using the masked query, but replace the special mask tokens with a mask token supported by the retriever vocabulary.

2.3.3 Title to Section Generation

Finally, we consider a more abstractive generation task, generating sections from Wikipedia articles, given the article and section title. Here, the query corresponds to the title of the article, together with the title of the section, and the output corresponds to the text of the section. We exclude sections “See also”, “References”, “Further reading” and “External links”.

2.4 Efficient Retriever Fine-tuning

Retrieval is facilitated by using a document *index*, which is a pre-computed collection of the document embeddings for all the documents in the retrieval corpus. When jointly training the retriever and language model, the index needs to be updated regularly, otherwise, the embeddings of the documents stored in the index become stale relative to the updated retriever. This means that we need to recompute the embeddings for the full collection of documents regularly during training to keep the index fresh, which can be computationally expensive for large indices. This is particularly true at *fine-tuning* time, where the number of training examples could be small relative to the number of documents in the index. Training the retriever could thus add an important computational overhead compared to standard language model finetuning. In this section, we analyse strategies that might make this process more efficient, alleviating the need to re-compute the embeddings of all the documents too often.

2.4.1 Full Index Update

Let us start by analysing the overhead due to updating the index compared to using a fixed retriever. To compare the computation time of different models, we will make the following assumption: the time required to perform a forward pass on a document with a model of P parameters is $O(P)$. While this computation model may seem naive, the main assumption is that document sizes are constant.[1] Since we split long documents into passages with similar number of words, and use padding when processing documents of different sizes, this assumption is reasonable in practice. Let K be the number of documents that are retrieved and processed by the language model, P_{LM} be the number of parameters of the language model and B the batch size. Each training step has a complexity of $4 \times B \times K \times P_{\text{LM}}$.[2]

Next, let N be the number of documents in the index, and P_{RETR} be the number of parameters of the retriever. Then, re-computing the full index has a complexity of $N \times P_{\text{RETR}}$. If we refresh the index every R training steps, we obtain the following overhead:

$$\frac{N \times P_{\text{RETR}}}{4 \times B \times K \times P_{\text{LM}} \times R}.$$

If we use the BERT base architecture for our retriever and T5-XL for our language model, we get $\frac{P_{\text{RETR}}}{P_{\text{LM}}} \approx \frac{1}{25}$, leading to the overhead:

$$\frac{N}{100 \times B \times K \times R}.$$

If we use an index containing $37M$ documents (the size of our Wikipedia index), train with a batch size of 64 with 20 retrieved documents and refresh the index every 1000 steps, this results in an overhead of $\sim 30\%$.

2.4.2 Re-ranking

A second strategy is to retrieve a larger number of documents L with the retriever, and to re-embed and rerank these documents with the up-to-date retriever, and pass the resulting top-K to the language model. In that case, the overhead of reranking the top-L documents is equal to $B \times L \times P_{\text{RETR}}$. Since we perform this operation at every time step, the overhead is equal to

$$\frac{L \times P_{\text{RETR}}}{4 \times K \times P_{\text{LM}}}.$$

Using the same assumption as before, we finally get that the overhead is of the order of $\frac{L}{100 \times K}$. If we re-rank 10x more documents than what the language model processes (that is $L = 10 \times K$), we get an overhead of 10%. However, note that if many updates are performed on the retriever, the index might still need to be fully updated, as the true top-k documents may not be retrieved in the top-L results from the stale index. In practice, it is possible to track the positions of the top-K re-ranked documents in the top-L, and estimate when the index needs to be updated.

1. See Hoffmann et al. (2022) for more details about the computation of the FLOPS corresponding to the forward and backward passes of transformer networks.
2. There is a factor 4 to account for the backward pass and activation checkpointing.

2.4.3 Query-side Fine-tuning

Finally, the last strategy is to decouple the encoding of the queries and documents as done in Guu et al. (2020). In this case, we fix the parameters corresponding to the document encoder, and only train the parameters corresponding to the query encoder. Thus, the embeddings of documents are fixed, and we do not need to refresh the index, and thus there is no computational overhead. As we will see in practice, the impact of fixing the documents encoder varies greatly for different tasks when a large training data set is available. For most of the few-shot settings that we consider, query-side finetuning does not have large performance impact, and sometimes even slightly improves performance.

3. Related Work

In this section we first review the literature on retrieval in language models, before giving an overview on few-shot learning in natural language processing.

3.1 Retrieval-augmented models in Natural Language Processing

There has been a long line of work studying the effect and potential benefits of retrieval augmentation for NLP tasks.

3.1.1 Retrieval for Knowledge Intensive Tasks

Previous work has shown that retrieval improves performance across a variety of tasks such as question answering (Voorhees, 1999; Chen et al., 2017; Kwiatkowski et al., 2019), fact checking (Thorne et al., 2018), dialogue (Dinan et al., 2019) or citation recommendation (Petroni et al., 2022). Historically, this information retrieval step was implemented using term-matching methods, such as TF-IDF or BM25 (Jones, 1972; Robertson et al., 1995). For open-domain question answering (Voorhees, 1999), documents are often retrieved from Wikipedia (Chen et al., 2017). Recently, dense retrievers based on neural networks have become popular. These usually follow a dual-encoder architecture (Yih et al., 2011; Huang et al., 2013; Shen et al., 2014), where queries and passages are encoded independently as vectors, and relevance is computed using the inner product or Euclidean distance. Popular supervised retrievers include DPR (Karpukhin et al., 2020), which is trained to discriminate the relevant passage among negative passages, and extensions such as ANCE (Xiong et al., 2021) which improved the hard negatives mining process. We refer the reader to Yates et al. (2021) for a survey of dense retrieval techniques.

After retrieval, the relevant documents are processed to produce the final output. In open-domain QA, models can extract a span of text from retrieved documents as the answer (Chen et al., 2017; Clark and Gardner, 2018; Wang et al., 2019; Karpukhin et al., 2020), a method inspired by reading comprehension (Richardson, 2013; Rajpurkar et al., 2016). Recently, generating the answer as free-form text, using a seq2seq model conditioned on retrieved documents have become prevalent (Lewis et al., 2020; Izacard and Grave, 2021a; Min et al., 2020). These architectures have also been shown to reduce hallucination in dialogue agents (Shuster et al., 2021).

3.1.2 Retriever training

The need for expensive query-document annotations for training the retriever can be bypassed, by leveraging signals from the language model, or using unsupervised learning. REALM (Guu et al., 2020) and RAG (Lewis et al., 2020) jointly train the retriever and language model by modelling documents as latent variable, and minimizing the objective with gradient descent. REALM pre-trains end-to-end with an MLM approach but uses an extractive BERT-style model (Devlin et al., 2019). Guu et al. (2020) also explore a query-side finetuning at finetuning time to avoid index refreshes, which is also explored in the context of phrase-based retrieval by Lee et al. (2021b). Izacard and Grave (2021a) proposed to use cross-attention scores as supervision with knowledge distillation. Sachan et al. (2021) perform joint training of the reader and the retriever by leveraging the likelihood of the output generated by the reader. Sachan et al. (2021) and Lee et al. (2021a) both employ salient span masking to pre-train retrievers, leveraging the likelihood and attention scores from the language model. The *inverse cloze task* was proposed by Lee et al. (2019) to pre-train dense retrievers in an unsupervised way. Paranjape et al. (2021) propose a method to train retrieval-augmented generators using a second "informed" retriever with access to the output, which the test-time retriever can be distilled from, and Hofstätter et al. (2022) recently proposed a training set filtering/weighting approach to train stronger retrieval-augmented generators. Izacard et al. (2022) explored different contrastive learning methods to train retrievers, while Ram et al. (2022) used recurring spans within a document to create pseudo-positive query-document pairs.

3.1.3 Retrieval-augmented language models

Continuous cache models (Grave et al., 2017b) defines a probability distribution over recent tokens, by computing the similarity between previous and current representations of tokens. This distribution is then interpolated with the distribution of the language model, to improve predictions. Later, the amount of tokens used to compute this distribution was extended to a much larger memory by leveraging approximate nearest neighbors search (Grave et al., 2017a). The related kNN-LM model (Khandelwal et al., 2020) replaced LSTMs by transformer networks, and scaled the memory to billions of tokens, leading to strong performance improvements. More recently, RETRO (Borgeaud et al., 2021) extended these by scaling the retrieval memory to trillions of tokens, and changing the model architecture to take retrieved documents as input.

3.1.4 Retrieval-Augmentation with Search Engines

Recently, different works have proposed to train large language models to interact with a search engine, by generating text queries, and using the retrieved documents as additional context (Nakano et al., 2021; Thoppilan et al., 2022; Shuster et al., 2022). In the context of few-shot question answering, Lazaridou et al. (2022) used the question to perform a search query, and retrieved documents are added to the prompt of a large language model performing in-context learning.

3.2 Few-shot Learning

Few-shot learning, the task of learning from very few examples, has been studied for decades (Thrun and Pratt, 1998; Fink, 2005; Vinyals et al., 2016), but has recently seen an explosion of interest in NLP with the arrival of large pre-trained models.

3.2.1 In-context Learning with Large Language Models

Providing language models with natural language descriptions of tasks, as proposed by Radford et al. (2019) has led to significant developments in few-shot learning. GPT-3 (Brown et al., 2020) demonstrated the ability of large language models to perform few-shot predictions, where the model is given a description of the task in natural language with few examples. Scaling model size, data and compute is crucial to enable this learning ability, leading to the further development of large models (Lieber et al., 2021; Rae et al., 2021; Smith et al., 2022; Chowdhery et al., 2022; Smith et al., 2022). Hoffmann et al. (2022) revisited the scaling law from Kaplan et al. (2020), suggesting that training on more data with a smaller model may be more effective, resulting in Chinchilla, a 70B parameter model with improved parameter efficiency.

3.2.2 Few-shot Finetuning and Prompt-based Learning

The above models perform few-shot learning with in-context instructions without training the parameters of the language model. Few-shot learning can also be accomplished by combining textual templates ("prompts") and various forms of model finetuning, either fully updating a model's parameters, for example for classification (Schick and Schütze, 2021a; Schick and Schutze, 2021; Gao et al., 2021; Tam et al., 2021) or generation (Schick and Schütze, 2021b). Prompts themselves can be optimized, for example by search (Jiang et al., 2020; Shin et al., 2020) or by only updating parts of the model (Logan et al., 2021), or learning "soft-prompts" (Lester et al., 2021; Li and Liang, 2021). Due to its simplicity, in this work we either employ simple prompts or simply feed in inputs without preprocessing, and perform full-model finetuning, a method similar to Le Scao and Rush (2021).

4. Experiments

In this section, we report empirical evaluations of our language models on few-shot learning. We start by introducing our experimental setup, describing our evaluation benchmarks in section 4.1, and giving the training details of our models in section 4.2. Then, we perform an ablation study to compare the different technical choices leading to our main model. We finally evaluate this model, called Atlas, on different natural language understanding tasks in few-shot and full data set settings.

4.1 Benchmarks

To evaluate our retrieval-augmented language models we consider the following benchmarks, which include different tasks.

4.1.1 Knowledge-Intensive Language Tasks (KILT)

First, we use the KILT evaluation suite (Petroni et al., 2020), containing 11 data sets corresponding to 5 tasks: fact checking, question answering, dialog generation, entity linking and slot-filling. To be solved, these different tasks require knowledge about the world, which can be found in Wikipedia. We evaluate our model on the following tasks and data sets included in KILT: question answering: Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017) and HotpotQA (Yang et al., 2018); slot filling: Zero Shot RE (Levy et al., 2017) and T-REx (Elsahar et al., 2018); entity linking: AIDA CoNLL-YAGO (Hoffart et al., 2011); dialogue: Wizard of Wikipedia (Dinan et al., 2019); and fact checking: FEVER (Thorne et al., 2018). The KILT versions of these data sets differ from their original versions, as instances requiring knowledge not present in the August 2019 Wikipedia dump have been removed.

