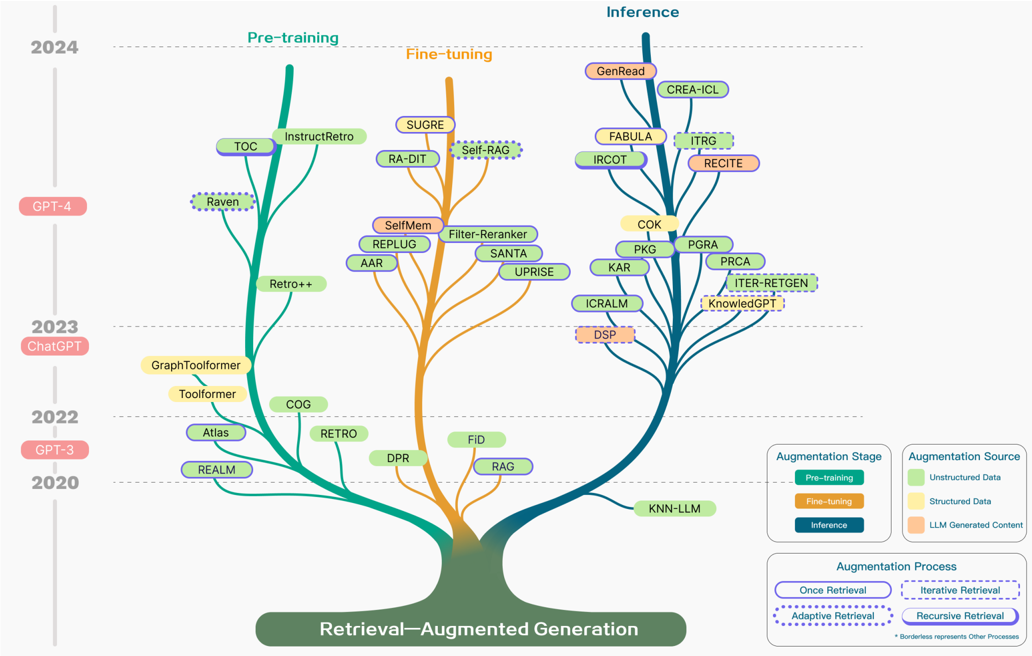
Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

Patrick Lewis[1,2] Ethan Perez[3] Aleksandra Piktus[1] Fabio Petroni[1]
Vladimir Karpukhin[1] Naman Goyal[1] Heinrich Küttler[1] Mike Lewis[1]
Wen-tau Yih[1] Tim Rocktäschel[1,2] Sebastian Riedel[1,2] Douwe Kiela[1]

[1]Facebook AI Research
[2]University College London
[3]New York University

Presenter: Shiwei Zhang



Inference
Pre-training
Fine-tuning
2024
2023
2022
2020
GPT-4
ChatGPT
GPT-3
GenRead
CREA-ICL
SUGRE
FABULA
ITRG
TOC
InstructRetro
Self-RAG
RA-DIT
IRCOT
RECITE
Raven
SelfMem
COK
Filter-Reranker
REPLUG
PKG
PGRA
SANTA
AAR
KAR
PRCA
UPRISE
Retro++
ITER-RETGEN
ICRALM
KnowledGPT
DSP
GraphToolformer
Toolformer
COG
Atlas
RETRO
FiD
DPR
RAG
REALM
KNN-LLM
Retrieval—Augmented Generation
Augmentation Stage
Pre-training
Fine-tuning
Inference
Augmentation Source
Unstructured Data
Structured Data
LLM Generated Content
Augmentation Process
Once Retrieval
Iterative Retrieval
Adaptive Retrieval
Recursive Retrieval
* Borderless represents Other Processes

Content

Introduction

Motivation

Pre-trained neural language models generate content based on parameterized implicit knowledge base. Such models have several downsides:

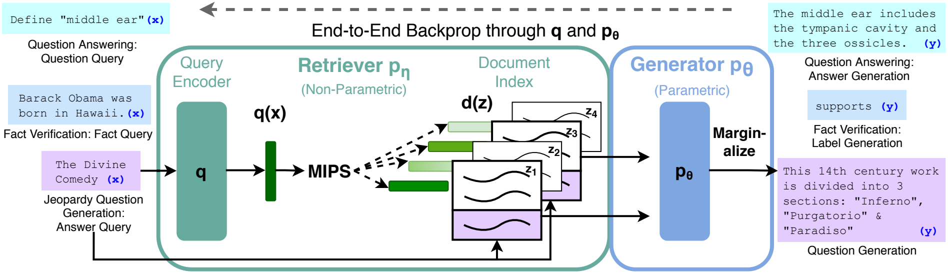
- They cannot easily expand or revise their memory.
- They cannot straightforwardly provide insights into the predictions.
- They may produce “hallucinations”.

Hybrid models that combine parametric memory with non-parametric (i.e. retrieval-based) memories may address these issues.

Overview

Retriever $p_\eta(z|x)$ returns (top-K) probabilities over documents for query x.

Generator $p_\theta(y_i|x, z, y_{1:i-1})$ 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.



Models

Marginalization

RAG treats the retrieved document as a latent variable and proposes two models to marginalize over the latent documents in different ways to produce a distribution over generated text.

$$\left.\begin{array}{l} p_\eta(z|x) \\ p_\theta(y_i|x, z, y_{1:i-1}) \end{array}\right\} \overset{?}{\Rightarrow} p(y|x)$$

RAG-Sequence Model

The RAG-Sequence model uses the same retrieved document to generate the complete sequence.

$$p_{\text{RAG-Sequence}}(y|x) \approx \sum_{z \in \text{top-k}(p(\cdot|x))} p_\eta(z|x) p_\theta(y|x, z)$$

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

RAG-Token Model

The RAG-Token model draws a different latent document for each target token. This allows the generator to choose content from several documents when producing an answer.

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

Retriever: DPR

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

$\mathbf{d}(z)$ is a vector representation of a document produced by a BERT model and $\mathbf{q}(x)$ is a vector representation of the query produced by another BERT model. Calculating $\text{top-k}(p_\eta(\cdot|x))$ is a Maximum Inner Product Search (MIPS) problem, which can be approximately solved in sub-linear time.

Generator: BART

BART is a seq2seq transformer model with 400M parameters. RAG concatenates the input x and the retrieved document z to produce the input for BART.

Training and Inference

Training

Given a fine-tuning training corpus of input/output pairs (x_j, y_j), the retriever and generator are jointly trained by minimizing the negative marginal log-likelihood $\sum_j -\log p(y_j|x_j)$.

The document encoder BERT_d is not updated during training as it is costly to do so (the document index needs to be updated as the model changes).

Decoding - RAG-Token

The RAG-Token model can be seen as a standard autoregressive seq2seq generator with transition probability:

$$p'_\theta(y_i|x, y_{i:i-1}) = \sum_{z \in \text{top-k}(p(\cdot|x))} p_\eta(z_i|x) p_\theta(y_i|x, z_i, y_{1:i-1})$$

Standard beam-search decoder can be used to sample the output.

Decoding - RAG-Sequence

For RAG-Sequence, RAG runs beam search for each document z, scoring each hypothesis using $p_\theta(y_i|x, z, y_{1:i-1})$ and yielding a set of hypotheses Y. Some of the hypotheses may not appear in the beams of all documents.

If a hypothesis y does not appear in a beam with document z, there are two options. The first option is to run an additional forward pass to get $p_\theta(y_i|x, z, y_{1:i-1})$. This is refered to as "Thorough Decoding". The other option is to assume $p_\theta(y|x, z_i) \approx 0$ if y was not generated during beam search for x, z_i. This is refered to as "Fast Decoding".

Experiments

Setup

- **Non-parametric knowledge**: Dec. 2018 Wikipedia dump split into 100 word chunks, totaling 21M documents.
- **MIPS solver**: FAISS with Hierarchical Navigable Small World approximation.
- **Hyper-parameters**: $k \in \{5, 10\}$ when retrieving the top-k documents.

