RAG with transformers.

Introduction to AI Agents.

Outline

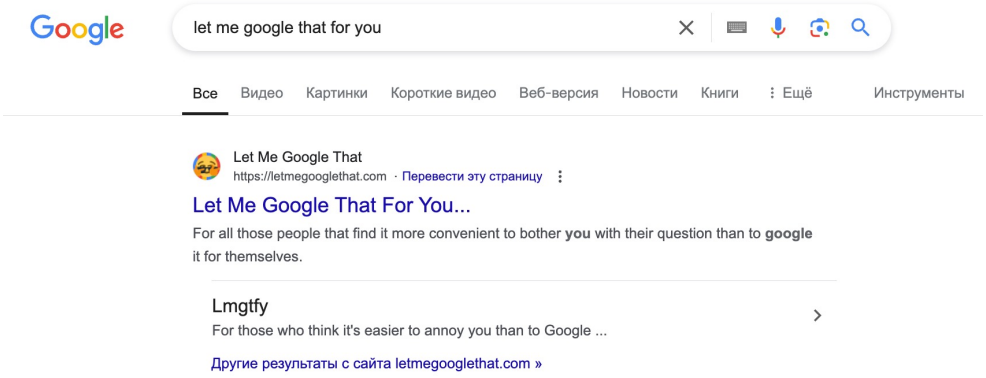
- Retriever-Augmented Generation (RAG)
- Retrieval
- Retrieval-Augmented Architectures
- RAG evaluation
- Intro to AI Agents
- AI Agents evaluation

Retriever-Augmented Generation

Motivation

Problem: LLMs are prone to lying and hallucinating.

Natural idea: give it ability to ”Google” factual information.



Solution: Retrieval-Augmented Generation (RAG).

Core idea: give LLM an access to some information database.

What is Retrieval-Augmented Generation (RAG)?

Retrieval-Augmented Generation (RAG) is:

- A method for **optimizing responses** generated by large language models (LLMs).
- It **combines text generation with information retrieval** from **a knowledge base**.
- It creates **more informative** and **accurate responses.**

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,

https://aíxiv.oíg/pdf/2005.11401

Why is RAG needed?

LLM challenges:

- May create “hallucinations” that are misleading.
- Rely on potentially outdated information.
- Ineffective when dealing with specific knowledge.
- Lack depth in specialized areas.
- Underdeveloped in the area of logical ability.
- Lack controllability.
- Cannot trace the source of knowledge.
- Cannot ensure confidentiality of data.
- The cost of their training is high.

How does RAG help?

- Improves accuracy and relevance using contextually significant knowledge from external sources.
- Can respond “I don’t know” when relevant information is not found instead of fabricating an answer.

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 knowl-

How does RAG help?

Question: *«У какой планеты в солнечной системе больше всего спутников?»* (ChatGPT vs NASA)

ChatGPT before RAG

ChatGPT

Jupiter holds the title for having the most moons in our solar system. As of my last update, it boasts a whopping 79 moons confirmed by NASA. Ganymede, one of Jupiter's moons, is the largest moon in the solar system, even bigger than the planet Mercury.

ChatGPT (gpt-3.5-tuíbo),12.05.2024
https://platform.openai.com/docs/models/gpt-3-5-tuíbo

MODEL	DESCRIPTION	CONTEXT WINDOW	TRAINING DATA
gpt-3.5-turbo-0125	New **Updated GPT 3.5 Turbo** The latest GPT-3.5 Turbo model with higher accuracy at responding in requested formats and a fix for a bug which caused a text encoding issue for non-English language function calls. Returns a maximum of 4,096 output tokens. Learn more.	16,385 tokens	Up to Sep 2021
gpt-3.5-turbo	Currently points to gpt-3.5-turbo-	16,385 tokens	Up to Sep 2021

Accoíding to the NASA/JPL Solaí System Dynamics team, the cuíient tally of moons oíbiting planets in ouí solaí system is 293: One moon foí Eaíth; two foí Maís; 95 at Jupiter; 146 at Satuín; 28 at Uíanus; 16 at Neptune; and five foí dwaíf planet Pluto.

NASA,12.05.2024
https://science.nasa.gov/solaí-system/moons/

How does RAG help?

Question: *«У какой планеты в солнечной системе больше всего спутников?»* (ChatGPT vs NASA)

ChatGPT before RAG

ChatGPT

Jupiter holds the title for having the most moons in our solar system. As of my last update, it boasts a whopping 79 moons confirmed by NASA. Ganymede, one of Jupiter's moons, is the largest moon in the solar system, even bigger than the planet Mercury.

ChatGPT (gpt-3.5-tuíbo),12.05.2024
https://platform.openai.com/docs/models/gpt-3-5-tuíbo

MODEL	DESCRIPTION	CONTEXT WINDOW	TRAINING DATA
gpt-3.5-turbo-0125	New **Updated GPT 3.5 Turbo** The latest GPT-3.5 Turbo model with higher accuracy at responding in requested formats and a fix for a bug which caused a text encoding issue for non-English language function calls. Returns a maximum of 4,096 output tokens. Learn more.	16,385 tokens	Up to Sep 2021
gpt-3.5-turbo	Currently points to gpt-3.5-turbo-	16,385 tokens	Up to Sep 2021

Accoíding to the NASA/JPL Solaí System Dynamics team, the cuíient tally of moons oíbiting planets in ouí solaí system is 293: One moon foí Eaíth; two foí Maís; 95 at Jupiter; 146 at Satuín; 28 at Uíanus; 16 at Neptune; and five foí dwaíf planet Pluto.

NASA,12.05.2024
https://science.nasa.gov/solaí-system/moons/

ChatGPT now

На сегодняшний день планетой с наибольшим количеством спутников в Солнечной системе является **Сатурн**. У него известно **145 спутников** (по состоянию на 2023 год), включая подтвержденные спутники и кандидаты. Это делает Сатурн рекордсменом по числу спутников среди планет Солнечной системы.

Для сравнения:

- **Юпитер** занимает второе место с **95 подтвержденными спутниками**.
- За Юпитером и Сатурном идут Уран и Нептун, у которых гораздо меньше спутников (27 и 14 соответственно).

RAG applications

Question-Answering Systems:

- *Enables models to understand query context and generate more informative answers.*

Text Summarization:

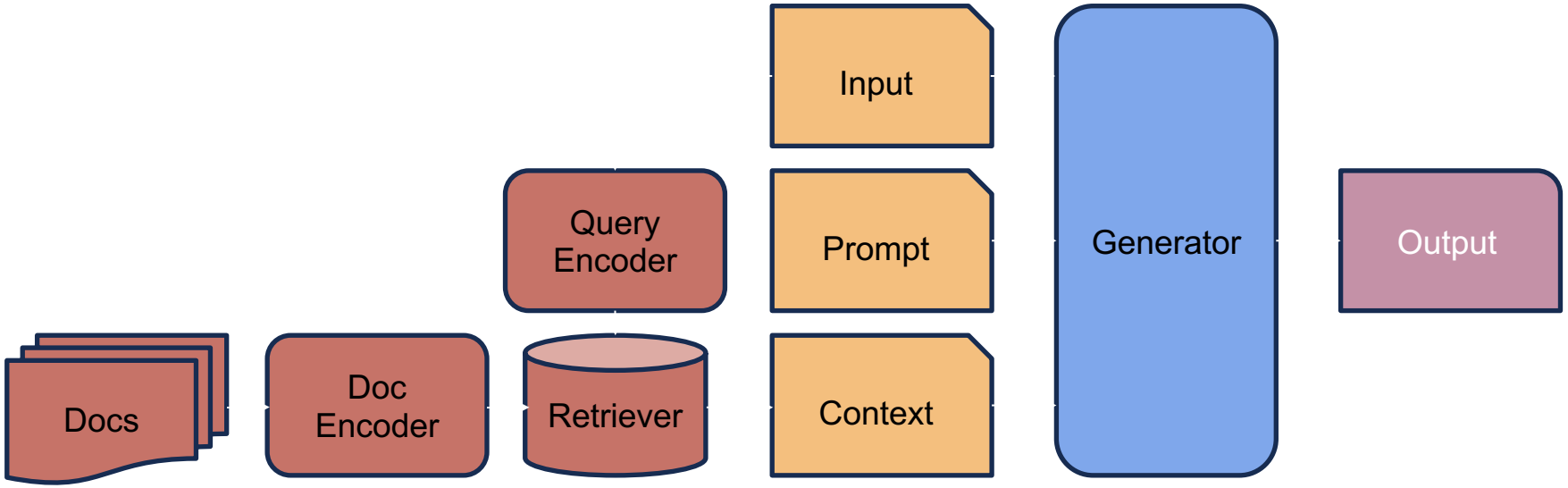
- *Incorporates knowledge base context to produce more complete and accurate summaries.*

Other use cases:

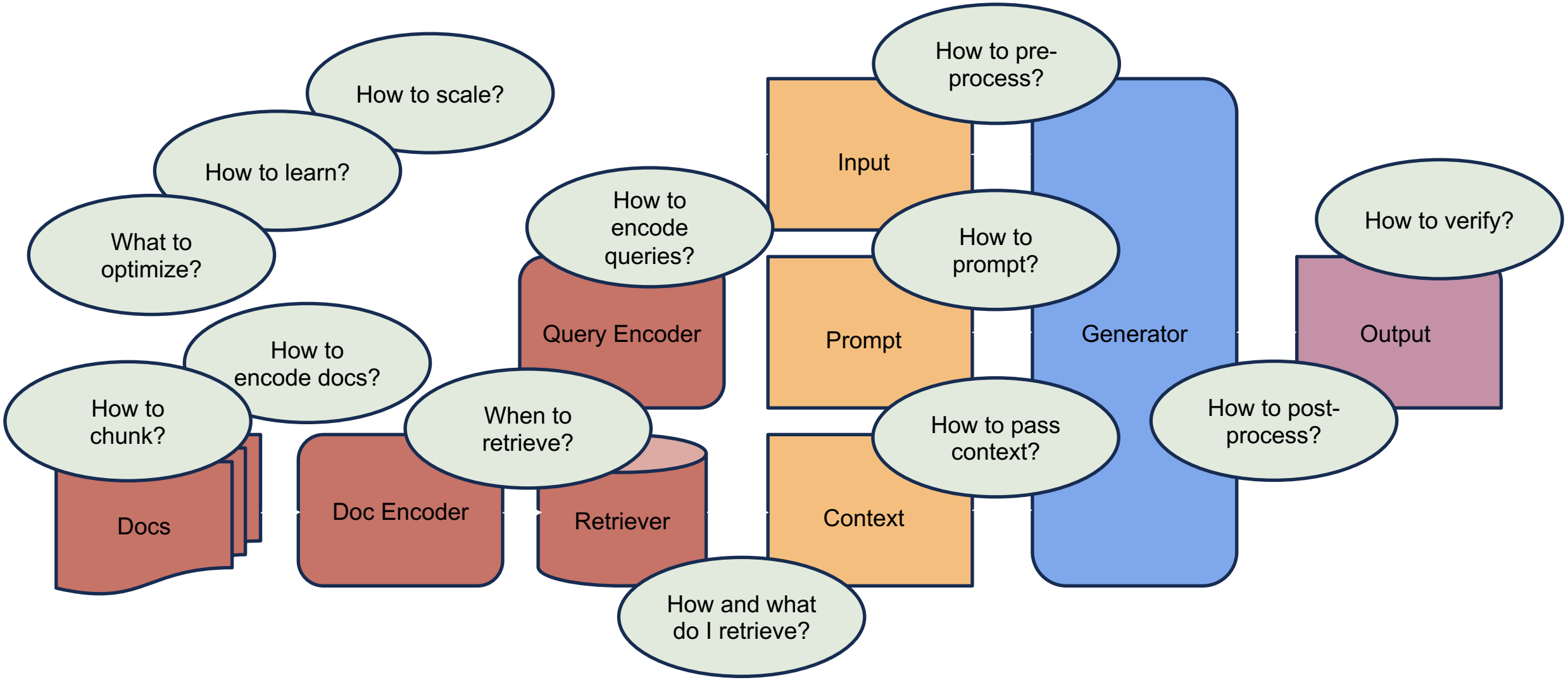
- *Any scenario where text generation relies on knowledge bases or any factual information.*

RAG concepts

- **Indexing:** Splits documents into fragments and builds a vector index with an encoder.
- **Retrieval:** Finds relevant document fragments based on semantic similarity.
- **Generation:** Produces answers based on retrieved context.