4.1.2 Massively-Multitask Language Understanding (MMLU)

Our second main evaluation benchmark is MMLU (Hendrycks et al., 2021), which contains 57 multi-choice question answering data sets (referred to as domains), sourced from real examinations designed for humans. These cover a very broad range of topics, for example high school mathematics, professional law, logical fallacies and clinical knowledge and can be broadly categorized in four subsets: humanities, social sciences, STEM and “other”. We focus on few-shot learning, and the authors of the benchmarks suggest to use 5 training examples per domain. Beyond the 5-shot setting, We also consider three additional settings. The first is a *zero-shot* setting, with no training data at all. The second, which we call *multi-task few-shot*, is where we train a single model on the 5-shot data from all tasks, hence leading to a training set of 285 examples. The last, which we call *transfer learning*, leverages additional training examples from other multiple-choice QA tasks provided by the MMLU authors, namely MCTest (Richardson et al., 2013), RACE (Lai et al., 2017), ARC (Clark et al., 2018) and OBQA (Mihaylov et al., 2018) leading to a training set of 95k examples.

4.1.3 Additional Benchmarks

Additionally, we report results on the original open-domain versions of the popular Natural Questions (Kwiatkowski et al., 2019), and TriviaQA (Joshi et al., 2017) data sets. Generated answers are evaluated with the standard exact match metric (EM), as used by Rajpurkar et al. (2016). A generated answer is considered correct if it matches any answer of the list of acceptable answers after normalization. This normalization step consists in lowercasing and removing articles, punctuation and duplicated whitespaces. We also evaluate our model on the original version of FEVER (Thorne et al., 2018), which presents fact checking as a three-way classification problem for textual claims (either “Supported”: the text is supported by evidence in Wikipedia, “refuted”: the claim is not consistent with evidence in Wikipedia, or “not enough info”, where there is insufficient evidence to make a judgement). We also perform experiments to assess temporal sensitivity of our models. Here, we construct a data set from TempLAMA (Dhingra et al., 2022), consisting of a set of time-sensitive cloze questions on a range of topics, where the answer changes from 2017 to 2020. We assess the accuracy of our models when supplied with a index from 2017 vs 2020 to assess to what degree models faithfully reflect the content of the index supplied to them at test time, and how effective updating the index is as a *continual learning* or model updateability method.

4.2 Technical Details

We now describe the procedure for pre-training and fine-tuning our models. We focus on the setting used for the ablation studies performed in Section 4.3 and Section 4.4. We give more details about the hyperparameters used for our final model later.

4.2.1 Pre-training

For the pre-training, we initialize the retriever module using the unsupervised *Contriever* model (Izacard et al., 2022), which uses the BERT base architecture. We initialize the language model with the T5 pre-trained weight (Raffel et al., 2019). As the original T5 pre-trained model included supervised data in the training set, we use the version 1.1 models which were trained on unlabeled text only. Specifically, we initialize from the `T5-lm-adapt` variants due to their improved stability.

For the ablation studies performed in Section 4.3 and Section 4.4, we use T5-XL which contains 3B weights. We pre-train all our models for 10,000 iterations, using AdamW with a batch size of 64 and a learning rate of 10^{-4} for the reader and 10^{-5} for the retriever with linear decay and 1,000 warmup steps. We refresh the index every 1,000 steps. This means that the index is recomputed 10 times during the pre-training, leading to an overhead of around 30%, compared to training with a fixed retriever. We set the number of retrieved documents to 20. We detail the hyperparameters used for the training of our final model at the beginning of Section 5.

4.2.2 Fine-tuning

When performing a downstream task, either in a few-shot setting or with a large training set, we employ fine-tuning to adapt our models to these tasks. For the few-shot KILT ablation experiments, we perform a fixed number of fine-tuning iterations, instead of using early-stopping. More precisely, we decided to use 50 iterations for the 64-shot setting and 200 iterations in the 1024-shot setting. In both cases, we use a batch size of 32 examples, a learning rate of 4×10^{-5} with linear decay and 5 warmup steps for both the reader and the retriever.

4.2.3 Unlabeled Data Sets

Finally, we discuss the unlabeled text data sets that we use to train our models, which form the retrieval index. First, we consider the Dec. 20, 2021 Wikipedia dump, for which we keep the lists and infoboxes, which are linearized by adding a semi-colon separator between the entries. We split articles by section, and split long sections into passages of equal sizes and containing less than 200 words. This leads to a total of 37M passages, containing 78 words in average. We also use documents from the 2020-10 common crawl dump, preprocessed with the CCNet pipeline (Wenzek et al., 2020). We perform additional document filtering, in a similar fashion to Gopher (Rae et al., 2021). More precisely, we filter documents based on document length, average word length, ratio of alphanumeric characters and number of

		64-shot				1024-shot			
	MLM	NQ	WoW	FEVER	Avg.	NQ	WoW	FEVER	Avg.
Closed-book	1.083	6.5	14.1	59.0	26.5	10.7	16.5	75.3	34.2
No joint pre-training	-	9.0	14.1	67.0	30.0	9.9	16.6	78.3	34.9
Fixed retriever	0.823	39.9	14.3	72.4	42.2	45.3	<u>17.9</u>	90.0	<u>51.1</u>
ADist	<u>0.780</u>	40.9	14.4	73.8	43.0	<u>46.2</u>	17.2	**90.9**	**51.4**
EMDR^2	0.783	<u>43.3</u>	<u>14.6</u>	72.1	43.3	44.9	**18.3**	85.7	49.6
LDist	0.783	**45.0**	**15.0**	**77.0**	**45.7**	44.9	<u>17.9</u>	<u>90.2</u>	51.0
LOOL	**0.766**	41.8	**15.0**	<u>74.4</u>	<u>43.7</u>	**47.1**	<u>17.9</u>	87.5	50.8

Table 1: Loss function ablation. We compare different loss functions to pre-train the retriever jointly with the language model. We compare different loss functions to pre-train the retriever jointly with the language model. We use the prefix MLM task for pre-training. Fine-tuning is performed with query-side fine-tuning and the loss used for pre-training. Best result is bold, second highest underlined.

repeated tokens. This leads to a total of 350M passages. The same passages are used for the index and model pre-training. During pre-training, we ensure the passage we are training on is filtered out from the retrieved documents, to prevent the model from simply retrieving the passage it is de-nosing/generating, and trivially using it to solve the pre-training task.

4.3 Pre-training Loss and Tasks

We start our ablation study by comparing different pre-training tasks and objective functions to jointly train the retriever and the language model. Our goal here is to answer the following research questions:

(RQ 1) Does jointly pre-training the whole model lead to better few-shot performance?

(RQ 2) What is the best objective function for the retriever, and the best pretext task?

We start by comparing the training objectives of the retriever, introduced in Section 2.2, by pre-training models using the masked language modelling task. We evaluate these models on a subset of the 64-shot and 1024-shot KILT benchmark: Natural Questions, FEVER and Wizard of Wikipedia, along with three baselines: a "closed-book" model, a model without joint pre-training, and a model pre-trained with a fixed retriever. The closed-book baseline is a non-retrieval-augmented T5 model, initialized with T5-XL, and further pre-trained on the same data as the other models with the masked language modelling task to ensure that all models are trained on a similar amount of tokens. Finally, the closed-book model is fine-tuned without retrieval augmentation. For the baseline without joint pre-training: the reader is also pre-trained without retrieval, and the retriever is initialized at finetuning from Contriever and trained with the LDist loss. Similarly the model pre-trained with a fixed retriever is fine-tuned with the LDist loss. We report results in Table 1. First, we note the poor performance of the closed-book baseline, indicating the importance of augmentation. Next, we observe that pre-training our model with retrieval is important to obtain good performance on few-shot tasks. Indeed, all models that include retrieval during pre-training

	64-shot				1024-shot			
	NQ	WoW	FEVER	Avg.	NQ	WoW	FEVER	Avg.
Prefix Language Modelling	41.0	14.5	64.9	40.1	**44.7**	17.9	86.0	49.5
Masked Language Modelling	**42.7**	**14.9**	**69.7**	**42.4**	**44.7**	**18.3**	**88.8**	**50.6**
Title-to-section generation	41.1	15.2	66.1	40.8	45.4	17.9	84.6	49.3

Table 2: Pretext task ablation. We compare different pretext tasks, used to jointly pre-train our models. Examples are randomly sampled from the training set of the KILT version of the data set. We report the exact match on Natural Questions, the F1 score on Wizard of Wikipedia and the accuracy on FEVER.

		64-shot				1024-shot			
Index	Training data	NQ	WoW	FEVER	Avg.	NQ	WoW	FEVER	Avg.
Wiki	Wiki	**42.7**	14.9	69.7	**42.4**	44.7	18.3	88.8	**50.6**
Wiki	CCNet	40.9	**15.3**	67.3	41.2	**44.8**	**18.4**	88.1	50.4
CCNet	Wiki	32.9	14.5	**72.1**	39.8	37.8	17.1	85.8	46.9
CCNet	CCNet	38.4	14.9	70.1	41.1	42.0	17.3	**88.9**	49.4

Table 3: Index content ablation. In this table, we report results for models where the content of the index was changed between the pre-training and the fine-tuning.

strongly outperform the baseline without joint pre-training. Next, we compare a model that was pre-trained with a fixed retriever, and models using the various retriever training objectives. On the MLM validation metric corresponding to the pre-training objective, we observe that jointly training the retriever leads to strong improvements. This effect tends to be less marked on 64-shot downstream tasks, and almost non-existent for 1024-shot. We believe that this is evidence that the biggest impact of pre-training is on the language model, which learns to use and aggregate information from the retrieved documents. Lastly, we do not observe significant systematic differences between the different retriever training objectives. We thus decide adopt use Likelihood Distillation for subsequent experiments, as it tends to be more stable than EMDR^2 or ADist, and more computationally efficient than LOOL.

Next, we compare the different self-supervised pretext tasks introduced in Section 2.3 in Table 2. Here we observe similar results for all three tasks, with a small advantage for masked language modelling. Thus, in what follows, we adopt masked language modelling for pre-training.

Finally, we consider different combinations of data sources—Wikipedia and common crawl—for the index and training data during pre-training. In all cases, we use the Wikipedia 2021 dump as the index when performing few-shot fine-tuning. We report results in Table 3. First, we observe that using a Wikipedia-based index leads to better downstream performance. There could be two explanations for this: first, as we use Wikipedia for the few-shot tasks, the model might be better adapted when trained using the same data. Another explanation

	64-shot				1024-shot			
	NQ	WoW	FEVER	Avg.	NQ	WoW	FEVER	Avg.
Standard fine-tuning	44.3	14.9	73.2	44.1	47.0	18.4	89.7	**51.7**
Top-100 re-ranking	44.2	14.6	75.4	**44.7**	**47.1**	**18.7**	88.9	51.6
Query-side fine-tuning	**45.0**	**15.0**	**77.0**	**45.7**	44.9	17.9	**90.2**	51.0
Fixed retriever	36.8	14.5	72.0	41.1	38.0	17.7	89.3	48.3

Table 4: Retriever fine-tuning ablation. Here, we compare different strategies to fine-tune the retriever in a few-shot setting.

might be that Wikipedia is a higher-quality and denser source of knowledge than common crawl. Second, when using a common crawl index, we observe that pre-training on Wikipedia data leads to lower performance than using common crawl data. We believe that the primary reason is that the distribution mismatch between the two domains leads to generally-less relevant retrieved documents. In turn, this probably means that the pre-training is less efficient, because the language model does not leverage as much information from the documents. In the following, we decide to combine the data from both domains for the index and the pre-training data to extend the coverage.

4.4 Fine-tuning

In this section, we perform an ablation study on how to apply our models on downstream tasks, which relies on fine-tuning. In particular, we want to investigate the following research question:

(RQ 3) How to efficiently fine-tune Atlas on tasks with limited training data?

To answer this question, we compare the different strategies to fine-tune the retriever module, described in Section 2.4. We report results in Table 4. First, as for pre-training, we observe that keeping the retriever fixed during fine-tuning leads to a significant performance drops, for both 64- and 1024-shot settings. Second, the re-ranking strategy (row 2) leads to very similar results to fully updating the index (row 1), while being significantly more efficient. Lastly, fine-tuning only the query encoder also leads to strong results: in particular, in the 64-shot setup, this is slightly stronger than performing full fine-tuning, which we attribute to there being less opportunity for over-fitting. On the other hand, on the 1024-shot setting, performing a full fine-tuning leads to stronger results, especially on Natural Questions. In the following, we use query-side fine-tuning for experiments with less than 64 examples, and standard fine-tuning for larger data sets.

