Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

Research Report for the Cryptonite NLP Taskphase

By Pratham Shah, Research AI JTP 240905614

- Authors: 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
- Published: 2020
- PDF: https://arxiv.org/pdf/2005.11401.pdf

This report briefly described Retrieval-Augmented Generation (RAG) models, approaches which combine large language models with external, non-parametric knowledge sources - most notably a dense index of Wikipedia documents.

RAG is specifically designed for knowledge-intensive natural language processing (NLP) tasks that require access to up-to-date or detailed factual information that can exceed what is stored in a model's parameters.

Introduction

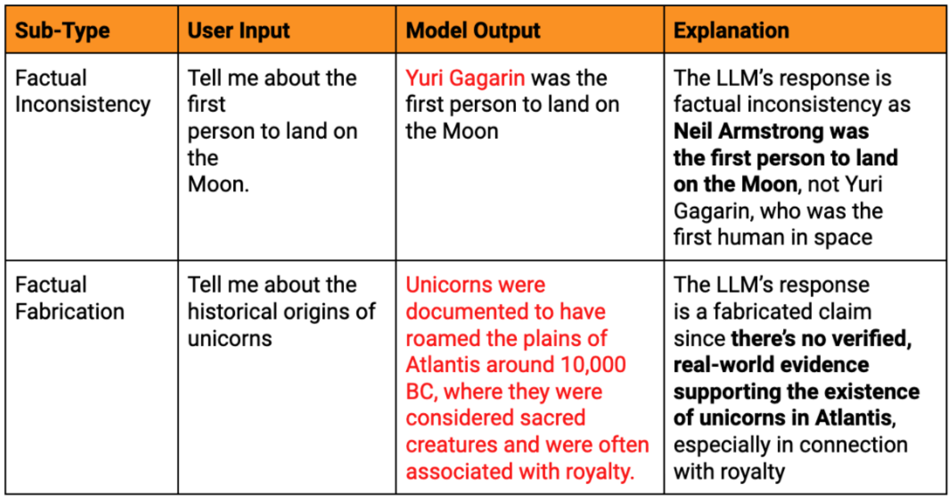
Before the development of Retrieval-Augmented Generation (RAG) models, large language models and neural sequence-to-sequence architectures faced persistent limitations when applied to knowledge-intensive tasks such as open-domain question answering, fact verification, and knowledge-grounded dialogue.

Challenges with Pre-RAG Approaches included:

- *Static Knowledge Storage:* Traditional models encode information into their parameters during training. This makes their knowledge static and difficult to update without expensive retraining.

Modern systems like ChatGPT apply RAG-like techniques (e.g., web search) to stay current. ChatGPT's pre-trained knowledge cutoff date was June 2024.

- *Limited Factual Recall:* The capacity to capture all necessary knowledge in model weights is restricted, often resulting in incorrect or "hallucinated" facts especially for rare or updated information.



Sub-Type	User Input	Model Output	Explanation
Factual Inconsistency	Tell me about the first person to land on the Moon.	Yuri Gagarin was the first person to land on the Moon	The LLM's response is factual inconsistency as **Neil Armstrong was the first person to land on the Moon**, not Yuri Gagarin, who was the first human in space
Factual Fabrication	Tell me about the historical origins of unicorns	Unicorns were documented to have roamed the plains of Atlantis around 10,000 BC, where they were considered sacred creatures and were often associated with royalty.	The LLM's response is a fabricated claim since **there's no verified, real-world evidence supporting the existence of unicorns in Atlantis,** especially in connection with royalty

LLM Hallucinations

- *Poor Citing and Verification:* Standard language models offer little to no transparency regarding the source of generated content. There is no way to trace an answer back to a verifiable document.
- *Knowledge Updating Difficulties*: To keep models current, the entire network must be retrained or fine-tuned with new data—a slow, computationally expensive, and often infeasible process for very large LLMs.
- *Factual Consistency*: Answers may lack external grounding, reducing confidence in their accuracy, especially for critical applications like fact checking or during emergencies. (refer to the above example, where an LLM hallucinated to think Yuri Gagarin was the 1st to land on the moon.)

These issues restricted purely parametric models in tasks that needed up-to-date, precise, and explainable/citable results.

Problem Statement

Despite huge advances in natural language generation, traditional neural models struggle to reliably provide accurate and up-to-date knowledge, explain their outputs with credible sources, and efficiently add new information.