Open-domain Question Answering

The four columns corresponds to four datasets.

	Model	NQ	TQA	WQ	CT
Closed Book	T5-11B [52]	34.5	- /50.1	37.4	-
	T5-11B+SSM[52]	36.6	- /60.5	44.7	-
Open Book	REALM [20]	40.4	- / -	40.7	46.8
	DPR [26]	41.5	**57.9**/ -	41.1	50.6
	RAG-Token	44.1	55.2/66.1	**45.5**	50.0
	RAG-Seq.	**44.5**	56.8/**68.0**	45.2	**52.2**

RAG can generate correct answers even if it is not in any retrieved document, where extractive models would score 0%.

Abstractive Question Answering

This task consists of questions, ten gold passages retrieved from a search engine for each question, and a full sentence answer annotated from the retrieved passages.

RAG does not use the gold passages and relies only on its parametric and non-parametric (Wikipedia) knowledges.

<table>
<tr><th rowspan="2">Model</th><th colspan="2">Jeopardy</th><th colspan="2">MSMARCO</th><th>FVR3</th><th>FVR2</th></tr>
<tr><th>B-1</th><th>QB-1</th><th>R-L</th><th>B-1</th><th colspan="2">Label Acc.</th></tr>
<tr><td>SotA</td><td>-</td><td>-</td><td>49.8*</td><td>49.9*</td><td>76.8</td><td>92.2*</td></tr>
<tr><td>BART</td><td>15.1</td><td>19.7</td><td>38.2</td><td>41.6</td><td>64.0</td><td>81.1</td></tr>
<tr><td>RAG-Tok.</td><td>17.3</td><td>22.2</td><td>40.1</td><td>41.5</td><td rowspan="2">72.5</td><td rowspan="2">89.5</td></tr>
<tr><td>RAG-Seq.</td><td>14.7</td><td>21.4</td><td>40.8</td><td>44.2</td></tr>
</table>

Jeopardy Question Generation

Jeopardy is about guessing an entity from a fact about that entity.

Model	Jeopardy B-1	Jeopardy QB-1	MSMARCO R-L	MSMARCO B-1	FVR3 Label Acc.	FVR2 Label Acc.
SotA	-	-	**49.8***	**49.9***	**76.8**	**92.2***
BART	15.1	19.7	38.2	41.6	64.0	81.1
RAG-Tok.	**17.3**	**22.2**	40.1	41.5	72.5	89.5
RAG-Seq.	14.7	21.4	40.8	44.2		

Jeopardy questions often contain two separate pieces of information, and RAG-Token may perform best because it can generate responses that combine content from several documents.

Fact Verification

This task requires classifying a claim is supported or reduted by Wikipedia, or whether there is not enough information.

<table>
<tr><th rowspan="2">Model</th><th colspan="2">Jeopardy</th><th colspan="2">MSMARCO</th><th>FVR3</th><th>FVR2</th></tr>
<tr><th>B-1</th><th>QB-1</th><th>R-L</th><th>B-1</th><th colspan="2">Label Acc.</th></tr>
<tr><td>SotA</td><td>-</td><td>-</td><td>49.8*</td><td>49.9*</td><td>76.8</td><td>92.2*</td></tr>
<tr><td>BART</td><td>15.1</td><td>19.7</td><td>38.2</td><td>41.6</td><td>64.0</td><td>81.1</td></tr>
<tr><td>RAG-Tok.</td><td>17.3</td><td>22.2</td><td>40.1</td><td>41.5</td><td rowspan="2">72.5</td><td rowspan="2"><u>89.5</u></td></tr>
<tr><td>RAG-Seq.</td><td>14.7</td><td>21.4</td><td><u>40.8</u></td><td><u>44.2</u></td></tr>
</table>

Conclusion

Conclusion

Strength:

- Addresses important problems.
- General and relatively simple formulation.

Limitation (and Oppotunities):

- Does not actually solve the hallucination problem.
- Needs to run k times more inference passes during generation.
- The input x needs to be additionally processed by another model.
- The retrieving process is likely to be disk-IO intensive or memory demanding.

Thank you!