5. Training and Evaluating Atlas

In this section, we apply the findings from the ablations of the previous sections to train a family of Atlas models, ranging from 770M to 11B parameters. More specifically, we use the Likelihood Distillation objective function, along with the masked language modelling

	5-shot			5-shot (multi-task)			Full / Transfer		
	770M	3B	11B	770M	3B	11B	770M	3B	11B
Closed-book T5	29.2	35.7	36.1	26.5	40.0	43.5	42.4	50.4	54.0
Atlas	38.9	42.3	43.4	42.1	48.7	56.4	56.3	59.9	65.8
Δ	+9.8	+6.6	+7.3	+15.6	+8.7	+12.9	+13.9	+9.5	+11.8

Table 5: Performance on MMLU as a function of model size. We report performance of Atlas on MMLU as a function of model size and compare it to closed-book T5.

pretext task. We pre-train these models using a mix of Wikipedia and Common Crawl data, for both the training data and content of the index. During pre-training, the reader generates based on 20 documents retrieved using the re-ranking strategy described in 2.4. For this we first retrieve 100 documents from the index containing embeddings, which are potentially stale, then these documents are re-embed and re-ranked using the up-to-date retriever. The index is updated every 2,500 steps. We pre-train models for 10,000 iterations using AdamW with a batch size of 128. While training longer continued to improve perplexity, we did not observe further improvements on downstream tasks after finetuning by training longer.

5.1 MMLU Results

As mentioned in section 4.1, we consider four setting for MMLU: 1) a zero-shot setting where we directly apply the pre-trained model with no few-shot finetuning 2) a 5-shot setting, where we finetune a model using 5 training examples for each of the 57 domains 3) a 5-shot multitask setting, where, rather than finetuning a model independently for each domain, we train a single model to perform all tasks and 4) a setting with access to a number of auxiliary data sets, with 95K total training examples. We train the models to generate the letter corresponding to the correct answer option (‘A’, ‘B’, ‘C’ or ‘D’), and pick the answer with the most likely of the 4 letters at test time. Full technical details can be found in appendix A.1.

5.1.1 Performance vs Parameters

We start by comparing Atlas to closed-book models of different sizes for 5-shot, 5-shot multitask and the full setting, and report results in Table 5. Across these settings, Atlas outperforms the closed-book baselines by between 6.6 and 15.6 points, demonstrating consistent utility of retrieval for few-shot language understanding across 57 domains. The closed-book T5 struggles to perform significantly better than random (25%) in few-shot settings with 770M parameters, whereas the equivalent Atlas achieves around 40%, significantly better than random, despite its small size. All models improve with more data, but interestingly, the 770M models do not benefit as much from few-shot multitask learning compared to larger models (for closed-book, it actually loses 3 points) suggesting smaller models struggle to grasp the synergies between the tasks in the few-shot setting. Larger models exploit the multi-task setting well, with Atlas improving more than closed-book. For example, Atlas-11B improves by 13 points ($43.4 \rightarrow 56.4$), but equivalent closed-book only improves by 7 (36.1 → 43.5). Finally, on the transfer learning setting, all models improve, but the relative gaps between closed-book at Atlas models remain similar.

5.1.2 De-biasing

When finetuning, we permute which answer option appears with which answer letter to reduce over-fitting and encourage a uniform prior over answer letters. However, the model may still exhibit a bias towards some letters, especially in few-shot settings, so we also include a second 'de-biased' inference mode in addition the standard inference used above. Here, we run 4 forward passes, one for each cyclic permutation of the answer letter-answer option assignment in the question, for example the answer option assigned to letter 'A' becomes 'B', what was 'B' becomes 'C' etc.[3] We then sum the 4 probabilities to obtain the final prediction, which reduces spurious bias towards one of the answer letters (further details in appendix A.1). The results are shown in Table 6. We find that in zero-shot and 5-shot settings, de-biasing is very effective, improving results by 10.3 and 4.5 points respectively. When more training data is available, the need for de-biasing decreases, leading to only 0.2 point improvement in the multi-task and full data settings.

5.1.3 Comparison to Published Works

Next, we compare our Atlas-11B results with de-biasing to recently reported results with state-of-the-art large language models such as GPT-3 or Chinchilla, which required significantly more amount of computation to train. We report results in Table 7. We find that Atlas is able to perform significantly better than random in zero-shot, and in conjunction with de-biased inference, achieves zero-shot scores that exceed 5-shot results reported with GPT3 in the literature (47.1% vs 43.9%) (Hendrycks et al., 2021). For the 5-shot setting, Atlas outperforms GPT-3 by 4%, while using 15× less parameters, and 10× less pre-training compute.[4] When multitask-training on the combined 5-shot data, Atlas improves to 56.6% close to the 5-shot performance of Gopher (60.0%). Finally, on the full data setting, where we train on auxiliary data recommended by the MMLU authors, Atlas reaches an overall accuracy of 65.6%, close to the state-of-the-art. Interestingly, in this setup, Atlas significantly outperforms GPT-3, while on the 5-shot setting, their performance is similar.

5.2 Open-domain Question Answering Results

Next we evaluate Atlas on two open-domain question answering benchmarks: Natural Questions and TriviaQA. We compare to prior work, both in a few-shot setting using 64 examples, and using the full training set, and report results in Table 8. On these benchmarks, which require high-degree of memorisation, we clearly see the benefits of retrieval-augmentation. Atlas-11B obtains state-of-the-art results on 64-shot question answering, for both Natural Questions and TriviaQA. In particular, it outperforms significantly larger

3. Exploring all answer option permutations would involve 24 forward passes, which improves results by an additional ∼1% over the 4 cyclic permutations, but requires much more compute, so we exclude it here, see Appendix A.1
4. Atlas's pre-training compute is dominated by the T5 pre-training. The computational requirements for the retrieval-augmented pre-train is orders of magnitude lower

	Zero-shot	5-shot	5-shot (multi-task)	Full / Transfer
Standard Inference	36.8	43.4	56.4	65.8
De-biased Inference	47.1	47.9	56.6	66.0

Table 6: Standard versus de-biased inference for MMLU. These results are reported for Atlas-11B, using cyclic permutations for de-biasing, which increases inference costs by a factor of 4×.

Setting	Model	Params	FLOPS	All	Hum.	Soc. Sci.	STEM	Other
zero-shot	Atlas	11B	3.5e22	47.1	43.6	54.1	38.0	54.4
5-shot	GPT-3	175B	3.1e23	43.9	40.8	50.4	36.7	48.8
	Gopher	280B	5.0e23	60.0	56.2	71.9	47.4	66.1
	Chinchilla	70B	5.0e23	**67.5**	**63.6**	**79.3**	**55.0**	73.9
	Atlas *	11B	3.5e22	47.9	46.1	54.6	38.8	52.8
5-shot MT	Atlas	11B	3.5e22	56.6	50.1	66.4	46.4	66.2
Transfer	UnifiedQA	11B	3.3e22	48.9	45.6	56.6	40.2	54.6
	GPT-3	175B	3.1e23	53.9	52.5	63.9	41.4	57.9
	Atlas	11B	3.5e22	66.0	61.1	77.2	53.2	**74.4**

Table 7: Comparison to state-of-the-art on MMLU. *For the 5-shot setting, Atlas uses fine-tuning, while previous works use in-context learning. The Atlas model uses de-biased inference. FLOPS refers to total the amount of computation necessary to train the model, including pre-training and/or fine-tuning. 5-shot MT refers to training a single model on multiple tasks, using 5 examples per task.

models, such as PaLM, or models that required significantly more training compute such as Chinchilla. When using the full training set, Atlas also obtains state-of-the-art results, for example improving the accuracy on Natural Questions from 55.9% to 60.4%. This result is obtained using an index comprised of CCNet and the December 2021 Wikipedia corpora, our default setting for the index. In section 6.2 we consider using indexes composed of Wikipedia corpus archived at different dates, and demonstrate an additional +3.6% on Natural Questions when using an index which is temporally matched to Natural Questions. We report performance as a function of model size as well as detailed hyperparameters in Appendix A.2.

Atlas also compares favorably to recent work exploring retrieval-augmented few-shot question answering with very large models. Lazaridou et al. (2022) explore Natural Questions in a 15-shot setup using Gopher, augmenting questions with 50 passages retrieved using Google Search. This method consists of generating 4 candidate answers from each retrieved passages, and then re-ranking using either a score inspired by RAG (Lewis et al., 2020) or a more expensive approach. This method (not shown in our tables) achieves exact match scores of 32.7% (RAG) and 38.4% (Ensemble), requiring 50 (RAG) or 450 (Ensemble) forward passes of Gopher-280B per test-time question. Atlas, using the same 15 training examples

	NQ		TriviaQA filtered		TriviaQA unfiltered	
Model	64-shot	Full	64-shot	Full	64-shot	Full
GPT-3 (Brown et al., 2020)	29.9	-	-	-	71.2	-
Gopher (Rae et al., 2021)	28.2	-	57.2	-	61.3	-
Chinchilla (Hoffmann et al., 2022)	35.5	-	64.6	-	72.3	-
PaLM (Chowdhery et al., 2022)	39.6	-	-	-	81.4	-
RETRO (Borgeaud et al., 2021)	-	45.5	-	-	-	-
FiD (Izacard and Grave, 2021a)	-	51.4	-	67.6	-	80.1
FiD-KD (Izacard and Grave, 2021b)	-	54.7	-	73.3	-	-
R2-D2 (Fajcik et al., 2021)	-	55.9	-	69.9	-	-
Atlas	**42.4**	**60.4**	**74.5**	**79.8**	**84.7**	**89.4**

Table 8: Comparison to state-of-the-art on question answering. We report results on Natural Questions, and on TriviaQA for both the filtered set, commonly used for open-domain question answering and the unfiltered hidden set for which evaluation is accessible online: `https://competitions.codalab.org/competitions/17208`. For the 64-shot setting, our model uses fine-tuning, while the other models use prompting.

and 50 passages achieves 38.7 EM, despite having 25× fewer parameters, and requiring comparatively negligible compute.

5.3 FEVER Results

We report results on the original 3-class FEVER fact checking test set in Table 9. We consider a 64-shot setting, with training examples uniformly sampled from the full training set. Unlike the development and test sets, the train set is imbalanced, with more positive labels than negative, posing a challenge for few-shot learning. In this setting, we achieve an accuracy of 64.3%. We also report a 15-shot setting, with 5 examples uniformly sampled from each class to compare with published results from Gopher (Rae et al., 2021), where Atlas scores 56.2%, outperforming Gopher by 5.1 points. Lastly we fine-tune our model on the full training set, and achieve a score of 78%, within 1.5% of the ProoFVer, which uses a specialized architecture, a retriever trained with sentence-level annotations, and is supplied with the Wikipedia corpus released with FEVER, whereas Atlas retrieves from CCNet and the December 2021 Wikipedia dump. If we give Atlas an index comprised of the FEVER Wikipedia corpus, we set a new state-of-the-art of 80.1%

5.4 KILT Results

Finally we evaluate Atlas on KILT, a benchmark composed of several different knowledge intensive tasks, which was described in section 4.1. We report results on test sets in Table 10 for which evaluation is available online.[5] The KILT versions of data sets are filtered, and thus results on Natural Questions, TriviaQA and FEVER reported elsewhere are not directly comparable on KILT. We consider both a 64-shot setting and a full fine-tuning setting, in both cases we train Atlas individually on each data set. More details on the hyperparameters and

5. `https://eval.ai/web/challenges/challenge-page/689`

	15-shot	65-shot	Full data set
Gopher (Rae et al., 2021)	51.1	-	-
ProoFVer (Krishna et al., 2022)	-	-	79.5
Atlas	**56.2**	**64.3**	78.0 / **80.1**†

Table 9: Comparison to state-of-the-art on FEVER. We report accuracy on FEVER test set, for which evaluation is available here: `https://competitions.codalab.org/competitions/18814`. For the few-shot settings, our model uses fine-tuning while other models use prompting. †uses an index composed of the FEVER Wikipedia corpus.