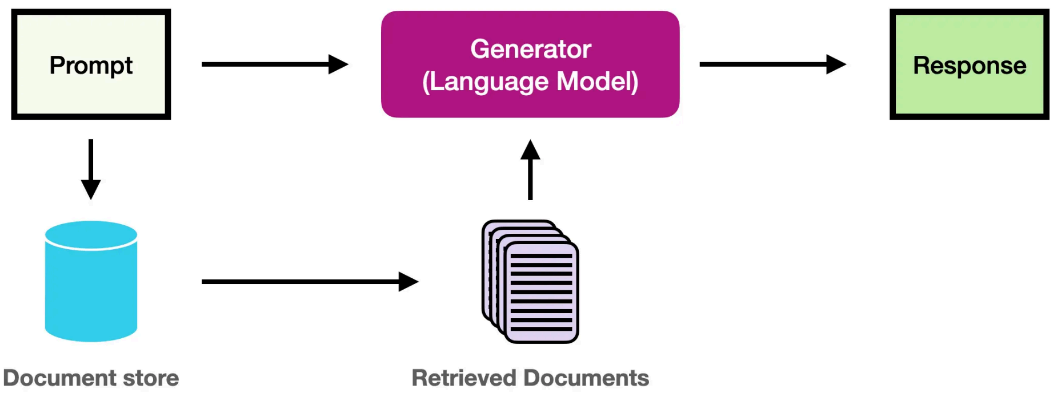
A need arises for systems that can dynamically retrieve relevant external documents to back up their responses, so that generated outputs remain correct, verifiable, and easy to update as knowledge evolves.

RAG models aim to integrate neural retrieval with generative architectures, combining flexible language modeling and live access to storehouses of external knowledge.

Retrieval Augmented Generation

RAG models augment standard sequence-to-sequence (seq2seq) language models with an explicit retrieval mechanism:

- *Parametric Memory:* A pre-trained neural seq2seq generator (e.g., BART).
- *Non-Parametric Memory:* An external, searchable document index (e.g., Wikipedia) accessed by a neural retriever (Dense Passage Retriever, DPR).



The workflow for a standard RAG Application is:

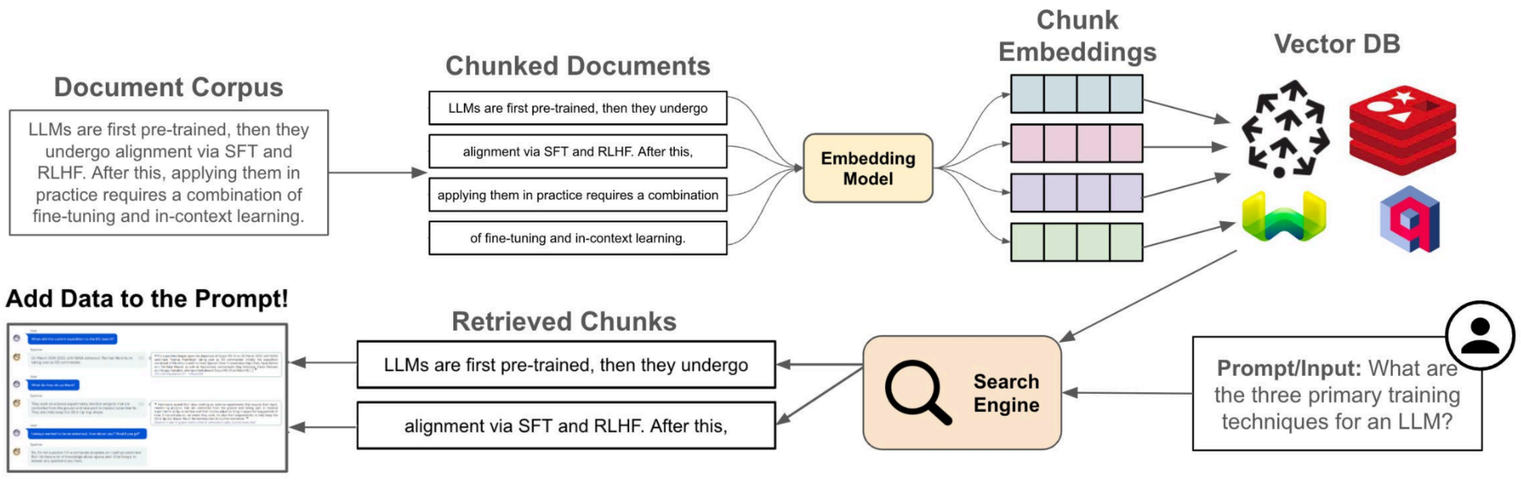
1. *Input:* A query (question, prompt, or statement) is given.
2. *Retrieval:* The retriever identifies the most relevant documents from the external knowledge index. This content store can be static or dynamically changing.
3. *Generation:* The seq2seq model generates a response, conditioned on both the input and the retrieved documents.
4. *Marginalization:* The final output is produced by considering multiple relevant documents—either using the same document for all output tokens (*RAG-Sequence*) or allowing different documents for each generated token (*RAG-Token*).

