Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

Patrick Lewis[†‡], **Ethan Perez**[⋆],

Aleksandra Piktus[†], **Fabio Petroni**[†], **Vladimir Karpukhin**[†], **Naman Goyal**[†], **Heinrich Küttler**[†],

Mike Lewis[†], **Wen-tau Yih**[†], **Tim Rocktäschel**[†‡], **Sebastian Riedel**[†‡], **Douwe Kiela**[†]

[†]Facebook AI Research; [‡]University College London; [⋆]New York University;
plewis@fb.com

Abstract

Large pre-trained language models have been shown to store factual knowledge in their parameters, and achieve state-of-the-art results when fine-tuned on downstream NLP tasks. However, their ability to access and precisely manipulate knowledge is still limited, and hence on knowledge-intensive tasks, their performance lags behind task-specific architectures. Additionally, providing provenance for their decisions and updating their world knowledge remain open research problems. Pre-trained models with a differentiable access mechanism to explicit non-parametric memory have so far been only investigated for extractive downstream tasks. We explore a general-purpose fine-tuning recipe for retrieval-augmented generation (RAG) — models which combine pre-trained parametric and non-parametric memory for language generation. We introduce RAG models where the parametric memory is a pre-trained seq2seq model and the non-parametric memory is a dense vector index of Wikipedia, accessed with a pre-trained neural retriever. We compare two RAG formulations, one which conditions on the same retrieved passages across the whole generated sequence, and another which can use different passages per token. We fine-tune and evaluate our models on a wide range of knowledge-intensive NLP tasks and set the state of the art on three open domain QA tasks, outperforming parametric seq2seq models and task-specific retrieve-and-extract architectures. For language generation tasks, we find that RAG models generate more specific, diverse and factual language than a state-of-the-art parametric-only seq2seq baseline.

1 Introduction

Pre-trained neural language models have been shown to learn a substantial amount of in-depth knowledge from data [47]. They can do so without any access to an external memory, as a parameterized implicit knowledge base [51, 52]. While this development is exciting, such models do have downsides: They cannot easily expand or revise their memory, can't straightforwardly provide insight into their predictions, and may produce "hallucinations" [38]. Hybrid models that combine parametric memory with non-parametric (i.e., retrieval-based) memories [20, 26, 48] can address some of these issues because knowledge can be directly revised and expanded, and accessed knowledge can be inspected and interpreted. REALM [20] and ORQA [31], two recently introduced models that combine masked language models [8] with a differentiable retriever, have shown promising results,

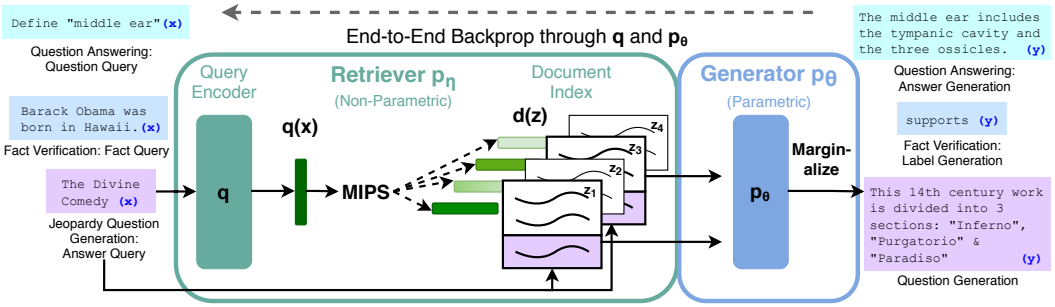

Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder* + *Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x, we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i. For final prediction y, we treat z as a latent variable and marginalize over seq2seq predictions given different documents.

but have only explored open-domain extractive question answering. Here, we bring hybrid parametric and non-parametric memory to the "workhorse of NLP," i.e. sequence-to-sequence (seq2seq) models.

We endow pre-trained, parametric-memory generation models with a non-parametric memory through a general-purpose fine-tuning approach which we refer to as retrieval-augmented generation (RAG). We build RAG models where the parametric memory is a pre-trained seq2seq transformer, and the non-parametric memory is a dense vector index of Wikipedia, accessed with a pre-trained neural retriever. We combine these components in a probabilistic model trained end-to-end (Fig. 1). The retriever (Dense Passage Retriever [26], henceforth DPR) provides latent documents conditioned on the input, and the seq2seq model (BART [32]) then conditions on these latent documents together with the input to generate the output. We marginalize the latent documents with a top-K approximation, either on a per-output basis (assuming the same document is responsible for all tokens) or a per-token basis (where different documents are responsible for different tokens). Like T5 [51] or BART, RAG can be fine-tuned on any seq2seq task, whereby both the generator and retriever are jointly learned.

There has been extensive previous work proposing architectures to enrich systems with non-parametric memory which are trained from scratch for specific tasks, e.g. memory networks [64, 55], stack-augmented networks [25] and memory layers [30]. In contrast, we explore a setting where both parametric and non-parametric memory components are pre-trained and pre-loaded with extensive knowledge. Crucially, by using pre-trained access mechanisms, the ability to access knowledge is present without additional training.