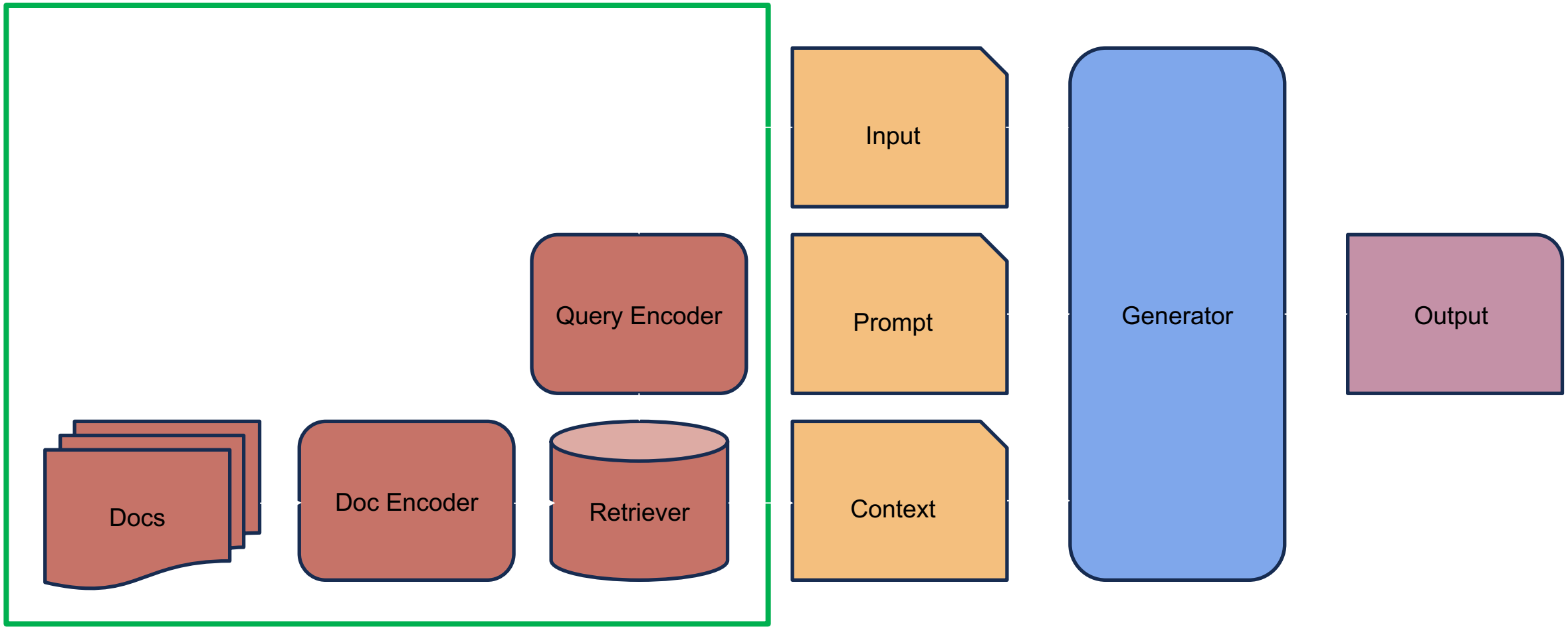


Not as simple as it may sound…



Retrieval

Information retrieval



How IR works?

Methods:

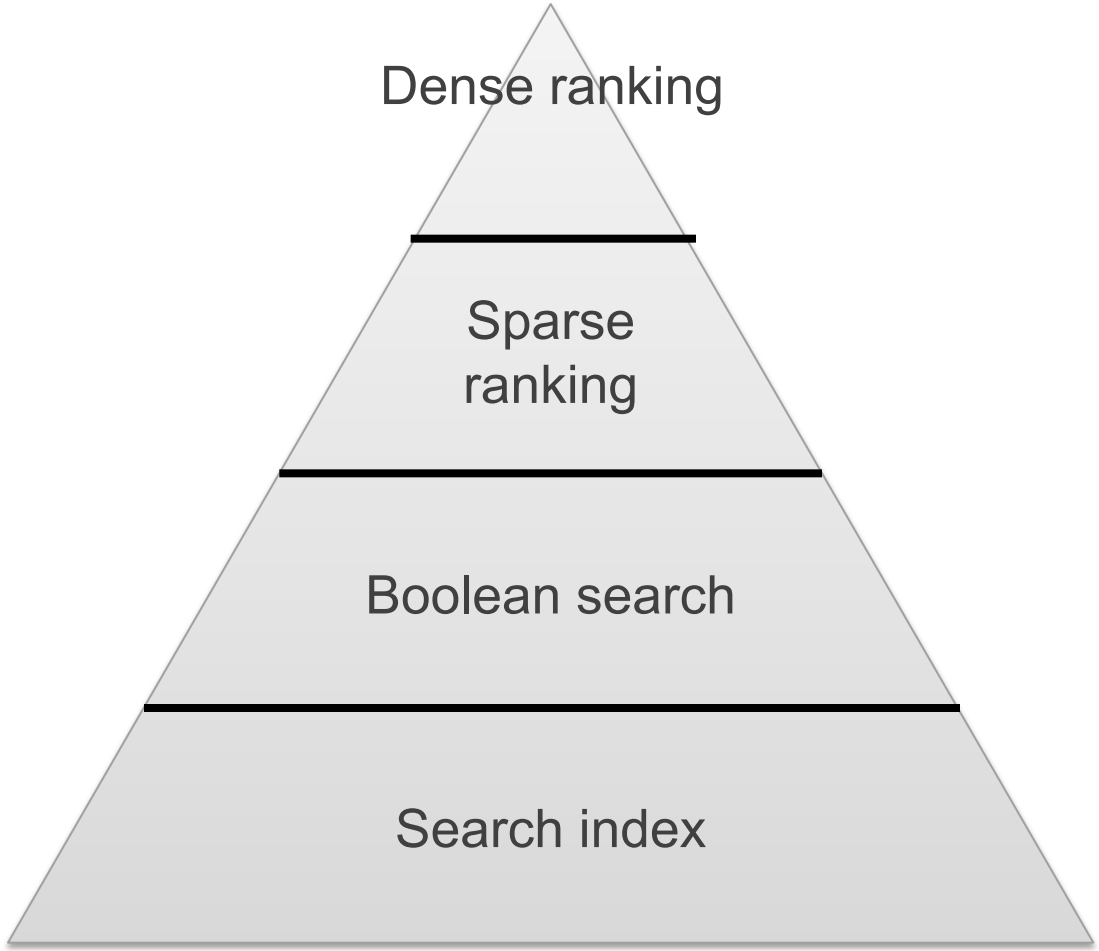
- Boolean search

Sparse vectors:

- TF-IDF
- BM25

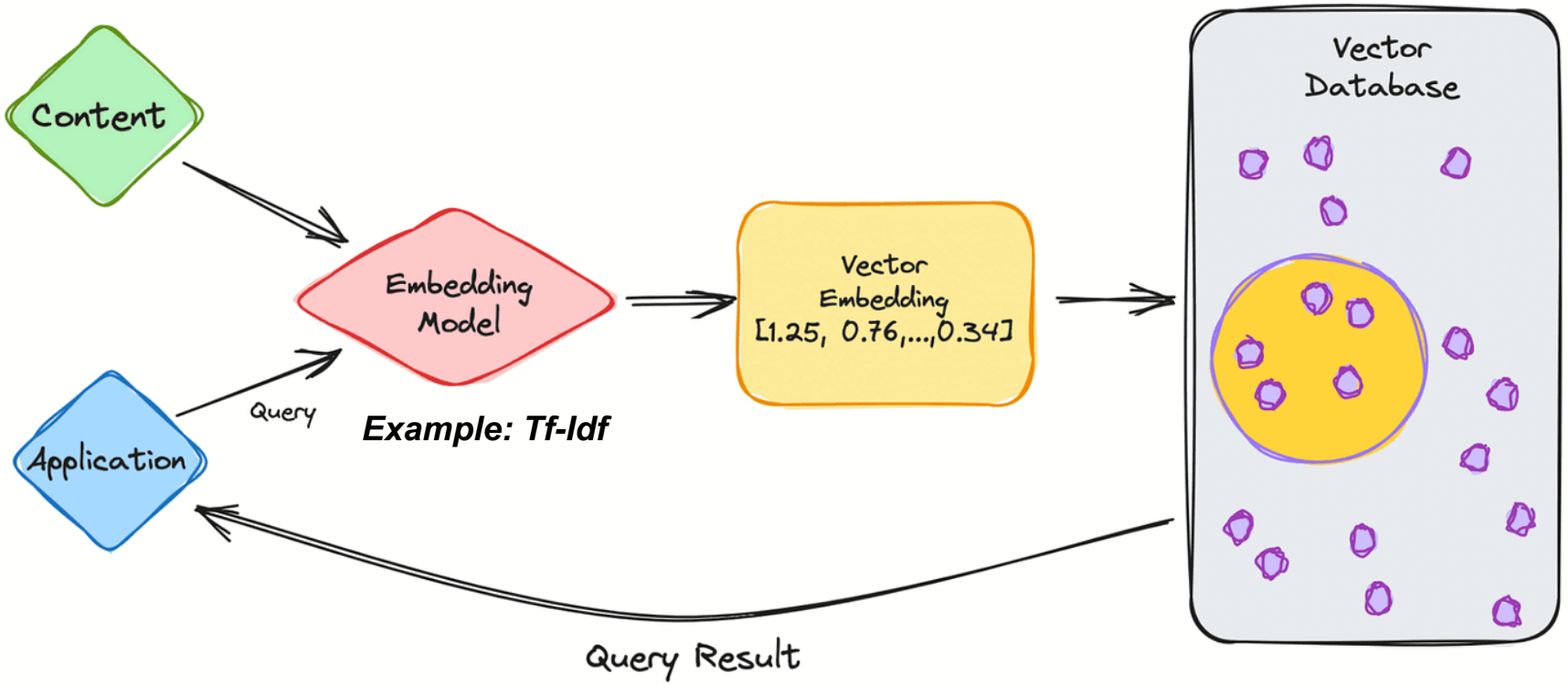
Dense vectors:

- encoders

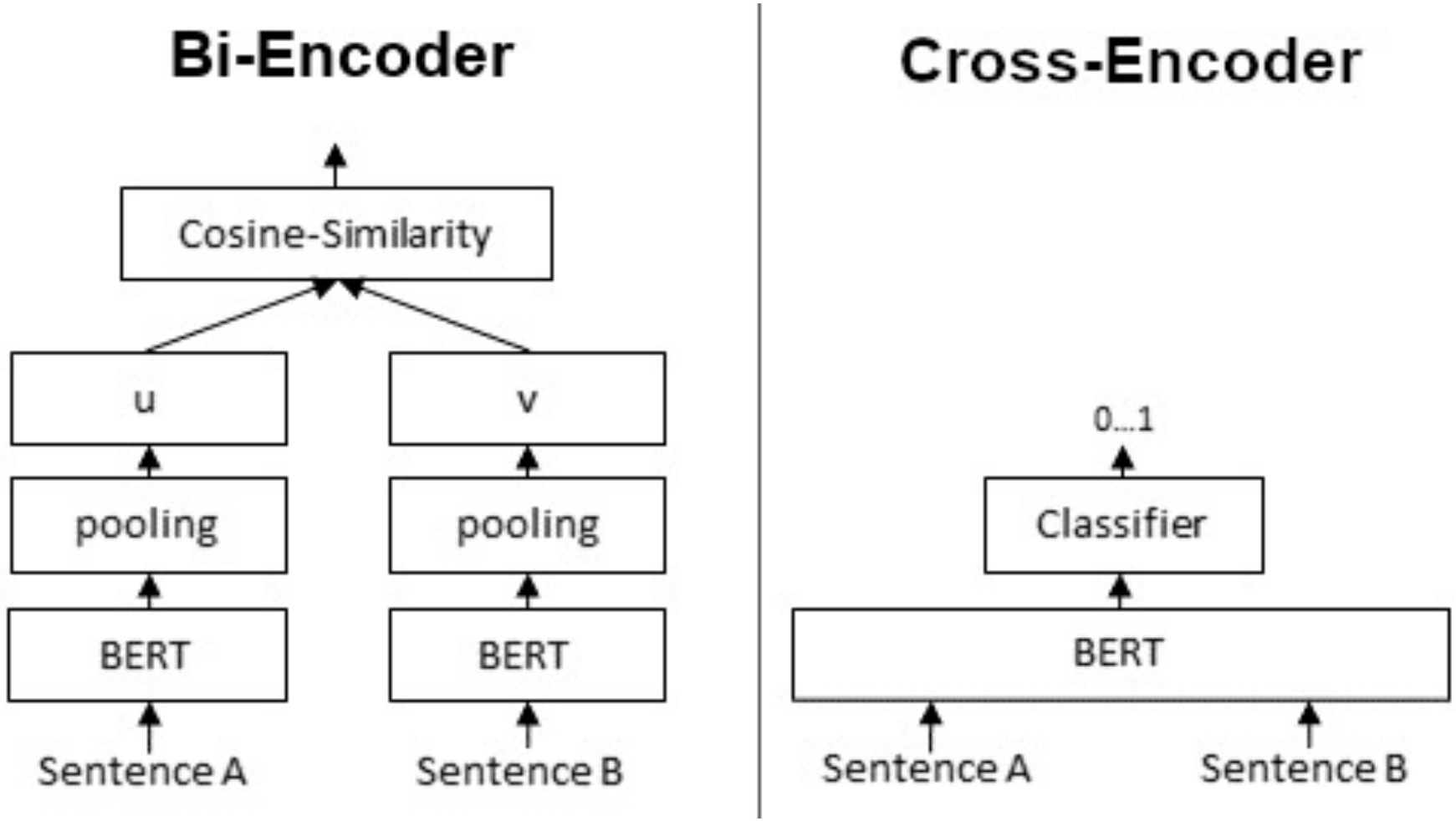


Vector database

Can efficiently find approximate nearest neighbors.



Encoder types



Encoder types

Bi-Encoders

Pros:

- Allows for caching - count everything once, then put it in the index afterward

Cons:

- Generally inferior to cross-encoder in terms of quality

Cross-Encoders

Pros:

- Generally superior to bi-encoder in terms of quality
- Good rerankers

Cons:

- No caching

Dense Passage Retrieval (DPR)

DPR is a bi-encoder model which uses different encoders for passages $E_P(.)$ and queries $E_Q(.)$.

Both encoders are BERT-based.

Use dot-product similarity to find similar documents.

$$\text{sim}(q, p) = E_Q(q)^\intercal E_P(p)$$

Dense Passage Retrieval for Open-Domain Question Answering

Vladimir Karpukhin[*], **Barlas Oğuz**[*], **Sewon Min**[†], **Patrick Lewis**,
Ledell Wu, Sergey Edunov, Danqi Chen[‡], **Wen-tau Yih**
Facebook AI [†]University of Washington [‡]Princeton University
{vladk, barlaso, plewis, ledell, edunov, scottyih}@fb.com
sewon@cs.washington.edu
danqic@cs.princeton.edu

https://arxiv.org/abs/2004.04906

Abstract

Retrieval in open-domain QA is usually imple-

Retrieval-Augmented Architectures

Training and inference

Training:

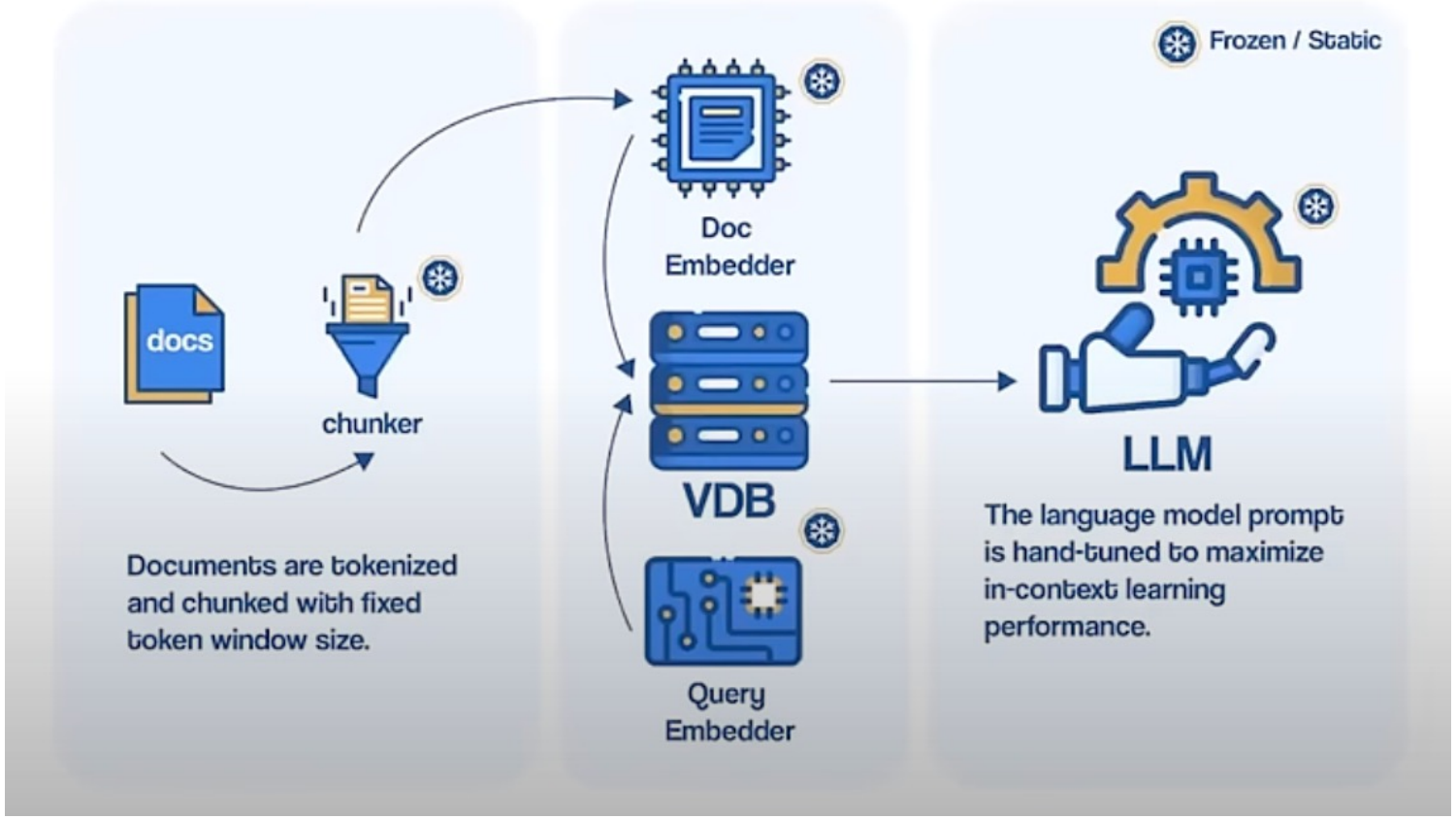
- generator?
- query encoder?
- document encoder?
- from scratch or fine-tune?

Inference:

- Same or another index?

Frozen RAG

No training, only «In-Context Learning»



Frozen RAG

No training,

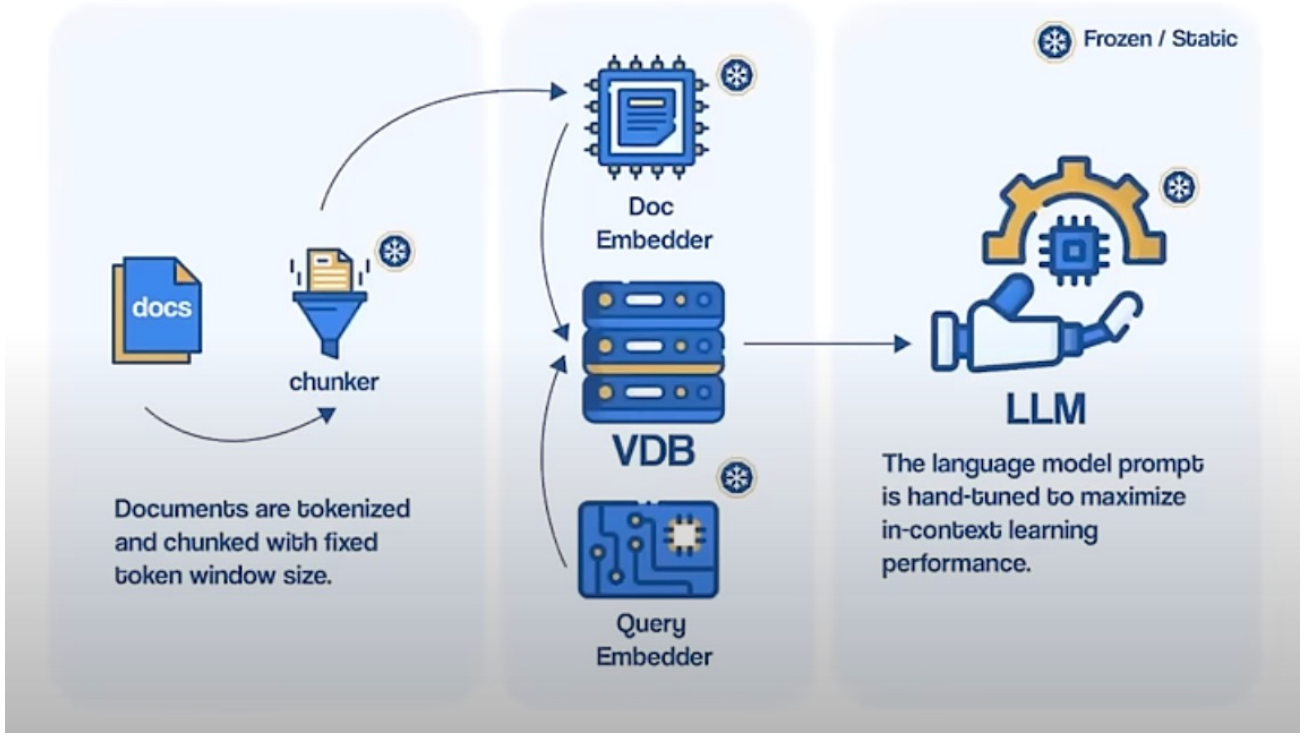
only «In-Context Learning»