Model	AIDA	FEV	T-REx	zsRE	NQ	HoPo	TQA	WoW
	ACC	ACC	ACC	ACC	EM	EM	EM	F1
GENRE (Cao et al., 2021)	89.9	-	-	-	-	-	-	-
Sphere (Piktus et al., 2021)	-	89.0	81.7	74.2	51.6	38.3	72.7	15.5
SEAL (Bevilacqua et al., 2022)	-	89.5	83.6	74.6	53.7	40.5	70.9	18.3
Re2G (Glass et al., 2022)	-	89.6	**87.7**	-	51.7	-	76.3	18.9
FID+RS (Hofstätter et al., 2022)	-	92.2	85.2	**83.7**	61.2	39.1	**84.6**	20.6
Atlas, 64-shot	66.5	87.1	58.9	74.9	43.6	34.7	76.4	15.5
Atlas, full train set	**90.6**	**93.5**	85.1	80.8	**61.3**	**50.6**	84.0	**21.6**

Table 10: Downstream results on the KILT hidden test sets. Downstream metrics are accuracy (AIDA CoNLL-YAGO, FEVER, T-REx, zero-shot RE), exact match (Natural Questions, HotpotQA, TriviaQA), or F1 (Wizard of Wikipedia).

development set results are reported in Appendix A.3. For 64-shot, we greatly exceed random performance, and are even competitive with some fully-finetuned models on the leaderboard, such as for FEVER, where our 64-shot Atlas is only 2-2.5 points behind Sphere, SEAL and Re2G, and outperforms Sphere and SEAL on zero-shot RE. In the full data set setting, Atlas is within 3% to the state-of-the-art for 3 data sets, and sets the state-of-the-art in the remaining five data sets.

6. Analysis

In this section we discuss specific aspects of Atlas as a retrieval-augmented language model. First, we analyse retrieved documents to interpret Atlas generations. Second, we probe the updateability and temporal sensitivity of Atlas when the content of the index is modified.

6.1 Interpretability and Leakage

An advantage of semi-parametric models like Atlas is the ability to inspect retrieved items to aid interpretability. To better understand how well Atlas retrieves, and how it uses retrieved passages, we examine the retrieved passages for multi-task few-shot MMLU. As

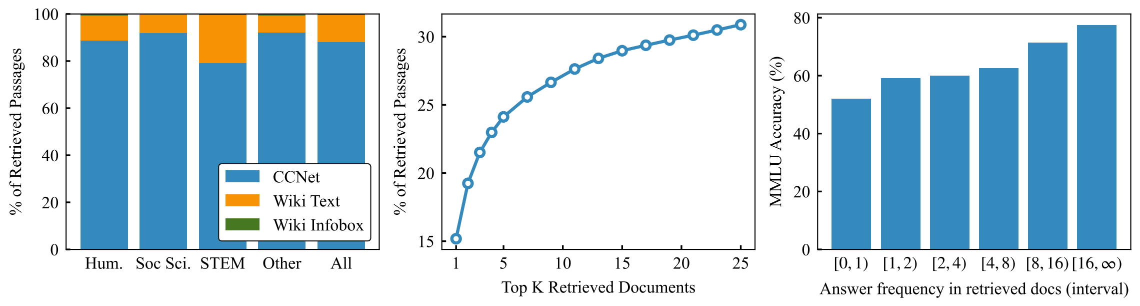

Figure 3: MMLU Retrieval Analysis. Left: Fraction of sources of top 30 retrieved passages for MMLU from CCNet, Wikipedia passages and info boxes for the 5-shot multitask Atlas. Center: How often the text of the correct MMLU answer option appears in retrieved passages, as a function of the number of retrieved passages. Right: MMLU accuracy as a function of answer occurrence frequency in retrieved passages set

shown in the left panel of Figure 3, the model retrieves the majority of its passages from CCNet (85% on average). Wikipedia makes up about 15% of retrieved passages, which is higher than we would expect under a uniform prior, given Wikipedia only makes up about 10% of the index. The fraction of Wikipedia retrieval varies between MMLU domains, with the model using Wikipedia to a greater extent for STEM domains, and least for social sciences. The domain making the greatest use of Wikipedia is "abstract algebra" (73%), and the least is "moral scenarios" (3%). We also note that the MMLU-finetuned Atlas does not make significant use of Wikipedia infobox passages.

We can also analyse the content of passages to assess how they may be useful for accomplishing the downstream task. The middle panel of Figure 3 shows how often retrieved documents contain the text of the correct answer option. There being at least one mention of the correct answer choice in 30% of test questions in the top 25 passages.[6] The right panel shows that the accuracy on MMLU increases when the correct answer option text occurs more frequently in retrieved passages, rising from 55% for questions when the answer option does not appear, to 77% for questions mentioned more than 15 times.

A human analysis of retrieved documents revealed that documents are helpful for answering questions in a number of different ways. Manual inspection of a sample of 50 correctly-answered questions revealed that 44% contained at least partially useful background information. These are documents that would improve the likelihood of a non-expert human answering correctly, such as contextual clues surrounding a quotation from a question, or helpful numerical figures for quantity-based questions, which help to narrow down the answer options to a smaller range. In a further 26% of cases, a passage contained all the necessary information to answer the question, stated in a straightforward way. If read competently, such passages make the question simple to answer, and often include information such as

6. Note: Depending on the question, it may not be important or useful to retrieve the exact text of the answer in MMLU, and as such, a hits@k value of 30% does not imply that retrieval fails to surface useful information in 70% of cases

canonical definitions, or the exact numerical answer requested in the question. 28% of retrieval sets did not contain obvious information which would make the question easier. Finally, 2% contained the verbatim question in a passage, together with its answer.

Given that MMLU has been created from pre-existing exams, it is possible that these questions appear on the open web. Models trained on web data (or, in our case, retrieving from it) run the risk of answering correctly not through generalisation, but by verbatim memorisation, which could lead to misleadingly high scores. In some very large language models, which can verbatim memorize and recall large parts of their pre-training data (Carlini et al., 2021), efforts have sometimes been made to filter occurrences of downstream instances from pre-training data, but this has not been performed for MMLU in the literature. In order to assess the prevalence of MMLU leakage in our index, we manually checked retrieval results for questions where the longest n-gram overlap between the question (without answer options) and a passage was at least 75% the length of the question. This resulted in an estimate of leakage of 2.8% of questions from our CCNet corpus.

A benefit of retrieval-augmented models such as Atlas is the editability of its knowledge (see Section 6.2 for additional analysis). To estimate pure, non-leaked performance, we can filter out any potentially-leaked passages from retrieved results and rerun the language model. The MMLU score drops slightly when controlling for this leakage from 56.4 to 55.8% (-.5%).We note that our CCNet corpus is relatively small compared to the pre-trained corpora of recent very large models, which are trained on up to 1.4 trillion tokens (Hoffmann et al., 2022), 35x the size of our index, making it likely that models trained on corpora of that size would observe more MMLU leaked examples, but detecting such leakage is challenging in non-retrieval-augmented models.

6.2 Temporal Sensitivity and Updateability

A benefit of retrieval-augmented models is that they can be kept up-to-date without retraining, by updating or swapping their index at test time. To assess the effectiveness of this mechanism in Atlas, we first construct a data set of time-sensitive questions derived from TempLAMA (Dhingra et al., 2022). TempLAMA is a collection of templated cloze questions derived from Wikidata and Wikipedia where the correct answer changes over time. We select a subset of questions from this data set which have a different answer in 2017 and 2020, for example, `Question:` *Theo Walcott plays for ___* `Answer:` *Arsenal F.C. (2017), Everton F.C. (2020)*, and form a small training set of 248 training, 112 development and 806 test questions.

Using this data set, we finetune closed-book T5-XXL and Atlas using the questions and the 2017 answers, supplying Atlas with a 2017 Wikipedia index, and then measure exact match accuracy on the 2017 test set. The results can be found in the first row and first two columns of Table 11. We first observe that, as expected, Atlas greatly outperforms T5 (57.7% c.f. 12.1%). We also note that, as desired, both T5 and Atlas almost never generate an answer from 2020 when trained with the 2017 answers, scoring 2.8% and 1.5% respectively (first row, second two columns of Table 11). However, as shown in row 2, we can swap the Atlas index to a 2020 Wikipedia index, *without retraining*, and find that Atlas updates its predictions accordingly, with 2020 accuracy rising to a similar level to its 2017 performance (53.1%), whereas the purely parametric T5 has no such updateability mechanism.

		2017 Test Set Acc.		2020 Test Set Acc.	
Train Set	Test-time Index	Closed-book	Atlas	Closed-book	Atlas
2017 answers	2017	12.1	57.7	2.9	1.5
	2020	12.1	10.2	2.9	53.1
2020 answers	2017	4.8	50.1	3.6	4.2
	2020	4.8	3.5	3.6	60.5

Table 11: Results on our TempLAMA-derived data set.
We report performance for a static, closed-book T5-11B, as well as Atlas-11B supplied with a test-time Wikipedia index from 2017 or 2020. We evaluate models finetuned on a small training set of 248 time-sensitive cloze-question-answer pairs, using answers either from 2017 or 2020. Good models should score highly when the test set year matches the year of the test-time index, and score low otherwise.

	Dec. 2017	Dec. 2018	Aug. 2019	Dec. 2020	Dec. 2021
64-shot	44.7	**45.1**	44.1	44.0	41.3
Full	63.2	**64.0**	62.4	61.1	59.6

Table 12: Impact of index data temporality on Natural Questions. We report exact match performance on Natural Questions using different Wikipedia dumps in the index. We observe that the dump from December 2018, commonly used for Natural Questions, leads to the best result.

This demonstrates that Atlas can be faithful and condition strongly on its supplied index. Furthermore, this zero-shot updateability mechanism has the useful property of staying up-to-date without requiring up-to-date annotated data, or continuous, lifelong pre-training, as would be may required for a large parametric-only model. Rows 3 and 4 of Table 11 complete the picture, where this time we train with 2020 answers, and demonstrate Atlas can zero-shot transfer backwards in time to 2017 effectively too (50.1%). Interestingly, T5 is unable to answer questions from 2020 well, even when trained with 2020 answers (3.6%), likely because it was pre-trained on data pre-dating 2020 (Dodge et al., 2021).

We also examine temporal effects for Natural Questions. Natural Questions is a data set composed of search queries collected via the Google search engine in a short period of time. Thus data have a strong temporal bias, with a lot of questions about the 2018 World Cup for example. Moreover some questions are ambiguous without specification of the temporal context. For instance, for the question *"when did ireland last beat england at twickenham"*, the expected answer is 2018 in Natural Questions, while Ireland also beat England at Twickenham in 2022 as well as many other times before. In Table 12, we report results obtained by finetuning Atlas using different Wikipedia dumps for the index. We observe that the 2018 December Wikipedia dump, which is close to the date of data collection,

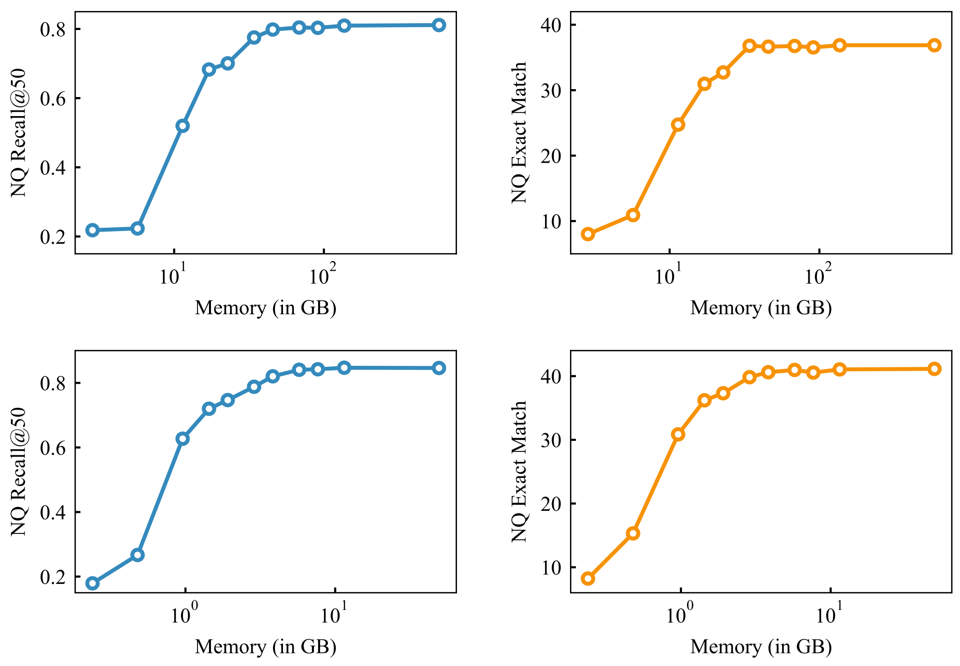

Figure 4: Index Compression: Atlas-3B 64-shot NQ performance (left column: Retrieval Recall@50, right column: QA Exact Match score), as a function of index size, for different levels of quantisation. The right-most point in each plot represents the uncompressed index. **Top Row:** Wikipedia + CCNet Index. **Bottom Row:** Wikipedia Index.

leads to the best results for both few-shot and full fine-tuning. In particular, it leads to a new state-of-the-art of 64.0 EM on Natural Questions.