Main Contributions

- Unified Model: RAG can be fine-tuned on any seq2seq task and supports both generation and classification tasks requiring external knowledge retrieval.

- Two Variants: RAG-Sequence and RAG-Token allow for choices in output generation given cost/time concerns.
- End-to-End Training: Both the retriever and generator are trained jointly, but without direct supervision on which document to retrieve.
- Knowledge Updating: The external memory can be replaced or updated without retraining the entire model, letting the system quickly adapt to new information.

Components of Retrieval, Augmentation and Generation


Document Corpus
LLMs are first pre-trained, then they undergo alignment via SFT and RLHF. After this, applying them in practice requires a combination of fine-tuning and in-context learning.
Chunked Documents
LLMs are first pre-trained, then they undergo
alignment via SFT and RLHF. After this,
applying them in practice requires a combination
of fine-tuning and in-context learning.
Embedding Model
Chunk Embeddings
Vector DB
Add Data to the Prompt!
Retrieved Chunks
LLMs are first pre-trained, then they undergo
alignment via SFT and RLHF. After this,
Search Engine
Prompt/Input: What are the three primary training techniques for an LLM?

1. Input:

A query is inputted from the user. The context of this query is to be understood and relevant documents are to be retrieved.

RAG uses a Dense Passage Retriever (DPR) to process the query.

- The query is first tokenized using a BERT tokenizer (WordPiece or similar).
- The tokenized sequence is passed through the Query Encoder — a BERT-based transformer.
- The output is a fixed-size dense vector representation of the query:
  q vec = Encoder q (tokenized query)

This is a 768-dimensional vector if using base BERT.

2. Vector-Similarity Based Document Search:

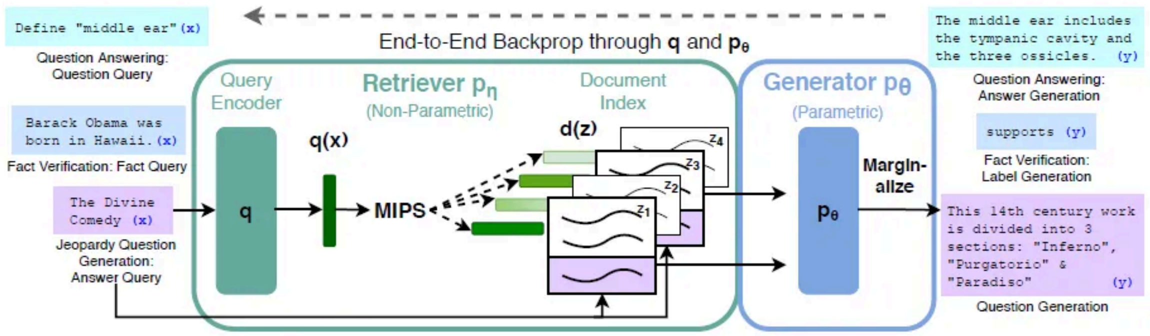

Define "middle ear" (x)
Question Answering: Question Query
Barack Obama was born in Hawaii. (x)
Fact Verification: Fact Query
The Divine Comedy (x)
Jeopardy Question Generation: Answer Query
End-to-End Backprop through q and p_θ
Query Encoder
Retriever p_η
(Non-Parametric)
Document Index
q(x)
d(z)
q
MIPS
z_4
z_3
z_2
z_1
Generator p_θ
(Parametric)
p_θ
Marginalize
The middle ear includes the tympanic cavity and the three ossicles. (y)
Question Answering: Answer Generation
supports (y)
Fact Verification: Label Generation
This 14th century work is divided into 3 sections: "Inferno", "Purgatorio" & "Paradiso" (y)
Question Generation