Pros:

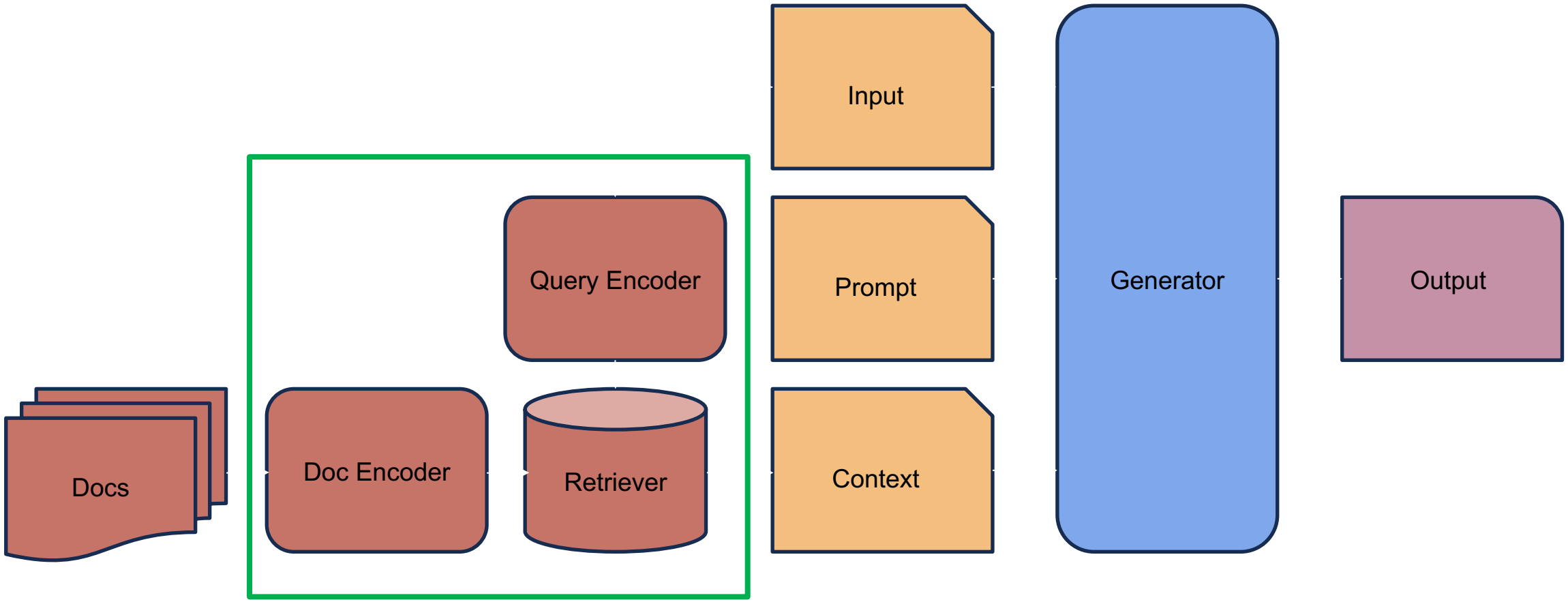
- Reasonable baseline.
- Can use black-box models for the retriever and the generator.

Cons:

- Not flexible enough.
- The resulting may not be sufficient.



Train retriever for generator



Train retriever for generator

RePlug (Shi et al 2023): train retriever, not the generator.

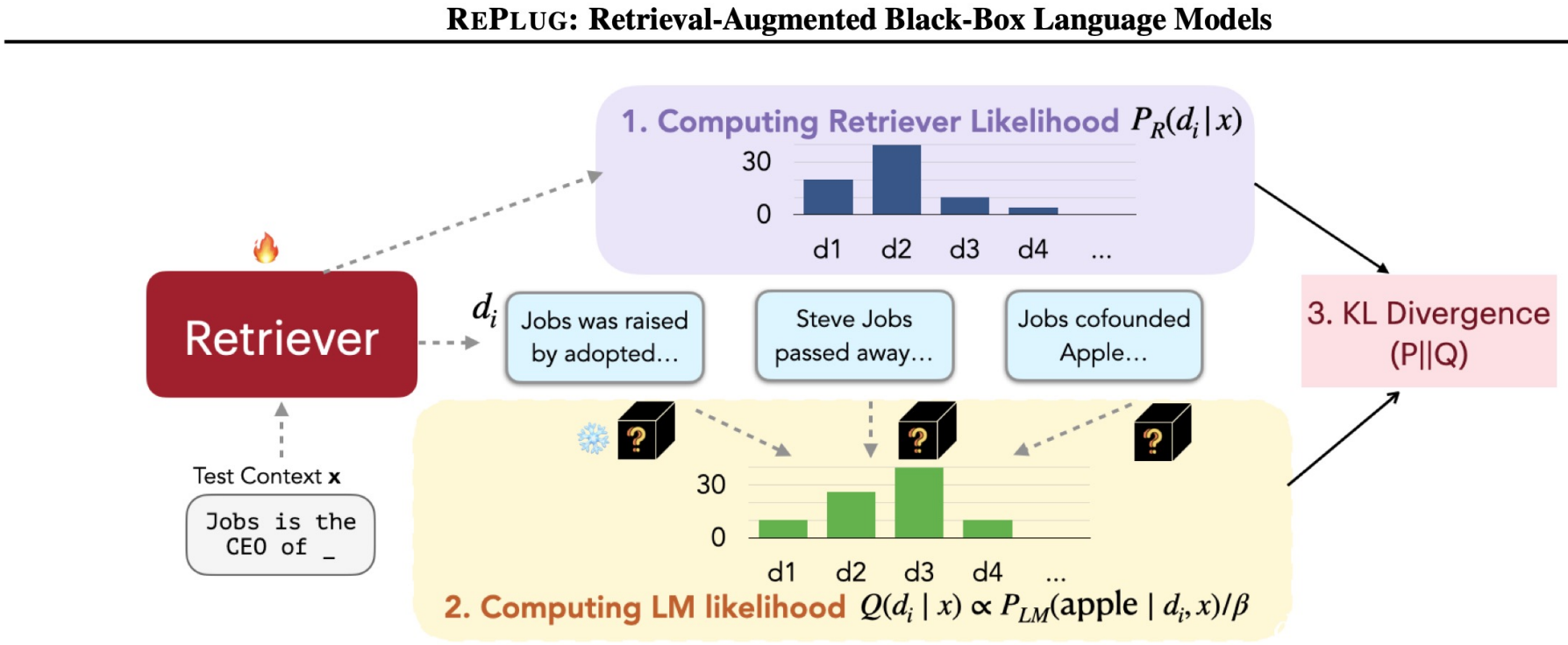


Figure 3. **REPLUG LSR training process (§4).** The retriever is trained using the output of a frozen language model as supervision signals.

Train retriever for generator

Pros:

- Retriever is fine-tune for the particular generator.
- May use black-box LLMs, such as ChatGPT, as a generator.

Cons:

- Good, but we want better!

Train retriever, not the generator.

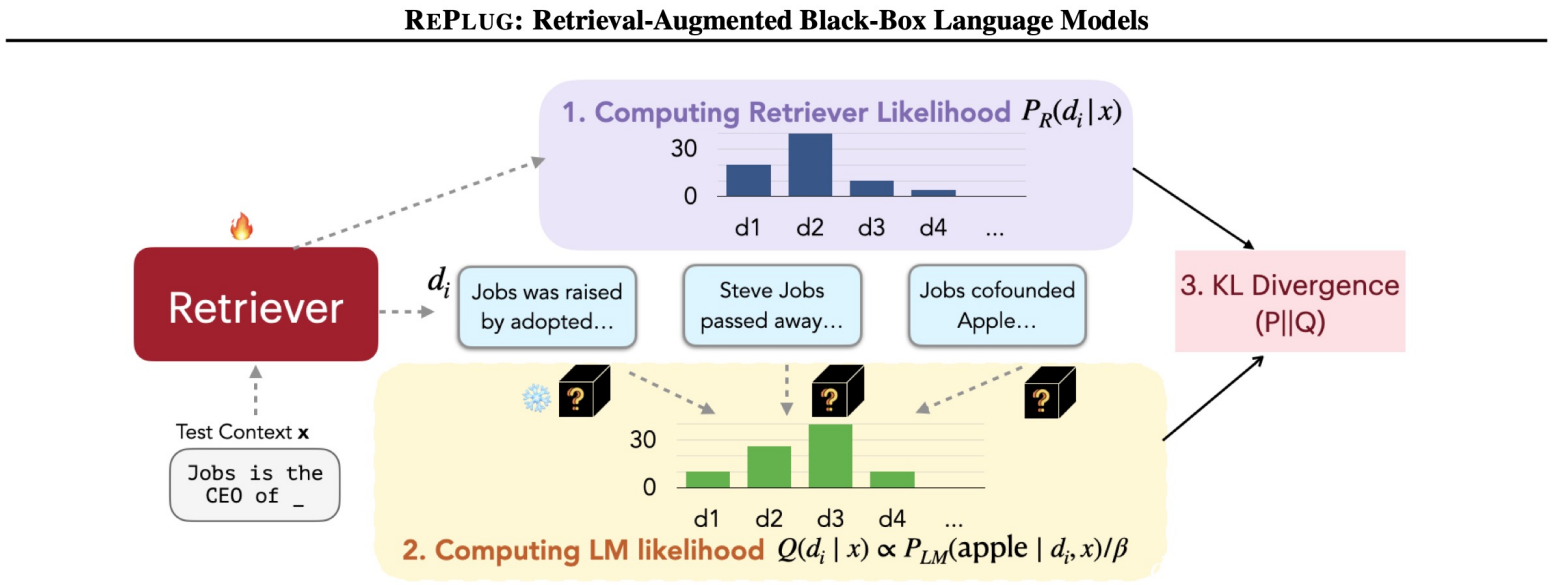
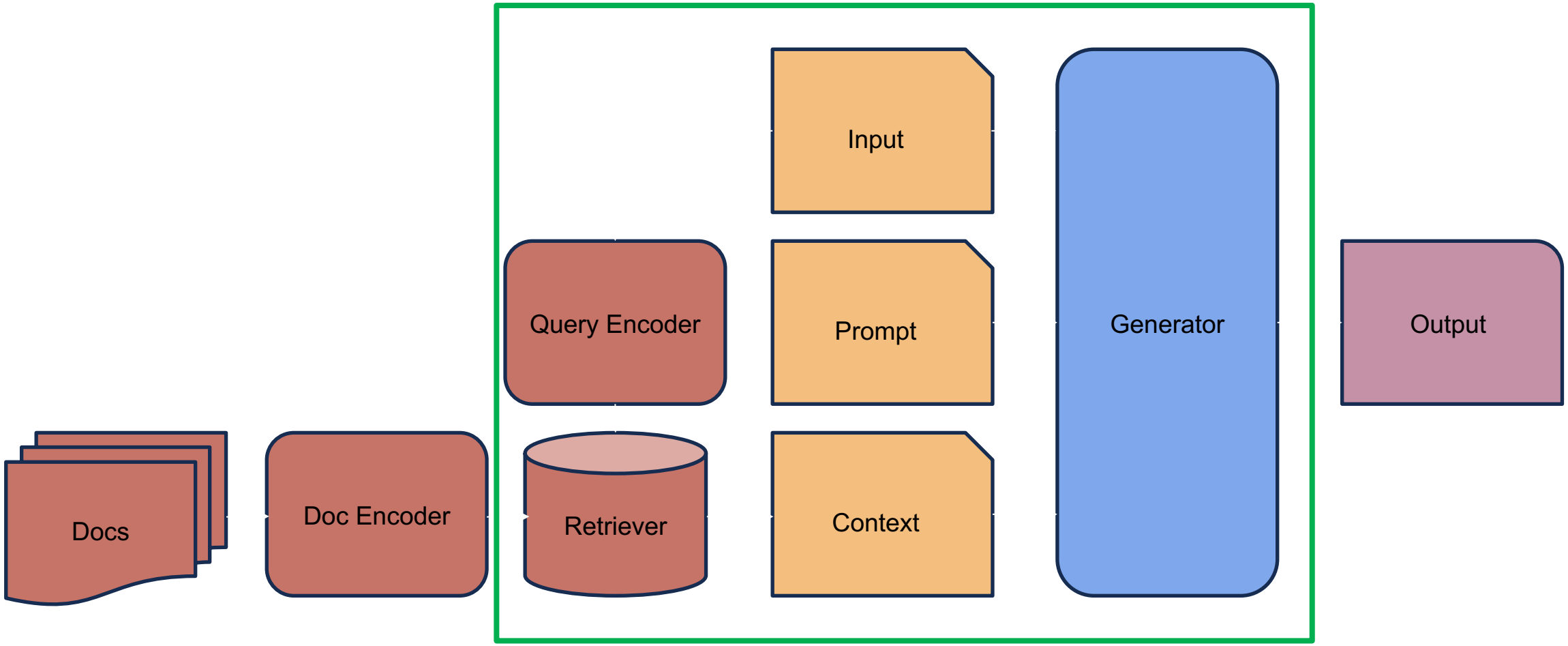