Our results highlight the benefits of combining parametric and non-parametric memory with generation for *knowledge-intensive tasks*—tasks that humans could not reasonably be expected to perform without access to an external knowledge source. Our RAG models achieve state-of-the-art results on open Natural Questions [29], WebQuestions [3] and CuratedTrec [2] and strongly outperform recent approaches that use specialised pre-training objectives on TriviaQA [24]. Despite these being extractive tasks, we find that unconstrained generation outperforms previous extractive approaches. For knowledge-intensive generation, we experiment with MS-MARCO [1] and Jeopardy question generation, and we find that our models generate responses that are more factual, specific, and diverse than a BART baseline. For FEVER [56] fact verification, we achieve results within 4.3% of state-of-the-art pipeline models which use strong retrieval supervision. Finally, we demonstrate that the non-parametric memory can be replaced to update the models' knowledge as the world changes.[1]

2 Methods

We explore RAG models, which use the input sequence x to retrieve text documents z and use them as additional context when generating the target sequence y. As shown in Figure 1, our models leverage two components: (i) a retriever $p_\eta(z|x)$ with parameters η that returns (top-K truncated) distributions over text passages given a query x and (ii) a generator $p_\theta(y_i|x, z, y_{1:i-1})$ parametrized

[1]Code to run experiments with RAG has been open-sourced as part of the HuggingFace Transformers Library [66] and can be found at `https://github.com/huggingface/transformers/blob/master/examples/rag/`. An interactive demo of RAG models can be found at `https://huggingface.co/rag/`

by θ that generates a current token based on a context of the previous $i-1$ tokens $y_{1:i-1}$, the original input x and a retrieved passage z.

To train the retriever and generator end-to-end, we treat the retrieved document as a latent variable. We propose two models that marginalize over the latent documents in different ways to produce a distribution over generated text. In one approach, *RAG-Sequence*, the model uses the same document to predict each target token. The second approach, *RAG-Token*, can predict each target token based on a different document. In the following, we formally introduce both models and then describe the p_η and p_θ components, as well as the training and decoding procedure.

2.1 Models

RAG-Sequence Model The RAG-Sequence model uses the same retrieved document to generate the complete *sequence*. Technically, it treats the retrieved document as a single latent variable that is marginalized to get the seq2seq probability $p(y|x)$ via a top-K approximation. Concretely, the top K documents are retrieved using the retriever, and the generator produces the output sequence probability for each document, which are then marginalized,

$$p_{\text{RAG-Sequence}}(y|x) \approx \sum_{z \in \text{top-}k(p(\cdot|x))} p_\eta(z|x) p_\theta(y|x,z) = \sum_{z \in \text{top-}k(p(\cdot|x))} p_\eta(z|x) \prod_i^N p_\theta(y_i|x,z,y_{1:i-1})$$

RAG-Token Model In the RAG-Token model we can draw a different latent document for each target *token* and marginalize accordingly. This allows the generator to choose content from several documents when producing an answer. Concretely, the top K documents are retrieved using the retriever, and then the generator produces a distribution for the next output token for each document, before marginalizing, and repeating the process with the following output token, Formally, we define:

$$p_{\text{RAG-Token}}(y|x) \approx \prod_i^N \sum_{z \in \text{top-}k(p(\cdot|x))} p_\eta(z|x) p_\theta(y_i|x,z,y_{1:i-1})$$

Finally, we note that RAG can be used for sequence classification tasks by considering the target class as a target sequence of length one, in which case RAG-Sequence and RAG-Token are equivalent.

2.2 Retriever: DPR

The retrieval component $p_\eta(z|x)$ is based on DPR [26]. DPR follows a bi-encoder architecture:

$$p_\eta(z|x) \propto \exp\left(\mathbf{d}(z)^\top \mathbf{q}(x)\right) \qquad \mathbf{d}(z) = \text{BERT}_d(z), \;\; \mathbf{q}(x) = \text{BERT}_q(x)$$

where $\mathbf{d}(z)$ is a dense representation of a document produced by a $\text{BERT}_{\text{BASE}}$ *document encoder* [8], and $\mathbf{q}(x)$ a query representation produced by a *query encoder*, also based on $\text{BERT}_{\text{BASE}}$. Calculating top-k$(p_\eta(\cdot|x))$, the list of k documents z with highest prior probability $p_\eta(z|x)$, is a Maximum Inner Product Search (MIPS) problem, which can be approximately solved in sub-linear time [23]. We use a pre-trained bi-encoder from DPR to initialize our retriever and to build the document index. This retriever was trained to retrieve documents which contain answers to TriviaQA [24] questions and Natural Questions [29]. We refer to the document index as the *non-parametric memory*.

2.3 Generator: BART

The generator component $p_\theta(y_i|x,z,y_{1:i-1})$ could be modelled using any encoder-decoder. We use BART-large [32], a pre-trained seq2seq transformer [58] with 400M parameters. To combine the input x with the retrieved content z when generating from BART, we simply concatenate them. BART was pre-trained using a denoising objective and a variety of different noising functions. It has obtained state-of-the-art results on a diverse set of generation tasks and outperforms comparably-sized T5 models [32]. We refer to the BART generator parameters θ as the *parametric memory* henceforth.

2.4 Training

We jointly train the retriever and generator components without any direct supervision on what document should be retrieved. Given a fine-tuning training corpus of input/output pairs (x_j, y_j), we