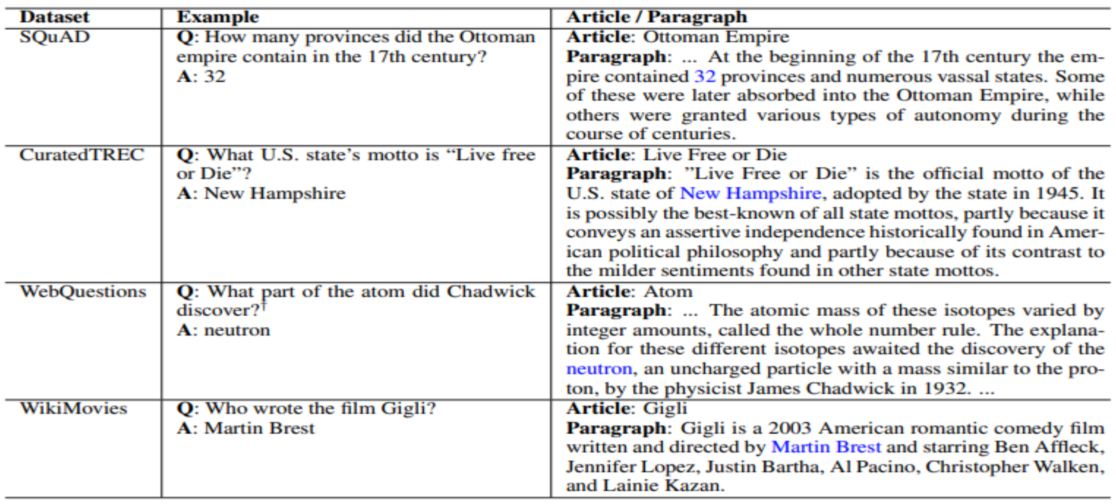
Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder* + *Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x, we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i. For final prediction y, we treat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

- The encoded query vector q vec is compared with all pre-encoded document vectors dk in the index.
- A dot product is used to score each document *score (q , dk) = q vec · dk*
- The top K documents with the highest similarity scores are selected.
- This is done efficiently using FAISS or a similar Approximate Nearest Neighbor (ANN) search library.

Each retrieved document includes:

- The original passage text (to condition the generator)
- Its relevance score P (dk ∣ x)

3. Generator Similarity Detection and Output generation:



Dataset	Example	Article / Paragraph
SQuAD	**Q**: How many provinces did the Ottoman empire contain in the 17th century? **A**: 32	**Article**: Ottoman Empire **Paragraph**: ... At the beginning of the 17th century the empire contained 32 provinces and numerous vassal states. Some of these were later absorbed into the Ottoman Empire, while others were granted various types of autonomy during the course of centuries.
CuratedTREC	**Q**: What U.S. state's motto is "Live free or Die"? **A**: New Hampshire	**Article**: Live Free or Die **Paragraph**: "Live Free or Die" is the official motto of the U.S. state of New Hampshire, adopted by the state in 1945. It is possibly the best-known of all state mottos, partly because it conveys an assertive independence historically found in American political philosophy and partly because of its contrast to the milder sentiments found in other state mottos.
WebQuestions	**Q**: What part of the atom did Chadwick discover?[†] **A**: neutron	**Article**: Atom **Paragraph**: ... The atomic mass of these isotopes varied by integer amounts, called the whole number rule. The explanation for these different isotopes awaited the discovery of the neutron, an uncharged particle with a mass similar to the proton, by the physicist James Chadwick in 1932. ...
WikiMovies	**Q**: Who wrote the film Gigli? **A**: Martin Brest	**Article**: Gigli **Paragraph**: Gigli is a 2003 American romantic comedy film written and directed by Martin Brest and starring Ben Affleck, Jennifer Lopez, Justin Bartha, Al Pacino, Christopher Walken, and Lainie Kazan.

Each retrieved document is concatenated with the original query to form an input for the generator. A custom system prompt can be given to the LLM to guide its generation, eg.: “Read the retrieved documents and answer the users questions based on them”.

An example query “Who wrote the Great Gatsby?” might return these top k (2) chunks:

d_1: "F. Scott Fitzgerald was an American novelist and the author of The Great Gatsby, published in 1925."
d_2: "The Great Gatsby is a 1925 novel that explores themes of wealth and society in the Jazz Age in the United States."