Figure 3. **REPLUG LSR training process (§4).** The retriever is trained using the output of a frozen language model as supervision signals.

Train both


Input
Query Encoder
Prompt
Generator
Output
Docs
Doc Encoder
Retriever
Context

Original RAG approach

Combines a pre-trained retriever (Query Encoder + Document Index) with a pre-trained seq2seq model (Generator) fine-tuned end-to-end.

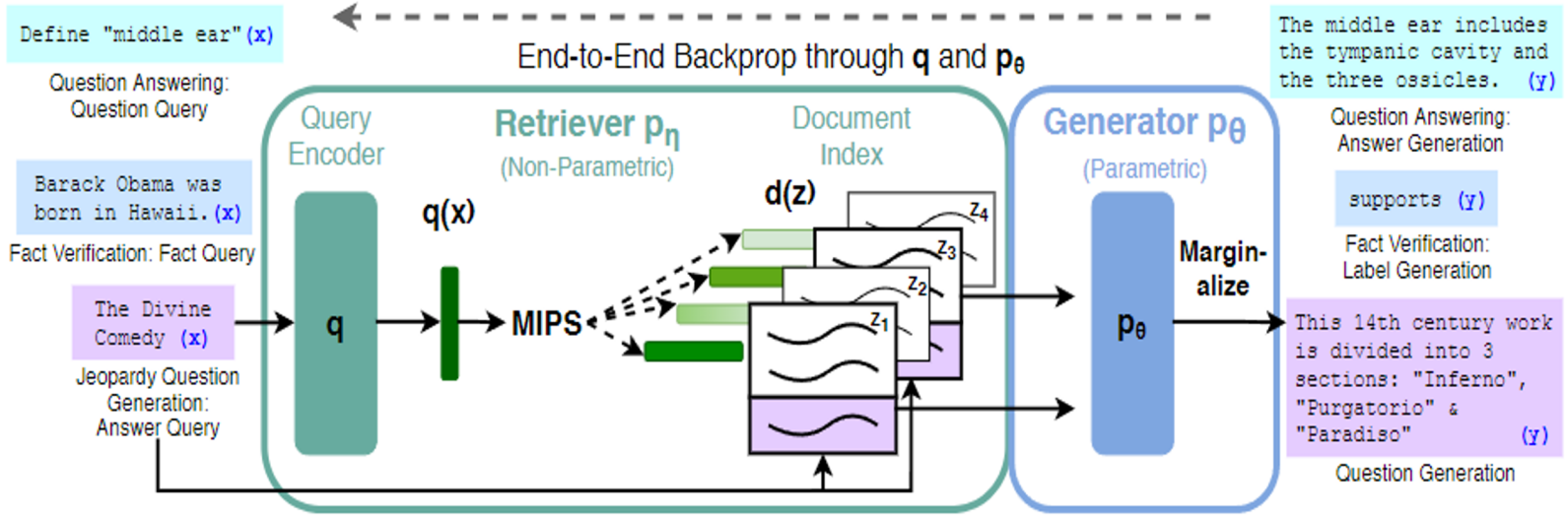


Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder* + *Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x, we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i. For final prediction y, we treat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

Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks - NeurIPS **2021**

https://arxiv.org/pdf/2005.11401

Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

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

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

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

RAG details

a retriever $p_\eta(z|x)$

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

Two models:

- **RAG-Sequence** 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** can draw a different latent document for each target token and marginalize accordingly.

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

https://arxiv.org/pdf/2005.11401

RAG details

a retriever $p_\eta(z|x)$

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

Two models:

- **RAG-Sequence** 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** can draw a different latent document for each target token and marginalize accordingly.

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

- Retriever is based on **DPR**.

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

- Generator is **a BART- large** model.

- **Training**: the retriever and generator components are **jointly trained** without any direct supervision on what document should be retrieved.

https://arxiv.org/pdf/2005.11401

Experiments & results

Takeaways:

- On all four open-domain QA tasks, RAG sets a new state of the art.
- RAG combines the generation flexibility of the “closed-book” (parametric only) approaches and the performance of "open-book" retrieval-based approaches.
-
- RAG can generate correct answers even when the correct answer is not in any retrieved document.

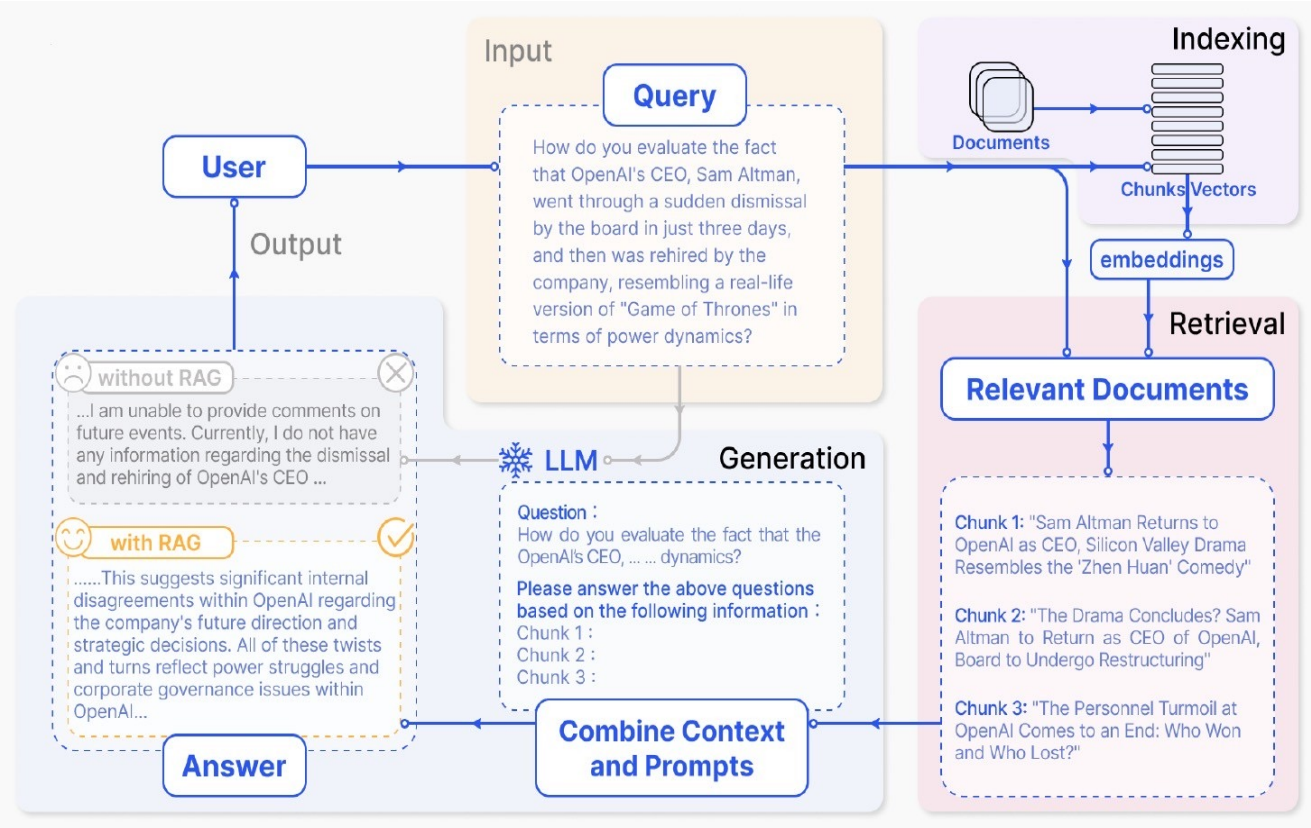
Table 1: Open-Domain QA Test Scores. For TQA, left column uses the standard test set for Open-Domain QA, right column uses the TQA-Wiki test set. See Appendix D for further details.

	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**

https://arxiv.org/pdf/2005.11401

Advanced Frozen RAG

- Active community
- Frameworks: **LangChain**, LlamaIndex, GigaChain
- Vector Databases: Chroma,
- Weaviate, etc
- Cool examples:
  - *Child-Parent Retriever*
  - *Hybrid Search*
  - *LLM reranker*
  - …

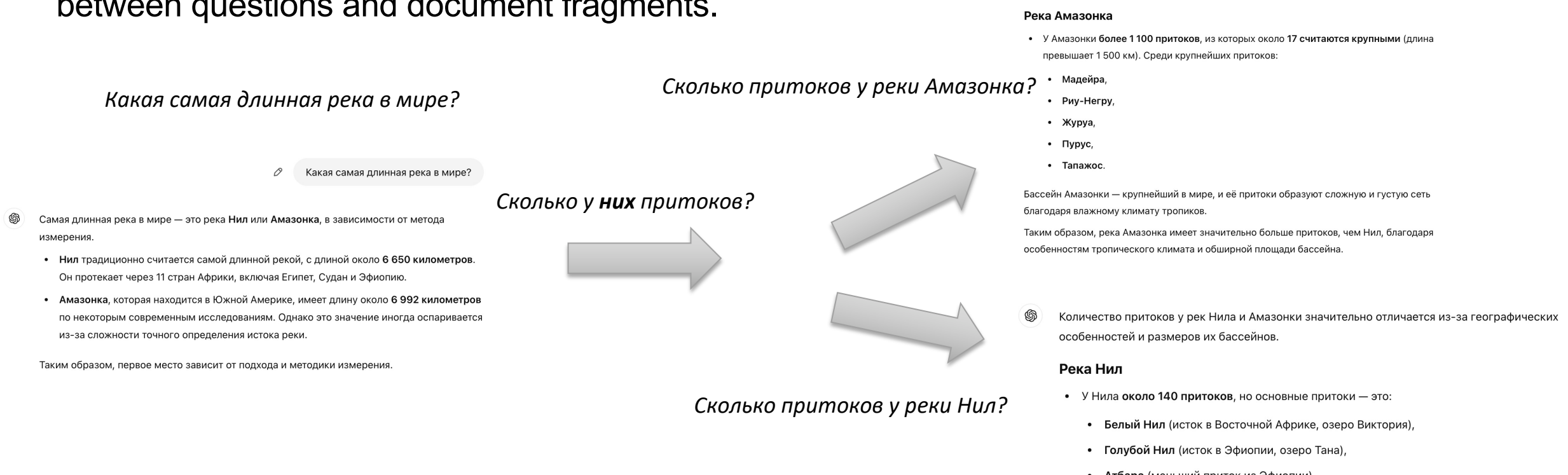


Advanced RAG: extra

The advanced RAG paradigm includes additional processing before retrieval and after retrieval.