6.2.1 Index Compression

Maintaining dense retrieval indices can be memory-intensive, especially as the number of indexed items is scaled. In this section, we briefly analyse the memory requirements of Atlas's index in the case of a) a Wikipedia index and b) the combined CCNet and Wikipedia index used in most of the experiments above.

There are two sources of memory pressure for Atlas's retrieval component—the passages themselves, and the document embedding index. The tokenized passages, once binarized, require 11GB and 130GB of storage for the Wikipedia and combined indices respectively. These passages do not need to be stored in expensive GPU RAM, and could even be memory-mapped to disk, sharded across nodes or compressed if required, and thus do not represent a limiting hardware challenge in this context. The embedding index itself, however, must be stored in GPU RAM for fast search, and thus its size is more sensitive. In the above experiments, we perform exact search over our index, which is achieved by sharding the index over all the the available GPUs, and computing the search in parallel. The index is stored at fp16 precision, resulting in a total GPU memory requirement of 49 GB and 587 GB for the Wikipedia and combined indices, respectively.

This large GPU memory requirement for the index limits accessibility and ease of deployment. However, many index compression techniques are available for nearest neighbour search, which can often dramatically reduce memory requirements at the cost of some retrieval accuracy. Following Izacard et al. (2020), we explore the effect of Product Quantization (PQ, Jegou et al., 2010), a popular lossy compression technique on Atlas-3B's accuracy for the 64-shot NQ task at different compression levels.

The results are shown in Figure 4. We find that substantial compression is possible before the onset of significant performance degradation. Namely, the Wikipedia index can be compressed from 49GB to 4GB with negligible drop in retrieval precision and exact match. Likewise, the combined index can be compressed from 587GB to 50GB without serious degradation, indicating that the combined index could be loaded onto a single 80GB GPU.

7. Discussion

In this paper, we introduce Atlas, a large retrieval-augmented language model. By jointly pre-training the retriever module and the language model, we show that Atlas has strong few-shot learning capabilities on a wide range of knowledge intensive tasks, including Natural Questions, TriviaQA, FEVER, 8 KILT tasks and 57 MMLU tasks. For example, Atlas-11B reaches more than 42% accuracy on Natural Questions and 84.7% on TriviaQA when training on 64 examples, which is an improvement of almost 3 points compared to PaLM, a 540B parameter model, which required 50x more pre-training compute. We also provided detailed ablations and analyses for what factors are important when training such retrieval-augmented models, and demonstrated Atlas's updateability, interpretability and controlability capabilities. Lastly, we demonstrated that Atlas is also powerful in full data set settings obtaining a new state-of-the-art results on Natural Questions, TriviaQA, FEVER, and 5 KILT tasks. The few-shot results presented in this paper were obtained by fine-tuning Atlas on few examples, rather than using in-context learning. In context learning presents significant practical advantages over fine-tuning, as it does not change the model weights. The development of retrieval-augmented language models preserving the ability of their non-augmented counterparts to generalize from few in-context examples and instructions is a crucial challenge toward general retrieval-augmented language models and their wider adoption.

Appendix A. Training details and Additional Results

In this appendix we present additional results and provide details about the parameters used to fine-tune models on MMLU, question answering data sets and KILT tasks.

A.1 MMLU

Here, we report results on the 57 MMLU domains, details about the fine-tuning and how the model predictions are de-biased.

A.1.1 Featurization

MMLU consists of multiple choice questions with four possible lexicalized answer options. We represent the input using the following template:

```
question: {question text}
options: (A) {answer 1} (B) {answer 2} (C) {answer 3} (D) {answer 4}
answer: [MASK_0]
```

and train the model to generate the mask token followed by the letter of the correct answer:

```
[MASK_0] {correct answer option letter}
```

This format closely matches the format of MLM pre-training objective, aiding few-shot learning. When training, we permute the order of the answer options, that is shuffling which answer option appears as letter A etc. This helps reduce overfitting, and encourages a uniform prior on the letters.

A.1.2 Standard Inference

Once trained we obtain predictions from the model by selecting the pre-softmax logits for the tokens A, B, C and D, and performing a softmax over them to obtain a distribution over the 4 answer options. For standard inference, we then simply return the answer corresponding to the argmax of this distribution.

A.1.3 De-biased Inference

As mentioned in the main text, even though our model is finetuned with data that encourages a uniform prior over answer letters (by permuting which answer option letter is used with which lexical answer option text in training data), this may not be enough to ensure the model has no residual bias towards specific letters. Consider answers a, questions q and a nuisance variable $z \in \mathcal{Z}$, which represents the ordering of the answer options or, equivalently, which answer letter gets assigned to which answer option text. There are 4 answer options in MMLU, and thus $|\mathcal{Z}| = 24$ unique ways they can be ordered, or assigned to given letters. Running our model with our standard inference for a question q, corresponds to calculating $p(a|q = \mathsf{q}, z = \mathsf{z})$ for the answer ordering z that happens to appear in the data set. We can control for z by running the model with all possible answer orderings in the input, and marginalizing: $p(a|q = \mathsf{q}) = \sum_{z' \in \mathcal{Z}} p(a|q = \mathsf{q}, z = z')p(z = z'|q = \mathsf{q})$, and assuming $p(z = z'|q = \mathsf{q})$ is uniform (no answer ordering is more likely than another), this reduces to simply $p(a|q = \mathsf{q}) \propto \sum_{z' \in \mathcal{Z}} p(a|q = \mathsf{q}, z = z')$. This procedure requires 24 forward passes, one for each answer ordering, so is 24× slower than standard inference. Table 13 shows the result of applying the full permutation de-biasing, which leads to an 12% improvement zero-shot and 6% in 5-shot performance overall. Empirically, using only the cyclic permutations of the answer order provided in the original data set (of which there are 4) works nearly as well, which is what we report in the main paper, and only increases inference compute by a factor of 4, rather than 24. Cyclic permutation de-biasing improves over standard inference by 10% in zero-shot and 5% in 5-shot. Empirically, de-biased inference is largely unnecessary when training in the 5-shot multitask or full data set setting, as there is enough data for the model to learn a more uniform prior over the letters.

Setting	Model	All	Hum.	Soc. Sci.	STEM	Other
zero-shot	Standard	36.8	37.5	39.0	30.2	39.7
	All permutations	48.5	45.7	55.2	39.4	54.4
	Cyclic Permutations	47.1	43.6	54.1	38.0	54.9
5-shot	Standard	43.4	41.8	49.3	33.9	48.8
	All permutations	49.0	46.0	56.1	40.5	54.6
	Cyclic Permutations	47.9	46.1	54.6	38.8	52.8

Table 13: MMLU scores with de-biasing.

A.1.4 Evaluation

We evaluate by following the method of Hendrycks et al. (2021), namely, micro-averaging across all 57 domains to obtain overall accuracy. We quote the results of GPT3 (Brown et al., 2020) and UnifiedQA (Khashabi et al., 2020) from the MMLU leaderboard at `https://github.com/hendrycks/test`. For Chinchilla and Gopher, we calculate the scores on the categories using the full MMLU results from Hoffmann et al. (2022).

A.1.5 Index

The index used for MMLU for all MMLU experiments in the main paper comprised of concatenation of the Wikipedia passages, Wikipedia info boxes and Common Crawl indices, for a total of 387M passages. We can assess the importance of the index by running a model without the common crawl data, leading to a 5-shot multitask result of 52.8%, compared to 56.4% for the full model, a drop of 3.6%. This indicates that whilst the Wikipedia data is sufficient do well on the task, the addition of the CCNet data improves results further.

A.1.6 Hyperparameters and Development Data

Selecting hyperparameters is challenging in few-shot settings. We do not assume access to an in-domain development set for the 5-shot task. Instead, we determine a set of hyperparameters for the 5-shot task using data from RACE, one of the auxiliary data sets provided by MMLU. Here, we sample 5 sets of 5-shot training data, and for each model size, we explore batch size $\{32, 64\}$, learning rates for the language model and retriever {(5e-5, 1e-5), (4e-5, 4e-5)}, retriever temperature $\{0.1, 0.01\}$ and a fixed number of training steps $\{16, 32, 64, 128\}$, picking the setting that achieves strongest RACE validation scores. Having determined these hyperparameters, we apply them directly to the 5-shot MMLU task. For the 5-shot multi-task and full/transfer settings, we use the same batch size, temperatures and learning rates as the 5-shot task, but use a set of 285 MMLU validation examples (5 per domain) in order to determine the total number of training steps and for early stopping. The hyperparameters selected in the MMLU experiments can be found in Table 14. We use query-side finetuning for the 5-shot and 5-shot multitask settings, and top-128 reranking for the full setting. For all MMLU runs we retrieve 30 documents.

	770M	3B	11B
Batch size	64	64	64
Learning rate	(5e-5, 1e-5)	(5e-5, 1e-5)	(5e-5, 1e-5)
Retriever Temperature	0.1	0.1	0.1
5-shot train steps	64	32	16
5-shot (multitask) max train steps	2000	500	250
Full / transfer max train steps	5000	2000	2000

Table 14: Hyperparameters for MMLU

Run #	All	Hum.	Soc. Sci.	STEM	Other
1	45.2	40.6	54.1	37.1	51.1
2	45.1	39.8	54.4	37.1	52.0
3	45.0	40.0	54.1	37.7	51.1
4	45.6	41.3	54.7	37.0	51.6
5	44.3	40.6	50.7	38.1	49.8
Ave:	45.0 ± 0.5	40.5 ± 0.6	53.6 ± 1.6	37.4 ± 0.5	51.1 ± 0.8

Table 15: Interrun Variance for 5-shot MMLU using Atlas-11B

A.1.7 Inter-run Variance

few-shot learning is well-known to suffer from high variance. In the main paper, we quote the result obtained with our first run. In order to assess the effect of noise and variance, we ran the 5-shot experiment with Atlas 5 times.[7] We observe high variance for individual domains, sometimes as high as 20%, however, once aggregated across all 57 domains, the inter-run variance is low. The overall scores for these different runs, when using the same hyperparameters are shown in Table 15. Due the effects of averaging over the many domains that comprise MMLU, the inter-run variance is quite modest on the aggregated metrics, with a std deviation of 0.5 in this experiment.

A.1.8 Closed-Book Baselines

The closed book baselines we compare Atlas to in Table 5 are initialized from the same T5 model as their respective Atlas, and then pre-trained with MLM for the same number of steps (10K) using the same pre-training data as Atlas, for fairer comparison. The same procedure as for Atlas was used to determine hyperparameters for MMLU for the closed-book models.

7. This experiment was performed with a slightly different index to the main experiments, which achieves a stronger result

A.1.9 Full Results

Tables 16 and 17 shows the full MMLU scores for each domain for Atlas and the closed book T5 respectively. The full results for the cyclic-permutation-de-biased Atlas-XXL can be found in Table 18.

A.2 Question Answering

We report additional training details on question answering tasks, as well as results obtained with models of different sizes.

A.2.1 Training Details

For question answering, similarly to the MMLU experiments, we format the input using the following template:

`question: {question text} answer: [MASK_0]`

and train the model to generate the mask token followed by the answer:

`[MASK_0] {answer}`.

We generate answers using greedy decoding. For both training and testing, we retrieve 40 passages, and truncate the result of the concatenation between the query and the passages to 384 tokens.

For few-shot fine-tuning we train Atlas for 30 steps using 64 random samples from the train sets. The retriever is trained using query-side fine-tuning. We select the model after 30 training steps. We use AdamW with a batch size of 32 and a learning rate of 4×10^{-5} with linear decay and 5 iterations of warmup for both the language model and the retriever.

For the fine-tuning on the full data sets, we train the model for 5k gradient steps and refresh the index every 500 steps for the first 1,000 training steps and every 2k training steps afterwards. We use AdamW with a batch size of 64 and a learning rate of 4×10^{-5} with linear decay and 5 iterations of warmup for both the language model and the retriever. We evaluate models every 500 steps and select the best one on the validation set based on the exact match score.