The BART generator assembles one input per retrieved document, combining the query and the document: (note that CLS may be replaced by <start> and SEP may be replaced by </s>)

[CLS] Who wrote The Great Gatsby? [SEP] F. Scott Fitzgerald was an American novelist and the author of The Great Gatsby, published in 1925.

[CLS] Who wrote The Great Gatsby? [SEP] The Great Gatsby is a 1925 novel that explores themes of wealth and society in the Jazz Age in the United States.

Now, BART understands this context. It has been trained on Q&A and generation tasks, and can answer questions / do tasks based on this applied context. It generates responses using both: 1. The input 2. The previously generated output. Here, with context, is the autoregressive stream:

Feature	RAG-Sequence	RAG-Token
Retrieval Granularity	One document for full sequence	Different document for each token
Marginalization	Over full sequence per doc	Per token over all documents
Output Flexibility	Less flexible, more consistent	More flexible, token-level document relevance
Best Use Case	When one document contains the full answer	When facts are distributed across multiple documents

Time step	Context y<t	Target yt	Top logits (softmaxed)
1	[BOS]	"F"	"F": 0.81, "He": 0.05, "The": 0.03
2	[BOS, "F"]	"."	".": 0.76, "Scott": 0.12, ",": 0.05
3	[BOS, "F", "."]	"Scott"	"Scott": 0.93, "Fitzgerald": 0.02
4	[BOS, "F", ".", "Scott"]	"Fitzgerald"	"Fitzgerald": 0.88, "was": 0.05

- **RAG-Sequence**

$$P(y \mid x) = \sum P(y \mid x, dk) \cdot P(dk \mid x)$$

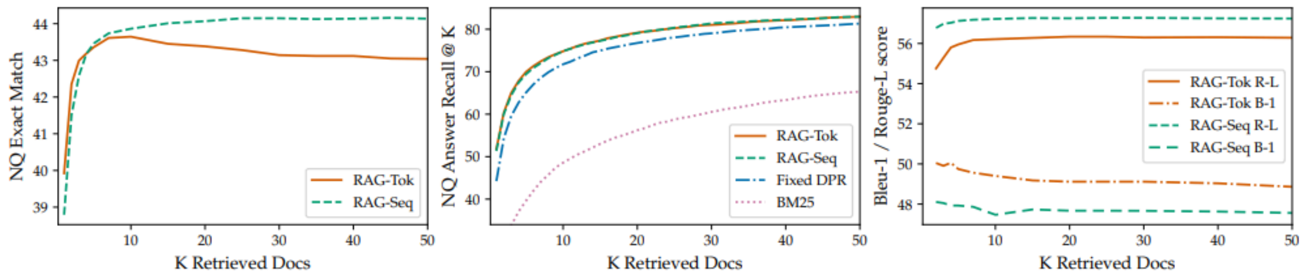
- For input x (the user's question), the model calculates document relevance $P(dk \mid x)$.
- It then computes generation probability $P(y \mid x, dk)$ for each document and weights each document's response accordingly.
- The final output is a relevance-weighted mixture across retrieved documents.

- **RAG-Token**

$$P(yt \mid y < t, x) = \sum P(yt \mid y < t, x, dk) \cdot P(dk \mid x) \text{ over } k$$

- At each token position t, the model:
  - Computes document relevance: $P(dk \mid x)$
  - Calculates the likelihood of generating token yt given prior tokens, the question, and each document: $P(yt \mid y < t, x, dk)$
- For each output token, token predictions across documents are merged, weighted by document relevance.
- This allows information from multiple documents to contribute at a finer granularity.

The more documents the RAG model processes, the higher are its scores on key datasets:



In summary:

- The RAG process integrates dense retrieval (using DPR and similarity search) with generative models, creating responses that are both relevant and grounded in external knowledge.
- RAG-Sequence is best when answers are contained within single documents, while RAG-Token is more flexible and better when information is across several sources.

Impacts and Scores

Table 1 shows exact match scores for QA tasks across various models. Compares "closed book" (no retrieval) with "open book" (retrieval-augmented) systems. It tabulates results on benchmarks like NQ, TQA, WQ, and CT, highlighting performance differences.

Table 2 presents test scores for generation and classification across datasets like Jeopardy, MS-MARCO, and FEVER. It benchmarks models on text generation (BLEU) and fact verification accuracy, distinguishing systems that use gold context from those that do not.