Before retrieval:

- Apply a classifier which tells, whether the user's input queries factual information or not.
- Reformulate the question to include context information and equalize semantic differences between questions and document fragments.

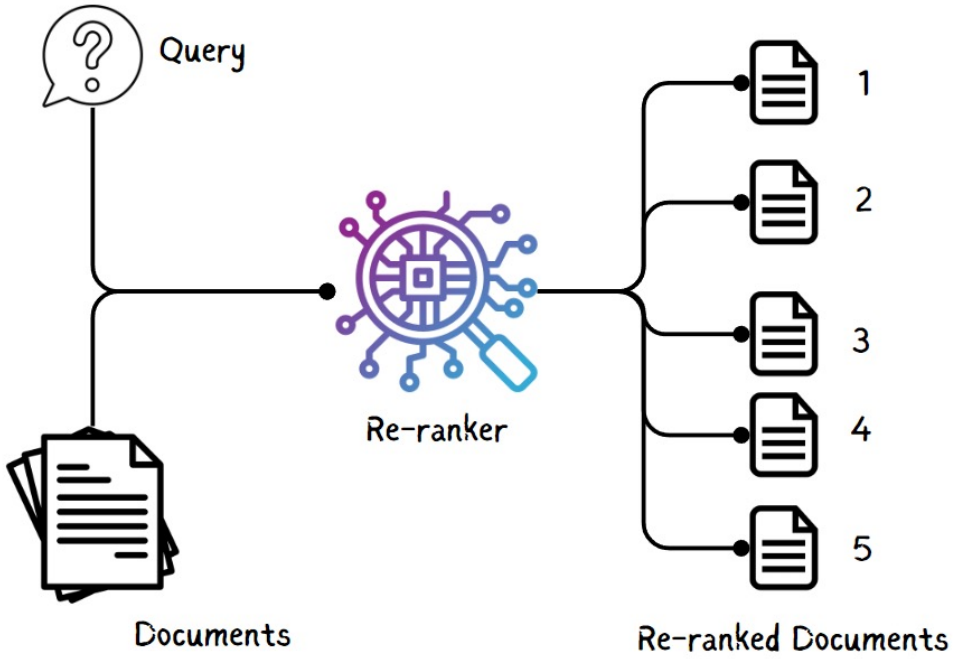


Advanced RAG: extra

The advanced RAG paradigm includes additional processing before retrieval and after retrieval.

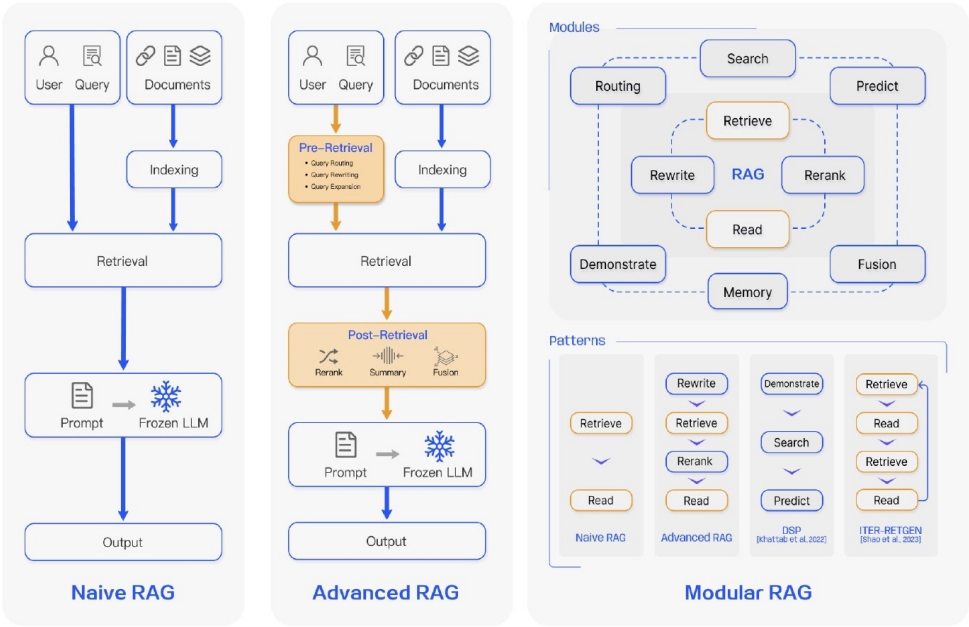
After retrieval:

- Rank the corpus of the retrieved documents.
- Filter the retrieved context and compress it to reduce the length of the context window.



Modular RAG

- Structurally, it is more flexible, with more specific functional modules such as **query search engines** and **multiple response pooling.**
- Technologically, it combines **search with fine-tuning**, r**einforcement learning** and other methods.



RAG Evaluation

Evaluation of RAG systems

Critical features of good RAG evaluation:

- **Realism:** quality correlation with real scenarios.
- **Richness:** diverse set of instances. (common and complex, use cases...)
- **Insightfulness:** (interpretability) understanding of performance on different slices of the data.
- **Reliability:** accurate GT, the metrics capture the model performance well, statistical significance.
- **Longevity:** long term data, refreshed data.

Table 3: Summary of evaluation frameworks

Evaluation Framework	Evaluation Targets	Evaluation Aspects	Quantitative Metrics
RGB†	Retrieval Quality Generation Quality	Noise Robustness Negative Rejection Information Integration Counterfactual Robustness	Accuracy EM Accuracy Accuracy
RECALL†	Generation Quality	Counterfactual Robustness	R-Rate (Reappearance Rate)
RAGAS‡	Retrieval Quality Generation Quality	Context Relevance Faithfulness Answer Relevance	* * Cosine Similarity
ARES‡	Retrieval Quality Generation Quality	Context Relevance Faithfulness Answer Relevance	Accuracy Accuracy Accuracy
TruLens‡	Retrieval Quality Generation Quality	Context Relevance Faithfulness Answer Relevance	* * *

Retrieval metrics

Reranking metrics: retriever can be evaluated with common ranking metrics (*NDCG, MRR, MAP, etc.*).

$$\text{Context Precision@K} = \frac{\sum_{k=1}^{K} (\text{Precision@k} \times v_k)}{\text{Total number of relevant items in the top } K \text{ results}}$$

$$\text{Precision@k} = \frac{\text{true positives@k}}{(\text{true positives@k} + \text{false positives@k})}$$

$$\text{Context Recall} = \frac{|\text{GT sentences that can be attributed to context}|}{|\text{Number of sentences in GT}|}$$

Entity-based: based on fine-grained context analysis. These metrics evaluate overlapping between entities in context (named entities or sentences).

$$\text{context relevancy} = \frac{|\text{Relevant sentences in retrieved context}|}{|\text{Total number of sentences in retrieved context}|}$$

$$\text{context entity recall} = \frac{|CE \cap GE|}{|GE|}$$

Generation metrics

Direct comparison with Ground Truth answers by EM or similarity.

Any distance in embedding space or string editing distances could be used as similarity.

Models	Accuracy	
	Retrieved Chunk	Ground-truth Chunk
GPT-4	**0.56**	**0.89**
ChatGPT	0.44	0.57
Llama-2-70b-chat-hf	0.28	0.32
Mixtral-8x7B-Instruct	0.32	0.36
Claude-2.1	0.52	0.56
Google-PaLM	0.47	0.74

Table 6: Generation accuracy of LLMs.

Generation metrics

LLM as a Judge:

- Direct scoring: Perfect, Acceptable, Missing, Incorrect
- Comparison with Ground Truth answer
- Evaluate aspect-specific quality:
- Harmlessness – Correctness - Maliciousness - Coherence
- Conciseness

Domain-specific metrics

Some domains allows to build specific metrics for answer evaluation.

For example:

- *In code domain we can use not only EM on the whole prediction but also the Identity Matching.*
- *Also we can run Unit Tests on the predicted snippet and obtain the real quality of the answer.*

ID	N.	Oracle	In-File	RepoCoder Iterations			
				1	2	3	4
1.	67	56.72	29.85	53.73	**55.22**	55.22	55.22
2.	146	43.84	27.40	41.78	43.84	**44.52**	44.52
3.	64	32.81	10.94	25.00	**34.38**	31.25	32.81
4.	32	34.38	28.13	34.38	**37.50**	34.38	34.38
5.	22	40.91	31.82	31.82	**36.36**	31.82	36.36
6.	42	38.10	9.52	28.57	**38.10**	38.10	38.10
All	373	42.63	23.32	38.34	**42.63**	41.82	42.36

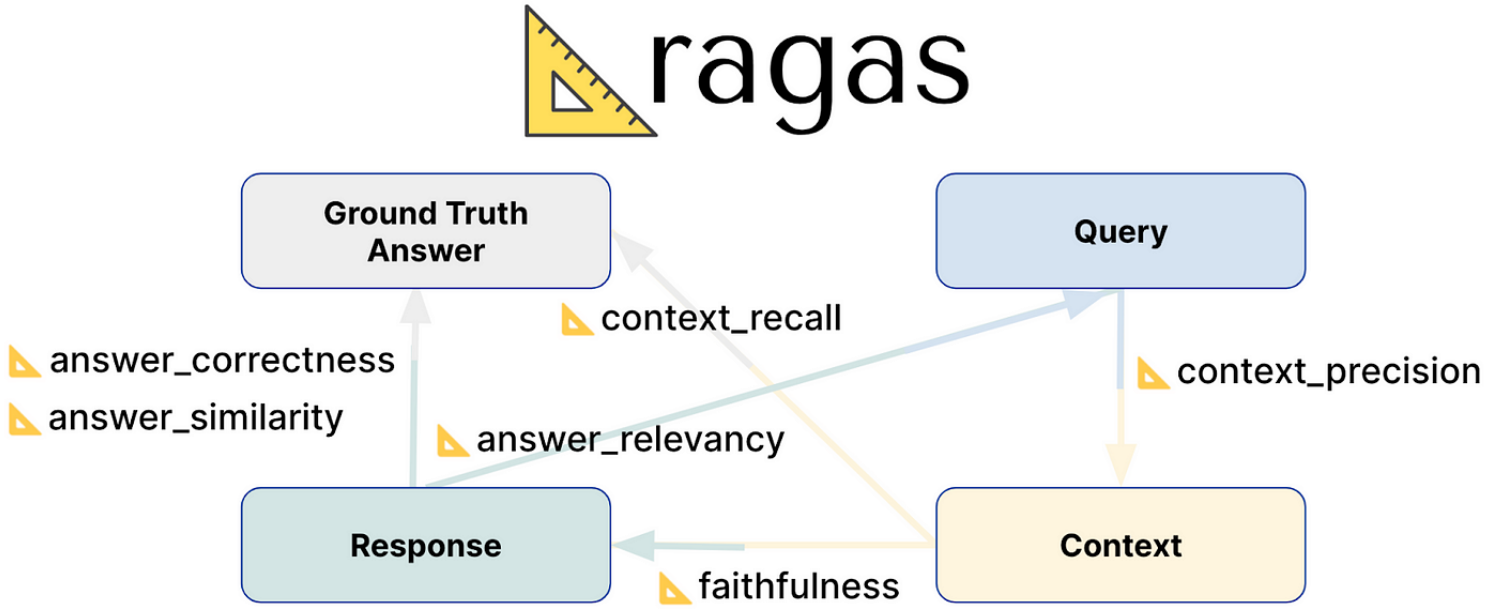
Table 3: Performance comparison on the function body completion dataset using GPT-3.5-Turbo. Results display the Pass Rate (PR) of each method as evaluated using test cases. Numbers are presented in percentage (%), with the best performance highlighted in bold. *ID* represents the repository IDs, and *N.* indicates the number of test samples in each repository.