A.2.2 Impact of Scaling

In Table 19, we report performance on Natural Questions and TriviaQA as a function of the number of parameters in the reader module. Both for few-shot learning and full fine-tuning we observe strong improvements by scaling the size of the reader module. However we can notice sign of saturation when finetuning on full data sets, with limited gains when scaling from 3B to 11B parameters (+0.6% on Natural Questions, +0.5% on TriviaQA). While performance improves substantially when scaling from 3B to 11B parameters with 64 training samples, with +3.7% and +1.2% improvement on Natural Questions and TriviaQA respectively. For these experiments we use a setup similar to the one use in Table 8, except that we use an index composed of the December 2018 Wikipedia dump processed as described in section 4.2.

	5-shot			5-shot (multi-task)			Full / Transfer		
	770M	3B	11B	770M	3B	11B	770M	3B	11B
All	38.9	42.3	43.4	42.1	48.7	56.4	56.3	59.9	65.8
Humanities	37.3	40.0	41.9	37.7	46.4	50.0	50.9	53.0	60.3
Social Sciences	41.7	46.8	49.3	47.5	53.7	65.6	66.0	70.8	77.2
STEM	32.3	35.0	33.9	34.4	39.4	46.2	44.8	50.7	53.4
Other	44.9	48.1	48.8	50.4	55.9	66.6	65.5	68.1	74.4
abstract algebra	30.0	27.0	28.0	27.0	31.0	30.0	22.0	27.0	33.0
anatomy	28.9	50.4	45.2	44.4	57.8	64.4	57.8	68.9	69.6
astronomy	55.3	59.9	59.2	52.6	66.4	67.8	69.1	78.3	79.6
business ethics	49.0	51.0	48.0	50.0	62.0	60.0	51.0	70.0	68.0
clinical knowledge	41.9	44.9	40.0	46.8	54.3	64.9	64.2	72.5	74.0
college biology	38.2	45.8	50.0	36.8	52.1	63.2	63.2	72.2	78.5
college chemistry	32.0	29.0	29.0	31.0	33.0	38.0	45.0	39.0	45.0
college computer science	33.0	35.0	30.0	23.0	29.0	30.0	43.0	48.0	47.0
college mathematics	31.0	31.0	28.0	29.0	27.0	34.0	32.0	29.0	36.0
college medicine	31.2	35.8	38.2	50.3	40.5	52.0	60.1	59.5	63.6
college physics	20.6	26.5	31.4	21.6	28.4	39.2	27.5	44.1	42.2
computer security	53.0	50.0	55.0	49.0	61.0	64.0	69.0	71.0	76.0
conceptual physics	34.9	41.7	37.4	40.9	43.4	57.0	53.2	58.3	59.6
econometrics	28.9	21.1	27.2	26.3	25.4	34.2	28.9	37.7	36.8
electrical engineering	26.9	31.7	31.7	38.6	44.1	51.7	61.4	60.7	67.6
elementary mathematics	25.9	28.8	29.4	29.6	30.2	32.8	29.6	35.5	33.9
formal logic	34.9	33.3	33.3	23.0	30.2	29.4	34.1	38.9	34.1
global facts	28.0	34.0	34.0	36.0	40.0	49.0	50.0	49.0	52.0
high school biology	24.8	37.7	27.7	48.7	57.1	66.5	66.5	76.8	81.9
high school chemistry	34.5	31.0	31.0	31.5	36.5	48.3	44.8	52.2	52.2
high school computer science	31.0	39.0	28.0	37.0	42.0	42.0	50.0	59.0	57.0
high school european history	42.4	49.7	53.3	50.9	58.2	69.7	70.9	73.9	80.0
high school geography	38.9	42.4	50.0	46.5	56.6	69.2	74.2	80.8	82.8
high school gov. and pol.	57.5	60.6	60.1	52.9	64.8	76.7	80.8	85.5	91.7
high school macroeconomics	32.8	39.7	44.9	39.0	45.6	57.2	55.1	63.1	66.7
high school mathematics	30.7	33.0	35.6	28.1	27.8	37.0	30.7	34.8	37.0
high school microeconomics	34.5	42.9	45.4	44.1	51.7	68.9	63.4	70.2	81.1
high school physics	18.5	24.5	22.5	25.8	25.8	33.1	27.2	30.5	39.7
high school psychology	52.8	61.1	59.8	56.7	67.2	79.4	76.3	84.0	87.0
high school statistics	39.8	29.6	34.7	27.3	34.7	38.0	37.0	43.1	45.8
high school us history	43.6	49.0	55.9	46.1	57.8	59.8	62.7	72.5	76.5
high school world history	48.1	52.7	59.9	48.1	66.2	65.4	70.0	78.5	79.7
human aging	46.2	44.8	39.5	48.0	55.2	60.1	56.1	68.2	73.1
human sexuality	41.2	43.5	27.5	46.6	51.1	59.5	77.1	72.5	81.7
international law	54.5	57.9	60.3	55.4	72.7	73.6	81.8	82.6	85.1
jurisprudence	38.9	55.6	32.4	53.7	60.2	73.1	76.9	73.1	81.5
logical fallacies	43.6	54.0	57.1	44.2	58.3	70.6	64.4	73.0	76.7
machine learning	36.6	34.8	28.6	31.3	37.5	46.4	36.6	47.3	50.9
management	45.6	51.5	52.4	48.5	52.4	81.6	78.6	75.7	87.4
marketing	59.4	67.1	70.5	66.7	74.4	83.8	83.8	83.3	91.9
medical genetics	50.0	53.0	58.0	56.0	61.0	75.0	68.0	78.0	81.0
miscellaneous	63.0	64.2	68.8	64.0	72.4	84.3	85.4	83.9	90.9
moral disputes	37.0	41.3	41.3	40.8	50.3	60.1	61.9	66.2	73.7
moral scenarios	24.7	24.7	26.5	21.9	26.9	26.6	23.8	23.8	35.8
nutrition	40.9	45.1	45.1	49.0	52.3	67.0	64.7	68.6	76.8
philosophy	48.6	50.5	56.3	49.8	59.2	69.5	70.4	73.0	77.8
prehistory	45.7	50.0	52.8	54.9	64.8	74.4	69.8	75.0	80.6
professional accounting	28.4	33.0	34.0	35.1	34.0	45.7	43.6	46.1	51.8
professional law	32.4	33.5	34.8	30.4	37.6	39.1	41.5	41.5	50.5
professional medicine	29.4	26.1	27.6	34.6	40.8	52.2	47.8	43.4	59.6
professional psychology	37.7	43.0	50.2	45.1	51.0	60.6	59.5	62.4	74.0
public relations	40.0	46.4	44.5	51.8	54.5	66.4	63.6	66.4	68.2
security studies	35.1	33.5	38.8	44.1	39.6	57.6	60.8	61.6	72.2
sociology	45.3	51.2	51.2	52.7	60.2	69.2	74.1	78.6	85.1
us foreign policy	58.0	70.0	73.0	63.0	63.0	74.0	80.0	80.0	83.0
virology	34.3	34.3	32.5	38.0	42.8	45.2	47.6	49.4	53.0
world religions	65.5	69.0	71.9	70.2	82.5	80.1	83.6	83.6	87.1

Table 16: MMLU Test set scores for Atlas for each model size and each of the 57 domains.

	5-shot			5-shot (multi-task)			Full / Transfer		
	770M	3B	11B	770M	3B	11B	770M	3B	11B
All	29.2	35.7	36.1	26.5	40.0	43.5	42.4	50.4	54.0
Humanities	30.5	35.4	35.5	27.3	38.5	41.6	41.0	48.6	51.3
Social Sciences	29.7	38.0	39.4	24.8	43.8	48.9	48.6	57.8	64.7
STEM	29.0	31.4	30.8	26.5	32.8	35.8	33.4	40.6	41.7
Other	26.7	37.7	38.6	27.0	45.0	48.5	46.8	55.2	59.1
abstract algebra	26.0	23.0	21.0	29.0	30.0	26.0	23.0	29.0	26.0
anatomy	21.5	40.0	40.7	27.4	39.3	45.9	35.6	43.7	42.2
astronomy	37.5	38.8	37.5	27.6	39.5	41.4	36.2	50.7	55.3
business ethics	29.0	54.0	42.0	26.0	47.0	55.0	53.0	64.0	60.0
clinical knowledge	32.5	33.6	40.0	28.7	44.2	47.9	45.3	52.8	57.7
college biology	29.9	34.7	34.0	29.9	34.7	40.3	38.2	46.5	52.1
college chemistry	37.0	22.0	32.0	20.0	35.0	33.0	36.0	34.0	36.0
college computer science	28.0	35.0	34.0	28.0	27.0	36.0	31.0	44.0	35.0
college mathematics	31.0	29.0	27.0	22.0	34.0	27.0	30.0	33.0	32.0
college medicine	24.3	34.7	34.1	27.2	40.5	40.5	35.8	41.6	48.6
college physics	33.3	23.5	23.5	22.5	19.6	26.5	22.5	32.4	24.5
computer security	36.0	42.0	46.0	31.0	49.0	52.0	50.0	65.0	61.0
conceptual physics	26.4	35.7	30.2	23.4	30.6	32.8	34.5	37.4	43.8
econometrics	26.3	21.9	28.9	17.5	19.3	24.6	29.8	25.4	29.8
electrical engineering	31.0	33.1	31.7	31.0	31.0	36.6	41.4	47.6	51.7
elementary mathematics	26.2	27.5	28.0	27.0	31.2	33.3	25.9	31.2	35.5
formal logic	34.1	34.1	31.7	15.1	34.9	31.0	31.7	38.1	42.1
global facts	32.0	30.0	25.0	34.0	34.0	27.0	28.0	34.0	30.0
high school biology	22.6	31.9	29.7	27.1	41.6	50.0	43.5	57.7	60.6
high school chemistry	27.1	26.6	27.6	28.6	31.5	29.1	30.5	36.5	38.9
high school computer science	26.0	32.0	25.0	33.0	37.0	45.0	45.0	55.0	48.0
high school european history	34.5	43.0	42.4	24.2	60.0	59.4	58.2	69.1	76.4
high school geography	31.3	40.4	36.9	24.7	45.5	50.5	56.1	66.7	74.2
high school gov. and pol.	28.0	49.2	51.3	19.2	56.0	59.6	55.4	70.5	75.6
high school macroeconomics	25.6	37.7	32.1	26.7	42.3	43.6	41.0	51.5	56.4
high school mathematics	35.9	35.2	35.9	28.1	26.7	31.1	27.8	36.7	31.9
high school microeconomics	27.3	29.8	36.1	20.6	35.7	42.9	42.9	50.8	60.5
high school physics	21.9	25.2	22.5	24.5	28.5	29.1	27.8	31.1	27.8
high school psychology	26.1	46.4	51.0	24.8	54.3	60.2	56.3	67.3	76.1
high school statistics	27.8	33.3	33.3	17.6	30.6	33.8	32.9	33.3	37.0
high school us history	30.4	39.7	45.6	27.5	46.1	58.3	51.0	63.2	72.5
high school world history	42.6	50.6	41.8	29.1	54.0	64.6	66.7	72.2	73.8
human aging	28.3	37.2	29.6	26.0	45.3	46.2	46.6	57.0	62.8
human sexuality	29.8	34.4	41.2	25.2	42.0	44.3	51.1	58.0	59.5
international law	57.9	57.9	41.3	44.6	57.9	58.7	62.8	71.9	71.1
jurisprudence	30.6	33.3	34.3	32.4	49.1	52.8	55.6	67.6	74.1
logical fallacies	40.5	55.8	46.6	25.8	51.5	62.0	43.6	69.3	71.2
machine learning	33.0	34.8	36.6	29.5	35.7	37.5	32.1	37.5	42.9
management	21.4	29.1	40.8	24.3	47.6	50.5	60.2	69.9	70.9
marketing	38.9	58.5	60.7	31.2	67.9	75.6	69.2	79.9	85.9
medical genetics	26.0	36.0	36.0	29.0	43.0	44.0	40.0	54.0	50.0
miscellaneous	24.5	45.2	46.4	27.1	52.2	58.2	51.3	64.6	72.7
moral disputes	32.4	37.3	38.7	28.6	43.4	43.4	49.7	64.7	64.7
moral scenarios	24.7	24.7	24.7	23.0	23.9	24.7	23.8	24.0	23.8
nutrition	30.1	33.0	34.6	25.8	42.5	44.1	50.3	55.6	61.1
philosophy	28.6	32.5	37.3	31.2	38.9	45.0	44.1	56.6	59.2
prehistory	33.6	37.0	41.4	27.5	39.8	50.6	41.0	51.5	57.7
professional accounting	21.3	28.0	30.5	25.9	35.5	34.0	37.2	41.5	42.2
professional law	28.2	33.4	34.0	27.6	35.4	35.5	38.3	43.0	45.6
professional medicine	19.5	26.5	24.3	20.2	32.0	37.9	38.6	40.8	46.0
professional psychology	27.8	32.8	32.8	26.6	39.5	43.6	38.4	48.0	58.3
public relations	22.7	43.6	40.0	21.8	47.3	56.4	50.0	55.5	60.0
security studies	37.6	26.1	31.0	20.4	34.7	44.1	56.3	61.6	66.9
sociology	43.3	41.8	38.8	30.8	45.8	52.7	60.2	66.7	72.1
us foreign policy	49.0	57.0	66.0	38.0	56.0	61.0	59.0	75.0	76.0
virology	29.5	26.5	34.3	30.1	36.1	39.8	44.0	46.4	41.6
world religions	24.0	40.9	47.4	32.7	49.1	57.3	48.0	63.7	70.2

Table 17: MMLU Test set scores for the T5 closed book baseline for each model size and each of the 57 domains.