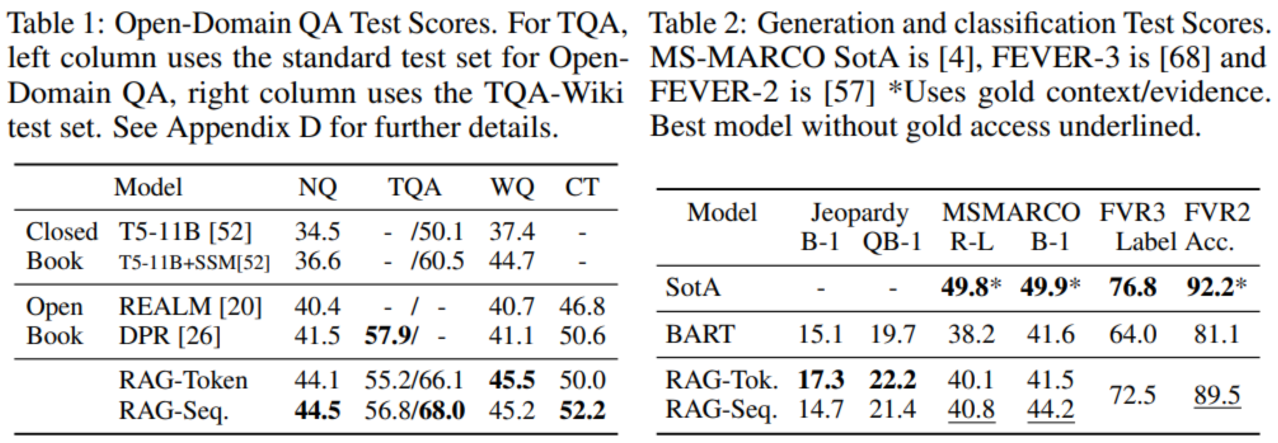


Table 1: Open-Domain QA Test Scores. For TQA, left column uses the standard test set for Open-Domain QA, right column uses the TQA-Wiki test set. See Appendix D for further details.

	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**

Table 2: Generation and classification Test Scores. MS-MARCO SotA is [4], FEVER-3 is [68] and FEVER-2 is [57] *Uses gold context/evidence. Best model without gold access underlined.

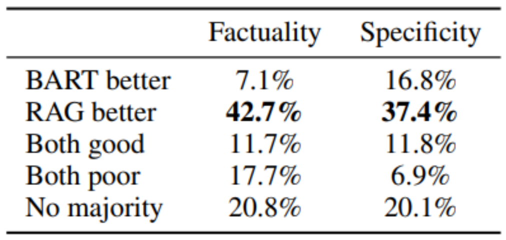
Model	Jeopardy		MSMARCO		FVR3	FVR2
	B-1	QB-1	R-L	B-1	Label Acc.	
SotA	-	-	**49.8***	**49.9***	**76.8**	**92.2***
BART	15.1	19.7	38.2	41.6	64.0	81.1
RAG-Tok.	**17.3**	**22.2**	40.1	41.5	72.5	89.5
RAG-Seq.	14.7	21.4	40.8	44.2		

Table 4 summarizes human evaluation of generated Jeopardy questions.

Table 5 reports the ratio of distinct to total tri-grams generated by models on MSMARCO and Jeopardy tasks. Higher values indicate greater lexical diversity in generated texts, compared across gold references and multiple models.

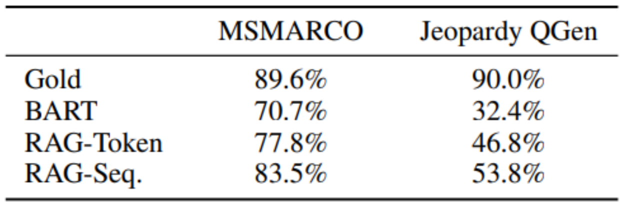
Table 6 shows ablation experiments replacing or freezing retrieval components and swapping retrieval methods.

Table 4: Human assessments for the Jeopardy Question Generation Task.