https://medium.com/@danushidk507/evaluation-with-ragas-873a574b86a9

Ragas

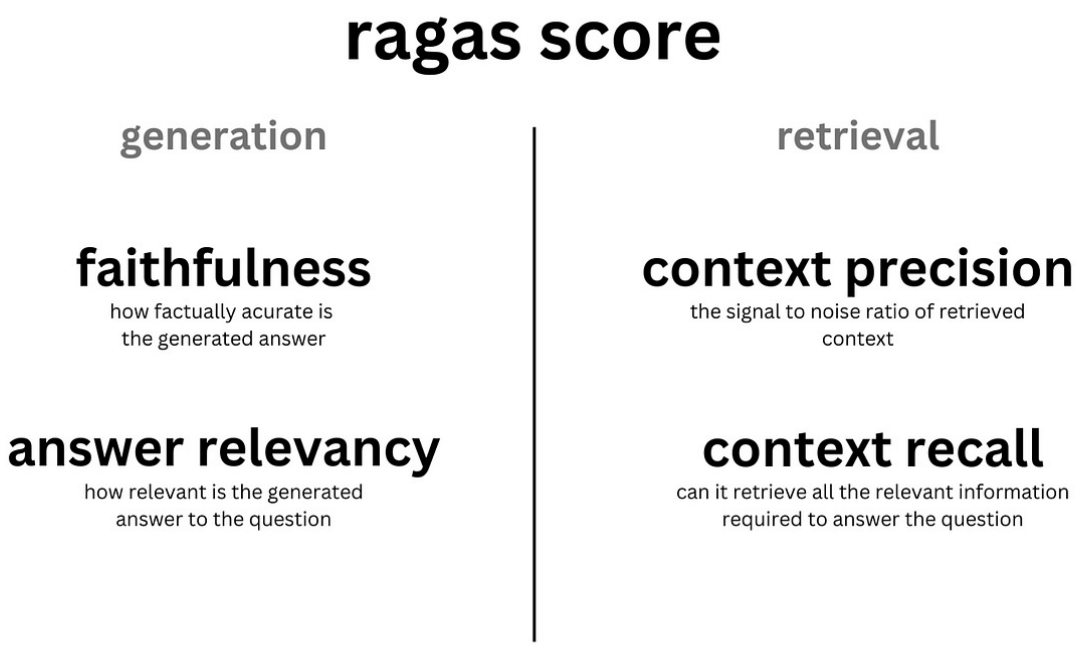
Ragas is a **framework designed to evaluate RAG systems**, which combine language models LLMs with external knowledge retrieval mechanisms.

Provides a structured approach to evaluate **both the generator and retriever components** of a RAG pipeline through a set of metrics that can be applied without requiring extensive labeled datasets.



https://medium.com/@danushidk507/evaluation-with-ragas-873a574b86a9

Key metrics of Ragas



https://medium.com/@danushidk507/evaluation-with-ragas-873a574b86a9

Key metrics of Ragas

Faithfulness: *the factual accuracy of the generated answer*s by checking if the statements made in the answers are supported by the provided context. It is computed by analyzing the validity of each statement derived from the generated answer against the context.

Answer Relevancy: Measures *how relevant* and directly related the generated answer is to the posed question. It evaluates the similarity between probable questions that could elicit the same answer and the actual question asked.

The Answer Relevancy is defined as the mean cosine similarity of the original `user_input` to a number of artifical questions, which where generated (reverse engineered) based on the `response`:

$$\text{answer relevancy} = \frac{1}{N}\sum_{i=1}^{N} cos(E_{g_i}, E_o)$$

$$\text{answer relevancy} = \frac{1}{N}\sum_{i=1}^{N} \frac{E_{g_i} \cdot E_o}{\|E_{g_i}\|\|E_o\|}$$

https://medium.com/@danushidk507/evaluation-with-ragas-873a574b86a9

Key metrics of Ragas

Context Precision: *Evaluates the signal-to-noise ratio within retrieved contexts*, determining how many sentences from the context are necessary to answer the question compared to the total number of sentences retrieved.

Context Recall: Checks whether all necessary information needed to support the generated answer can be found in the retrieved context. It compares statements from a ground truth answer against what has been retrieved.

Context Precision is a metric that measures the proportion of relevant chunks in the `retrieved_contexts`. It is calculated as the mean of the precision@k for each chunk in the context. Precision@k is the ratio of the number of relevant chunks at rank k to the total number of chunks at rank k.

$$\text{Context Precision@K} = \frac{\sum_{k=1}^{K} (\text{Precision@k} \times v_k)}{\text{Total number of relevant items in the top } K \text{ results}}$$

$$\text{Precision@k} = \frac{\text{true positives@k}}{(\text{true positives@k} + \text{false positives@k})}$$

Where K is the total number of chunks in `retrieved_contexts` and vk∈{0,1} is the relevance indicator at rank k.

Computed using `user_input`, `reference` and the `retrieved_contexts`, and the values range between 0 and 1, with higher values indicating better performance. This metric uses `reference` as a proxy to `reference_contexts` which also makes it easier to use as annotating reference contexts can be very time consuming. To estimate context recall from the `reference`, the reference is broken down into claims each claim in the `reference` answer is analyzed to determine whether it can be attributed to the retrieved context or not. In an ideal scenario, all claims in the reference answer should be attributable to the retrieved context.

$$\text{context recall} = \frac{|\text{GT claims that can be attributed to context}|}{|\text{Number of claims in GT}|}$$

https://medium.com/@danushidk507/evaluation-with-ragas-873a574b86a9

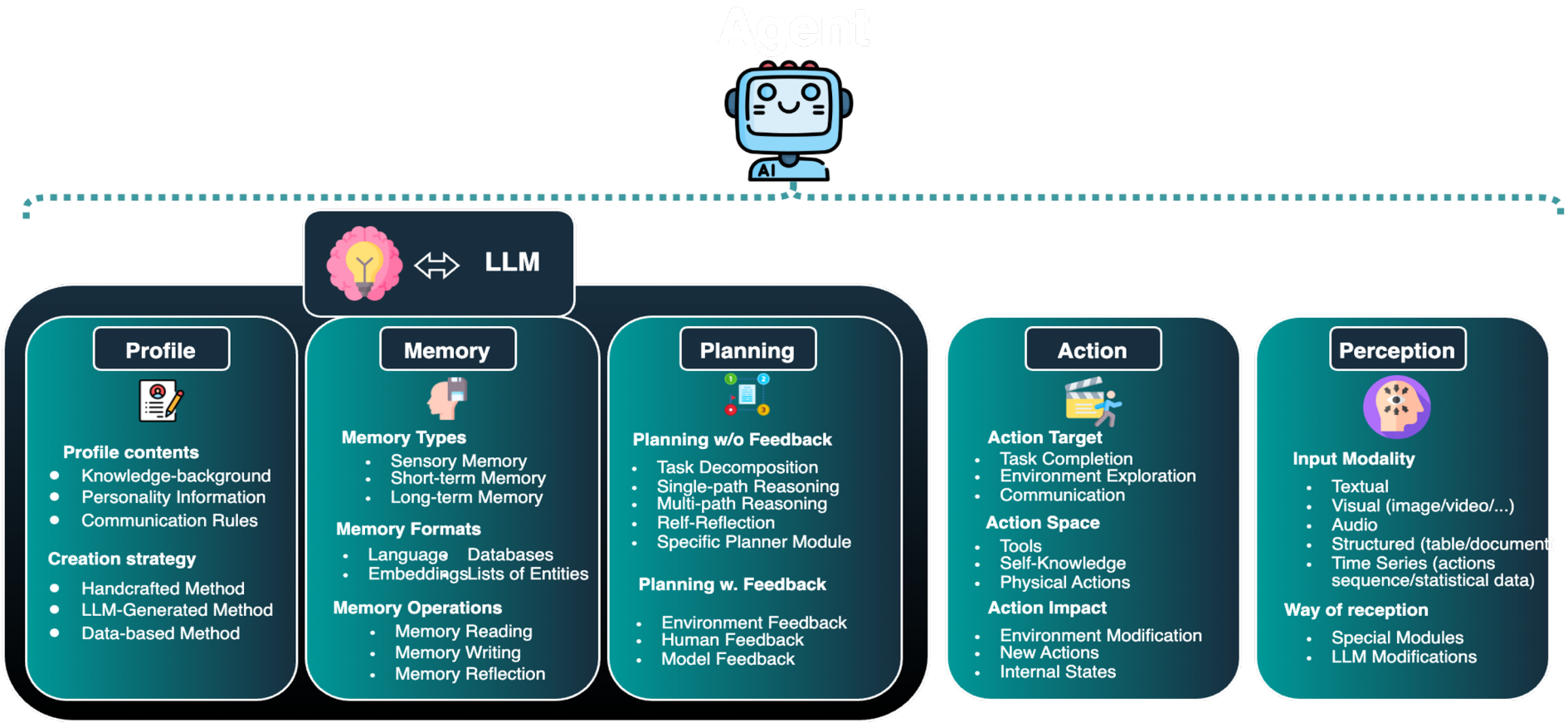
RAG conclusion

- There is no one “silver bullet”: **there is a wide variety of approaches and ideas**.
- For each use case, **you need a customized solution** based on cost/benefit considerations.
- Not only are system architectures important, but so **is the quality of the data** in the index.
- Special attention to **the evaluation of RAG systems**.