Domain	zero-shot	5-shot	5-shot (multi-task)	Full / Transfer
All	47.1	47.9	56.6	66.0
Humanities	43.6	46.1	50.1	61.1
Social Sciences	54.1	54.6	66.4	77.2
STEM	38.0	38.8	46.4	53.2
Other	53.9	52.8	66.2	74.4
abstract algebra	22.0	26.0	31.0	31.0
anatomy	48.9	47.4	62.2	70.4
astronomy	61.8	62.5	68.4	81.6
business ethics	60.0	57.0	62.0	70.0
clinical knowledge	50.6	49.4	66.4	72.8
college biology	51.4	53.5	61.1	77.8
college chemistry	36.0	39.0	39.0	45.0
college computer science	32.0	32.0	33.0	49.0
college mathematics	30.0	35.0	35.0	34.0
college medicine	44.5	41.0	52.6	67.6
college physics	24.5	26.5	37.3	42.2
computer security	59.0	59.0	68.0	76.0
conceptual physics	37.0	41.3	57.0	60.0
econometrics	20.2	20.2	36.8	37.7
electrical engineering	37.9	40.0	50.3	65.5
elementary mathematics	31.2	28.0	30.7	36.5
formal logic	27.8	27.0	32.5	35.7
global facts	41.0	43.0	51.0	53.0
high school biology	53.2	56.5	68.7	83.2
high school chemistry	41.9	41.4	49.3	51.2
high school computer science	40.0	36.0	46.0	60.0
high school european history	56.4	58.8	68.5	80.6
high school geography	57.1	59.6	71.2	81.3
high school gov. and pol.	67.9	67.9	77.2	90.2
high school macroeconomics	46.9	48.5	57.9	65.9
high school mathematics	28.1	28.9	34.1	31.5
high school microeconomics	51.7	51.7	68.9	82.4
high school physics	26.5	25.8	32.5	41.1
high school psychology	66.2	65.5	78.9	86.8
high school statistics	31.5	30.1	43.1	45.8
high school us history	57.8	54.9	64.7	77.5
high school world history	59.1	62.9	65.4	79.3
human aging	48.4	50.7	60.5	70.4
human sexuality	55.7	54.2	61.8	84.0
international law	66.1	72.7	71.9	84.3
jurisprudence	61.1	64.8	72.2	81.5
logical fallacies	54.6	57.7	71.2	77.9
machine learning	37.5	39.3	43.8	44.6
management	56.3	56.3	79.6	89.3
marketing	72.2	73.1	84.6	91.9
medical genetics	55.0	58.0	71.0	81.0
miscellaneous	69.7	67.8	83.8	90.4
moral disputes	45.1	46.8	60.1	72.3
moral scenarios	24.5	30.3	25.8	38.5
nutrition	56.5	53.9	67.0	77.1
philosophy	56.3	57.6	70.7	77.2
prehistory	59.3	60.5	71.6	78.7
professional accounting	35.1	33.0	42.2	50.7
professional law	36.3	38.4	39.4	51.7
professional medicine	35.7	33.1	52.2	60.7
professional psychology	47.7	49.3	60.9	74.0
public relations	54.5	53.6	68.2	68.2
security studies	47.3	45.7	59.2	73.9
sociology	62.2	62.7	71.6	84.6
us foreign policy	64.0	68.0	73.0	83.0
virology	39.8	40.4	44.6	51.8
world religions	77.2	74.9	80.7	87.1

Table 18: MMLU Test set scores for the de-biased Atlas-XXL using cyclic permutations for each of the 57 domains for zero-shot, 5 shot, 5-shot-multitask and the transfer setting.

Number of parameters	220M	770M	3B	11B
Natural Questions 64-shot	27.0	35.4	41.3	45.1
Natural Questions full	54.1	60.8	63.4	64.0
TriviaQA 64-shot	55.3	65.0	70.2	71.4
TriviaQA full	71.8	74.9	77.5	78.0

Table 19: **Impact of model size on question answering data sets.** We report exact match performance on the test sets of Natural Questions and TriviaQA filtered depending on the number of parameters in the reader module. For these experiments the index contains the December 2018 Wikipedia dump.

A.3 KILT

For the results on KILT reported in Table 10 we fine-tune Atlas individually on each data set. We format the input using a template similar to the one used for question answering:

`question: {query text} answer: [MASK_0]`

and train the model to generate the mask token followed by the expected output:

`[MASK_0] {output}`.

We retrieve 20 passages and generate answer using greedy decoding. In KILT, FEVER is a two-way classification task of claims. We lexicalize the "SUPPORTS" (resp. 'REFUTES") label into "true" (respectively "false").

For few-shot fine-tuning we train Atlas for 30 steps using 64 random samples from the train sets. The retriever is trained using query-side fine-tuning. We evaluate models every 5 steps and select the best one on the development set based on the reported metric. We use AdamW with a batch size of 32 and a learning rate of 4×10^{-5} with linear decay and 5 iterations of warmup for both the language model and the retriever.

For the fine-tuning on the full data sets, the model is trained for 5k gradient steps. We evaluate models every 500 steps and select the best one on the development set based on the reported metric. The index is refreshed every 500 step for the first 1000 iterations, and every 2k steps afterwards. We use AdamW with a batch size of 64 and a learning rate of 4×10^{-5} with linear decay and 500 iterations of warmup for both the language model and the retriever.

We report results on the development sets in Table 20.

References

M. Bevilacqua, G. Ottaviano, P. Lewis, S. Yih, S. Riedel, and F. Petroni. Autoregressive search engines: Generating substrings as document identifiers. In *NeurIPS*, 2022. URL `https://openreview.net/forum?id=Z4kZxAjg8Y`.

S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. v. d. Driessche, J.-B. Lespiau, B. Damoc, A. Clark, D. d. L. Casas, A. Guy, J. Menick, R. Ring, T. Hennigan,

Model	AIDA ACC	FEV ACC	T-REx ACC	zsRE ACC	NQ EM	HoPo EM	TQA EM	WoW F1
Atlas 64-shot	69.0	88.1	58.5	60.2	44.2	34.1	77.1	15.4
Atlas full data set	92.7	94.4	84.8	80.9	63.4	51.4	84.4	21.0

Table 20: Downstream results on the KILT dev sets. Downstream metrics are accuracy (AIDA CoNLL-YAGO, FEVER, T-REx, zero-shot RE), exact match (Natural Questions, HotpotQA, TriviaQA), or F1 (Wizard of Wikipedia).

S. Huang, L. Maggiore, C. Jones, A. Cassirer, A. Brock, M. Paganini, G. Irving, O. Vinyals, S. Osindero, K. Simonyan, J. W. Rae, E. Elsen, and L. Sifre. Improving language models by retrieving from trillions of tokens, 2021. URL https://arxiv.org/abs/2112.04426.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

N. D. Cao, G. Izacard, S. Riedel, and F. Petroni. Autoregressive entity retrieval. In *ICLR*, 2021. URL https://openreview.net/forum?id=5k8F6UU39V.

N. Carlini, F. Tramèr, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. B. Brown, D. Song, Úlfar Erlingsson, A. Oprea, and C. Raffel. Extracting training data from large language models. In *USENIX Security Symposium*, 2021. URL https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting.

D. Chen, A. Fisch, J. Weston, and A. Bordes. Reading Wikipedia to answer open-domain questions. In *ACL*, 2017. URL https://aclanthology.org/P17-1171.

A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel. Palm: Scaling language modeling with pathways, 2022. URL https://arxiv.org/abs/2204.02311.

C. Clark and M. Gardner. Simple and effective multi-paragraph reading comprehension. In *ACL*, 2018. URL https://aclanthology.org/P18-1078.

P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

B. Dhingra, J. R. Cole, J. M. Eisenschlos, D. Gillick, J. Eisenstein, and W. W. Cohen. Time-aware language models as temporal knowledge bases. *TACL*, 2022. URL https://aclanthology.org/2022.tacl-1.15.

E. Dinan, S. Roller, K. Shuster, A. Fan, M. Auli, and J. Weston. Wizard of wikipedia: Knowledge-powered conversational agents. In *ICLR*, 2019. URL https://openreview.net/forum?id=r1l73iRqKm.

J. Dodge, M. Sap, A. Marasović, W. Agnew, G. Ilharco, D. Groeneveld, M. Mitchell, and M. Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *EMNLP*, 2021. URL https://aclanthology.org/2021.emnlp-main.98.

H. Elsahar, P. Vougiouklis, A. Remaci, C. Gravier, J. Hare, F. Laforest, and E. Simperl. T-REx: A large scale alignment of natural language with knowledge base triples. In *LREC*, 2018. URL https://aclanthology.org/L18-1544.

M. Fajcik, M. Docekal, K. Ondrej, and P. Smrz. R2-D2: A modular baseline for open-domain question answering. In *Findings of EMNLP*, 2021. URL https://aclanthology.org/2021.findings-emnlp.73.

M. Fink. Object classification from a single example utilizing class relevance metrics. In *NIPS*, 2005. URL https://proceedings.neurips.cc/paper/2004/file/ef1e491a766ce3127556063d49bc2f98-Paper.pdf.

T. Gao, A. Fisch, and D. Chen. Making pre-trained language models better few-shot learners. In *ACL-IJCNLP*, 2021. URL https://aclanthology.org/2021.acl-long.295.

M. Glass, G. Rossiello, M. F. M. Chowdhury, A. R. Naik, P. Cai, and A. Gliozzo. Re2g: Retrieve, rerank, generate, 2022. URL https://arxiv.org/abs/2207.06300.

E. Grave, M. Cisse, and A. Joulin. Unbounded cache model for online language modeling with open vocabulary, 2017a. URL https://arxiv.org/abs/1711.02604.

E. Grave, A. Joulin, and N. Usunier. Improving neural language models with a continuous cache. In *ICLR*, 2017b. URL https://openreview.net/forum?id=B184E5qee.

K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang. Realm: Retrieval-augmented language model pre-training. *arXiv:2002.08909*, 2020. URL https://arxiv.org/abs/2002.08909.

K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. In *ICLR*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. Robust disambiguation of named entities in text. In *EMNLP*, 2011. URL https://aclanthology.org/D11-1072.

J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. v. d. Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

S. Hofstätter, J. Chen, K. Raman, and H. Zamani. Multi-task retrieval-augmented text generation with relevance sampling, 2022. URL https://arxiv.org/abs/2207.03030.

P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*, 2013.

G. Izacard and E. Grave. Leveraging passage retrieval with generative models for open domain question answering. In *EACL*, 2021a. URL https://aclanthology.org/2021.eacl-main.74.

G. Izacard and E. Grave. Distilling knowledge from reader to retriever for question answering. In *ICLR*, 2021b. URL https://openreview.net/forum?id=NTEz-6wysdb.

G. Izacard, F. Petroni, L. Hosseini, N. De Cao, S. Riedel, and E. Grave. A memory efficient baseline for open domain question answering, 2020. URL https://arxiv.org/abs/2012.15156.

G. Izacard, M. Caron, L. Hosseini, S. Riedel, P. Bojanowski, A. Joulin, and E. Grave. Unsupervised dense information retrieval with contrastive learning. *TMLR*, 2022. URL https://openreview.net/forum?id=jKN1pXi7b0.