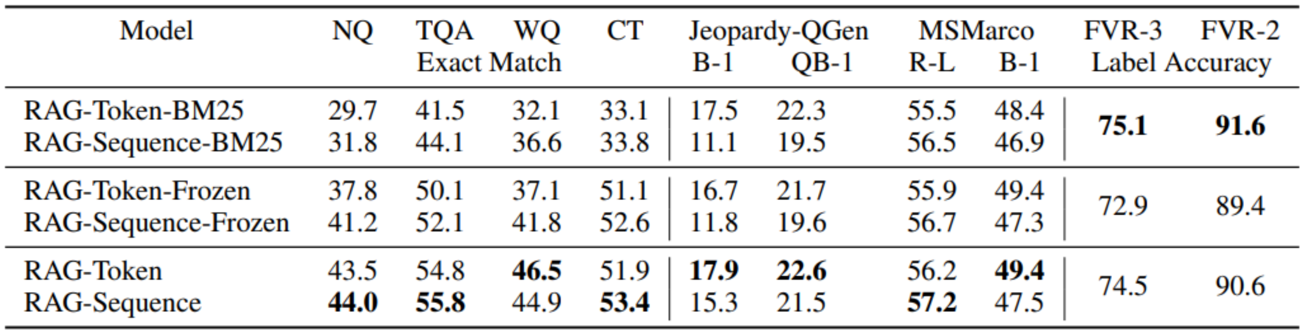
	Factuality	Specificity
BART better	7.1%	16.8%
RAG better	**42.7%**	**37.4%**
Both good	11.7%	11.8%
Both poor	17.7%	6.9%
No majority	20.8%	20.1%

Table 5: Ratio of distinct to total tri-grams for generation tasks.



	MSMARCO	Jeopardy QGen
Gold	89.6%	90.0%
BART	70.7%	32.4%
RAG-Token	77.8%	46.8%
RAG-Seq.	83.5%	53.8%

Table 6: Ablations on the dev set. As FEVER is a classification task, both RAG models are equivalent.



Model	NQ	TQA	WQ	CT	Jeopardy-QGen		MSMarco		FVR-3	FVR-2
		Exact Match			B-1	QB-1	R-L	B-1	Label Accuracy	
RAG-Token-BM25	29.7	41.5	32.1	33.1	17.5	22.3	55.5	48.4	**75.1**	**91.6**
RAG-Sequence-BM25	31.8	44.1	36.6	33.8	11.1	19.5	56.5	46.9		
RAG-Token-Frozen	37.8	50.1	37.1	51.1	16.7	21.7	55.9	49.4	72.9	89.4
RAG-Sequence-Frozen	41.2	52.1	41.8	52.6	11.8	19.6	56.7	47.3		
RAG-Token	43.5	54.8	**46.5**	51.9	**17.9**	**22.6**	56.2	**49.4**	74.5	90.6
RAG-Sequence	**44.0**	**55.8**	44.9	**53.4**	15.3	21.5	**57.2**	47.5		

Experimental Findings

- Open-Domain Question Answering: RAG achieves state-of-the-art results on popular datasets such as Natural Questions, TriviaQA, WebQuestions, and CuratedTrec.
- Abstractive QA and Generation: RAG outperforms strong baselines like BART in factuality, specificity, and diversity when generating answers and questions, especially in tasks such as Jeopardy question generation.
- Fact Verification: In classification tasks like FEVER, RAG matches/surpasses state-of-the-art accuracy without direct evidence supervision.
- Ablation studies show that learned retrieval is effective, and that swapping out the document index allows quick model updation - which is significantly harder than on fully parametric models.

Advantages and Impact

- More Factual and Interpretable Outputs: Grounding generations in retrieved documents reduces hallucinations and increases trust in model predictions.
- Easier Knowledge Updates: The non-parametric memory means the model can be kept up-to-date simply by simply refreshing the document index.
- Greater Diversity: RAG models generate more specific and diverse text than baseline seq2seq models.

Potential Applications

- Open-domain QA systems
- Fact-checking and verification
- Knowledge-grounded dialogue agents
- Any NLP application where factual correctness and provenance are critical

Limitations and Broader Impact

- Reliance on Quality of External Corpus: If the document index (such as Wikipedia) is outdated or biased, model outputs may inherit these issues.
- Ethical Considerations: While reducing hallucinations, RAG is still susceptible to misuse if the source knowledge is biased, falsified or manipulated.

Conclusion

RAG models represent a significant step toward robust, updatable, and interpretable language models for knowledge-intensive NLP tasks.

They blend neural generation with real-time neural retrieval from vast text corpora, setting state-of-the-art standards in Q&A, generation, verification and more.

****