H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE TPAMI*, 2010.

Z. Jiang, F. F. Xu, J. Araki, and G. Neubig. How can we know what language models know? *TACL*, 2020. URL https://aclanthology.org/2020.tacl-1.28.

K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*, 2017. URL https://aclanthology.org/P17-1147.

J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

V. Karpukhin, B. Oğuz, S. Min, L. Wu, S. Edunov, D. Chen, and W.-t. Yih. Dense passage retrieval for open-domain question answering. *arXiv:2004.04906*, 2020. URL `https://arxiv.org/abs/2004.04906`.

U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, and M. Lewis. Generalization through memorization: Nearest neighbor language models. In *ICLR*, 2020. URL `https://openreview.net/forum?id=HklBjCEKvH`.

D. Khashabi, S. Min, T. Khot, A. Sabharwal, O. Tafjord, P. Clark, and H. Hajishirzi. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of EMNLP*, 2020. URL `https://aclanthology.org/2020.findings-emnlp.171`.

A. Krishna, S. Riedel, and A. Vlachos. ProoFVer: Natural logic theorem proving for fact verification. *TACL*, 2022. URL `https://aclanthology.org/2022.tacl-1.59`.

T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov. Natural questions: A benchmark for question answering research. *TACL*, 2019. URL `https://aclanthology.org/Q19-1026`.

G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *EMNLP*, 2017. URL `https://aclanthology.org/D17-1082`.

A. Lazaridou, E. Gribovskaya, W. Stokowiec, and N. Grigorev. Internet-augmented language models through few-shot prompting for open-domain question answering, 2022. URL `https://arxiv.org/abs/2203.05115`.

T. Le Scao and A. Rush. How many data points is a prompt worth? In *NAACL-HLT*, 2021. URL `https://aclanthology.org/2021.naacl-main.208`.

H. Lee, A. Kedia, J. Lee, A. Paranjape, C. D. Manning, and K.-G. Woo. You only need one model for open-domain question answering, 2021a. URL `https://arxiv.org/abs/2112.07381`.

J. Lee, M. Sung, J. Kang, and D. Chen. Learning dense representations of phrases at scale. In *ACL-IJCNLP*, 2021b. URL `https://aclanthology.org/2021.acl-long.518`.

K. Lee, M.-W. Chang, and K. Toutanova. Latent retrieval for weakly supervised open domain question answering. In *ACL*, 2019. URL `https://aclanthology.org/P19-1612`.

B. Lester, R. Al-Rfou, and N. Constant. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, 2021. URL `https://aclanthology.org/2021.emnlp-main.243`.

O. Levy, M. Seo, E. Choi, and L. Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *CoNLL*, 2017. URL `https://aclanthology.org/K17-1034`.

P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *NeurIPS*, 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf`.

X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL-IJCNLP*, 2021. URL `https://aclanthology.org/2021.acl-long.353`.

O. Lieber, O. Sharir, B. Lenz, and Y. Shoham. Jurassic-1: Technical details and evaluation. Technical report, AI21 Labs, 2021.

I. R. L. Logan, I. Balavzevi'c, E. Wallace, F. Petroni, S. Singh, and S. Riedel. Cutting down on prompts and parameters: Simple few-shot learning with language models, 2021. URL `https://arxiv.org/abs/2106.13353`.

T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018. URL `https://aclanthology.org/D18-1260`.

S. Min, J. Michael, H. Hajishirzi, and L. Zettlemoyer. AmbigQA: Answering ambiguous open-domain questions. In *EMNLP*, 2020. URL `https://aclanthology.org/2020.emnlp-main.466`.

R. Nakano, J. Hilton, S. A. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, X. Jiang, K. Cobbe, T. Eloundou, G. Krueger, K. Button, M. Knight, B. Chess, and J. Schulman. Webgpt: Browser-assisted question-answering with human feedback, 2021. URL `https://arxiv.org/abs/2112.09332`.

A. Paranjape, O. Khattab, C. Potts, M. Zaharia, and C. D. Manning. Hindsight: Posterior-guided training of retrievers for improved open-ended generation, 2021. URL `https://arxiv.org/abs/2110.07752`.

F. Petroni, P. Lewis, A. Piktus, T. Rocktäschel, Y. Wu, A. H. Miller, and S. Riedel. How context affects language models' factual predictions. *arXiv:2005.04611*, 2020. URL `https://arxiv.org/abs/2005.04611`.

F. Petroni, S. Broscheit, A. Piktus, P. Lewis, G. Izacard, L. Hosseini, J. Dwivedi-Yu, M. Lomeli, T. Schick, P.-E. Mazaré, A. Joulin, E. Grave, and S. Riedel. Improving wikipedia verifiability with ai, 2022. URL `https://arxiv.org/abs/2207.06220`.

A. Piktus, F. Petroni, V. Karpukhin, D. Okhonko, S. Broscheit, G. Izacard, P. Lewis, B. Oğuz, E. Grave, W.-t. Yih, and S. Riedel. The web is your oyster - knowledge-intensive nlp against a very large web corpus, 2021. URL `https://arxiv.org/abs/2112.09924`.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI Technical Report*, 2019.

J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, E. Rutherford, T. Hennigan, J. Menick, A. Cassirer, R. Powell, G. v. d. Driessche, L. A. Hendricks, M. Rauh, P.-S. Huang, A. Glaese, J. Welbl, S. Dathathri, S. Huang, J. Uesato, J. Mellor, I. Higgins, A. Creswell, N. McAleese, A. Wu, E. Elsen, S. Jayakumar, E. Buchatskaya, D. Budden, E. Sutherland, K. Simonyan, M. Paganini, L. Sifre, L. Martens, X. L. Li, A. Kuncoro, A. Nematzadeh, E. Gribovskaya, D. Donato,

A. Lazaridou, A. Mensch, J.-B. Lespiau, M. Tsimpoukelli, N. Grigorev, D. Fritz, T. Sottiaux, M. Pajarskas, T. Pohlen, Z. Gong, D. Toyama, C. d. M. d'Autume, Y. Li, T. Terzi, V. Mikulik, I. Babuschkin, A. Clark, D. d. L. Casas, A. Guy, C. Jones, J. Bradbury, M. Johnson, B. Hechtman, L. Weidinger, I. Gabriel, W. Isaac, E. Lockhart, S. Osindero, L. Rimell, C. Dyer, O. Vinyals, K. Ayoub, J. Stanway, L. Bennett, D. Hassabis, K. Kavukcuoglu, and G. Irving. Scaling language models: Methods, analysis & insights from training gopher, 2021. URL `https://arxiv.org/abs/2112.11446`.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2019. URL `https://arxiv.org/abs/1910.10683`.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*, 2016. URL `https://aclanthology.org/D16-1264`.

O. Ram, G. Shachaf, O. Levy, J. Berant, and A. Globerson. Learning to retrieve passages without supervision. In *NAACL-HLT*, 2022. URL `https://aclanthology.org/2022.naacl-main.193`.

M. Richardson. MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text. In *EMNLP*, 2013. URL `https://www.microsoft.com/en-us/research/publication/mctest-challenge-dataset-open-domain-machine-comprehension-text/`.

M. Richardson, C. J. Burges, and E. Renshaw. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, 2013. URL `https://aclanthology.org/D13-1020`.

S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, et al. Okapi at TREC-3. *NIST Special Publication Sp*, 1995.

D. S. Sachan, S. Reddy, W. Hamilton, C. Dyer, and D. Yogatama. End-to-end training of multi-document reader and retriever for open-domain question answering, 2021. URL `https://arxiv.org/abs/2106.05346`.

T. Schick and H. Schutze. It's not just size that matters: Small language models are also few-shot learners, 2021. URL `https://arxiv.org/abs/2009.07118`.

T. Schick and H. Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *EACL*, 2021a. URL `https://aclanthology.org/2021.eacl-main.20`.

T. Schick and H. Schütze. Few-shot text generation with natural language instructions. In *EMNLP*, 2021b. URL `https://aclanthology.org/2021.emnlp-main.32`.

Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. Learning semantic representations using convolutional neural networks for web search. In *WWW*, 2014.

T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *EMNLP*, 2020. URL https://aclanthology.org/2020.emnlp-main.346.

K. Shuster, S. Poff, M. Chen, D. Kiela, and J. Weston. Retrieval augmentation reduces hallucination in conversation, 2021. URL https://arxiv.org/abs/2104.07567.

K. Shuster, M. Komeili, L. Adolphs, S. Roller, A. D. Szlam, and J. Weston. Language models that seek for knowledge: Modular search & generation for dialogue and prompt completion, 2022. URL https://arxiv.org/abs/2203.13224.

S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhumoye, G. Zerveas, V. Korthikanti, E. Zhang, R. Child, R. Y. Aminabadi, J. Bernauer, X. Song, M. Shoeybi, Y. He, M. Houston, S. Tiwary, and B. Catanzaro. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model, 2022. URL https://arxiv.org/abs/2201.11990.

D. Tam, R. R. Menon, M. Bansal, S. Srivastava, and C. Raffel. Improving and simplifying pattern exploiting training, 2021. URL https://arxiv.org/abs/2103.11955.

R. Thoppilan, D. De Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H.-T. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, Y. Li, H. Lee, H. S. Zheng, A. Ghafouri, M. Menegali, Y. Huang, M. Krikun, D. Lepikhin, J. Qin, D. Chen, Y. Xu, Z. Chen, A. Roberts, M. Bosma, V. Zhao, Y. Zhou, C.-C. Chang, I. Krivokon, W. Rusch, M. Pickett, P. Srinivasan, L. Man, K. Meier-Hellstern, M. R. Morris, T. Doshi, R. D. Santos, T. Duke, J. Soraker, B. Zevenbergen, V. Prabhakaran, M. Diaz, B. Hutchinson, K. Olson, A. Molina, E. Hoffman-John, J. Lee, L. Aroyo, R. Rajakumar, A. Butryna, M. Lamm, V. Kuzmina, J. Fenton, A. Cohen, R. Bernstein, R. Kurzweil, B. Aguera-Arcas, C. Cui, M. Croak, E. Chi, and Q. Le. Lamda: Language models for dialog applications, 2022. URL https://arxiv.org/abs/2201.08239.

J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal. FEVER: a large-scale dataset for fact extraction and VERification. In *NAACL-HTL*, 2018. URL https://aclanthology.org/N18-1074.

S. Thrun and L. Pratt. *Learning to Learn: Introduction and Overview*, page 3–17. Kluwer Academic Publishers, USA, 1998. ISBN 0792380479.

O. Vinyals, C. Blundell, T. Lillicrap, k. kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In *NIPS*, 2016. URL https://proceedings.neurips.cc/paper/2016/file/90e1357833654983612fb05e3ec9148c-Paper.pdf.

E. M. Voorhees. The TREC-8 question answering track report. In *LREC*, 1999. URL http://www.lrec-conf.org/proceedings/lrec2000/pdf/26.pdf.

Z. Wang, P. Ng, X. Ma, R. Nallapati, and B. Xiang. Multi-passage BERT: A globally normalized BERT model for open-domain question answering. In *EMNLP-IJCNLP*, 2019. URL https://aclanthology.org/D19-1599.

G. Wenzek, M.-A. Lachaux, A. Conneau, V. Chaudhary, F. Guzmán, A. Joulin, and E. Grave. CCNet: Extracting high quality monolingual datasets from web crawl data. In *LREC*, 2020. URL `https://aclanthology.org/2020.lrec-1.494`.

L. Xiong, C. Xiong, Y. Li, K.-F. Tang, J. Liu, P. N. Bennett, J. Ahmed, and A. Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *ICLR*, 2021. URL `https://openreview.net/forum?id=zeFrfgyZln`.

Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering, 2018. URL `https://arxiv.org/abs/1809.09600`.

A. Yates, R. Nogueira, and J. Lin. Pretrained Transformers for Text Ranking: BERT and Beyond. In *WSDM*, 2021.

W.-t. Yih, K. Toutanova, J. C. Platt, and C. Meek. Learning discriminative projections for text similarity measures. In *CoNLL*, 2011. URL `https://aclanthology.org/W11-0329`.

D. Yogatama, C. de Masson d'Autume, and L. Kong. Adaptive semiparametric language models. *TACL*, 2021. URL `https://aclanthology.org/2021.tacl-1.22`.