Intro to AI Agents

Introduction to AI Agents

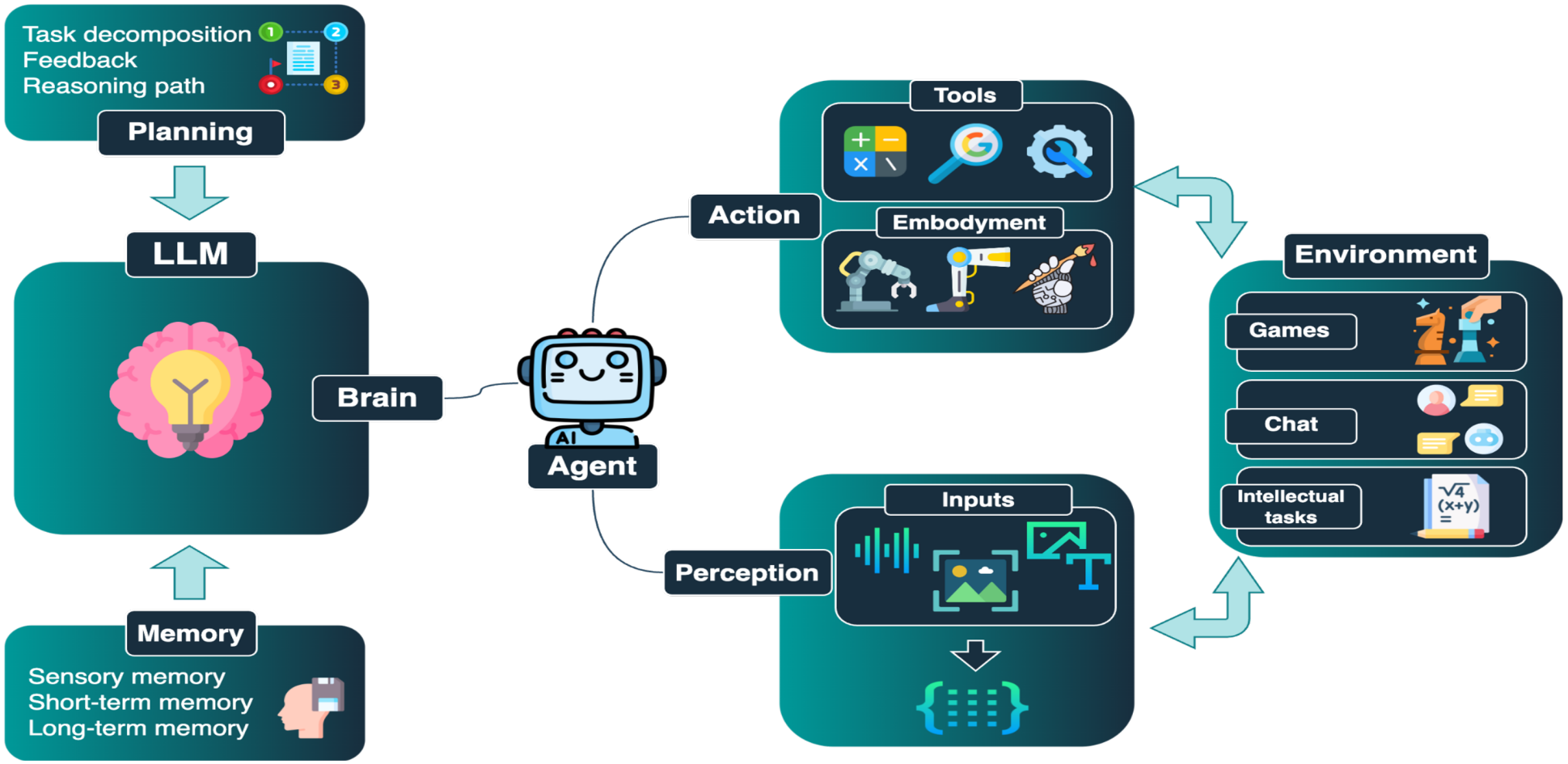
AI Agents are autonomous entities capable of perceiving their environment, making decisions, and executing actions to achieve specific goals.



Introduction to AI Agents

- **AI Agents** are autonomous entities capable of perceiving their environment, making decisions, and executing actions to achieve specific goals.
- AI agents have been developed to **operate in controlled environments**, such as games or simulated systems.
- Large Language Models (LLMs) has expanded the potential of AI agents, enabling them to handle **more complex** and **dynamic real-world tasks**.
- Evaluating the effectiveness of these AI agents, especially those based on LLMs, requires **robust benchmarking tools.**

LLM as an agent



On the way to multiagent

LLMs are instruction-following systems prone to errors, hallucinations, and challenges with multi-step reasoning in their responses.

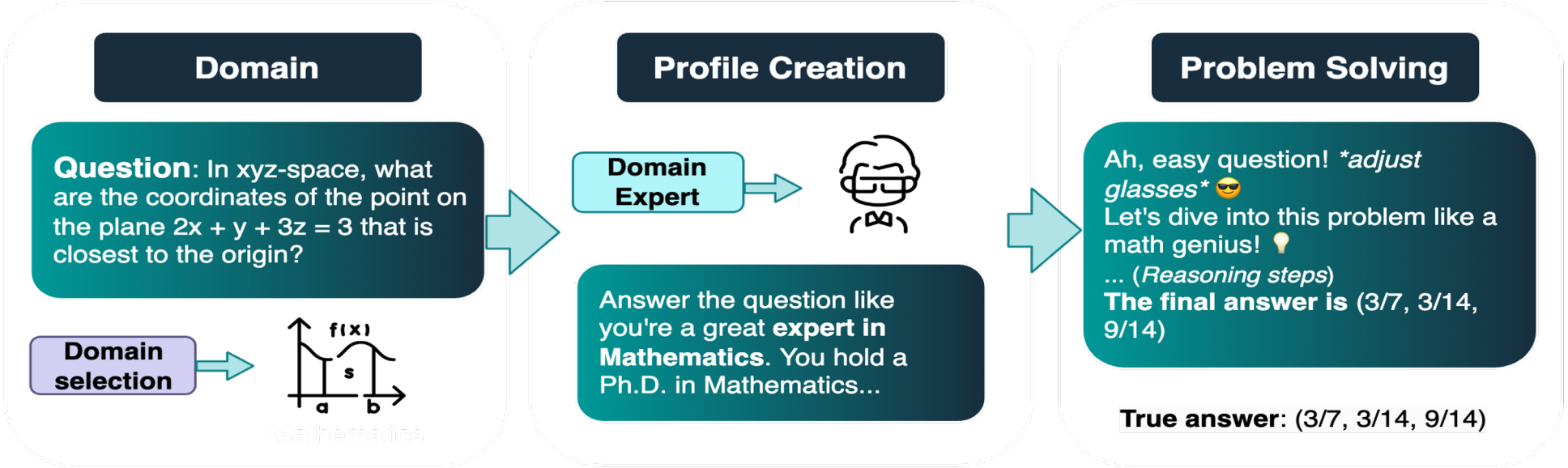
Integrating external tools + RAG reduces the risk of hallucinations and enables the handling of more complex tasks, such as code execution.

It enables solving tasks in multiple directions (**roles**) and establishing specialized forms of communication between agents for information exchange (shared context).

Agent roles

Dialogue roles for LLMs allow them to unlock knowledge within a specific domain that previously remained in the background — their "experience baggage." This capability enables the model to:

- create more effective chains of reasoning,
- solve tasks with greater accuracy.

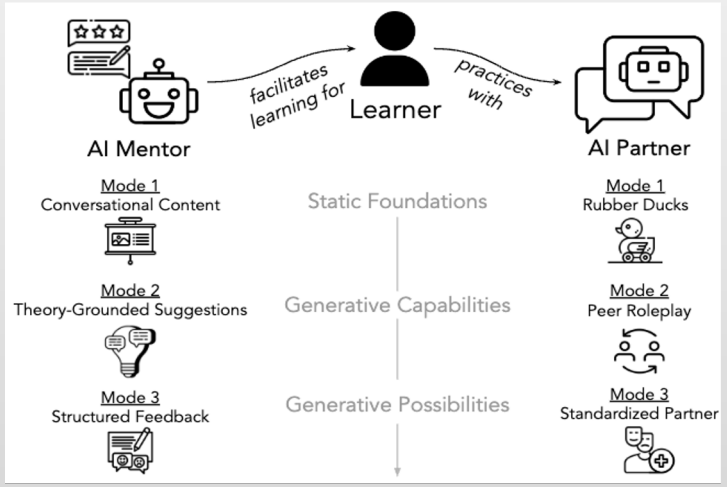


Agents

- An agent can be understood as either an independent SFT LLM or an LLM with a specific prompt-role.
- An agent can communicate, possess reasoning abilities, execute actions, and solve problems.
- It can manage complex and multi-layered information from the context (shared task context, agents, and a certain knowledge base).
- Automation of the communication process between various "specialist" models (e.g., financial analyst, lawyer, economist, etc.).
- Process chains can be fully or partially replaced by agents, depending on the tasks.

Task

SoftSkill development



Task

Based on the lease document, generate a contract in the form of a JSON structure.

AI Agents Evaluation

Agent evaluation challenges

Agent evaluation differs from LLM in fundamental ways.

Tasks for them are:

- *harder*
- *more realistic*
- *have more real-world utility*
- *not always a single correct answer*

Evaluation challenges:

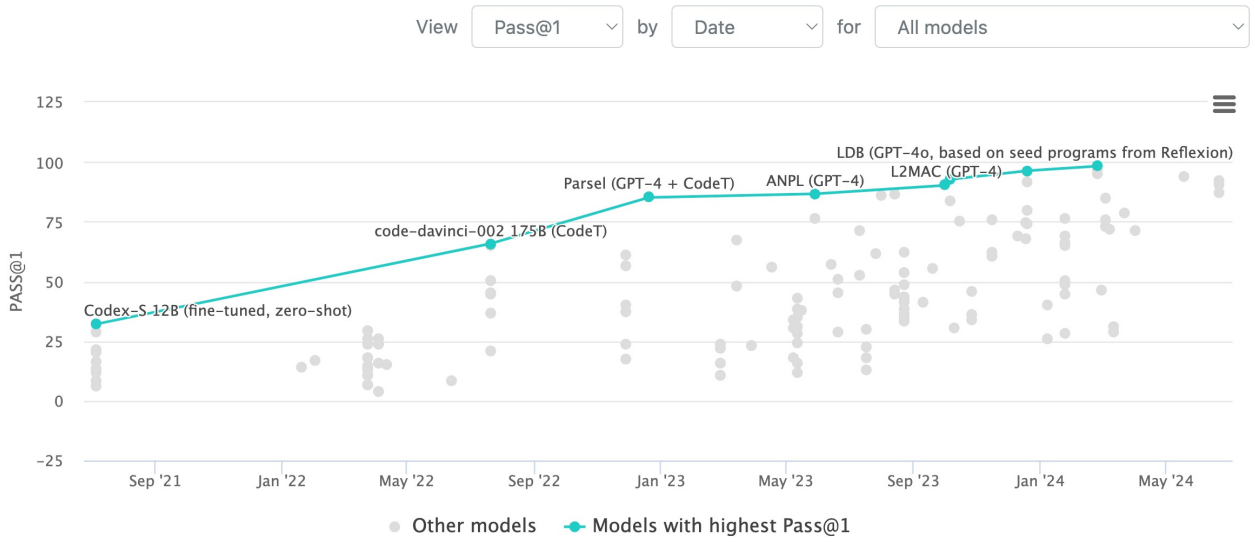
1. **AI agent evaluations must be cost-controlled.** The language models underlying most AI agents are stochastic. This means simply calling the model multiple times can increase accuracy and cost.
2. **Jointly optimizing accuracy and cost can yield better agent design.**
3. **Model developers and downstream developers have distinct benchmarking needs.** Benchmarks meant for model evaluation can be misleading for downstream evaluation. Downstream evaluation should account for dollar costs.
4. **Agent benchmarks enable shortcuts.** Without proper hold-out sets/tasks/domain, models can overfit, even unintentionally.
5. **Agent evaluations lack standardization and reproducibility**.

https://arxiv.org/pdf/2407.01502v1

Human eval is a code classics

Used to measure functional correctness for synthesizing programs from docstrings.

Consists of 164 original programming problems, assessing language comprehension, algorithms, and simple mathematics.




```python
def incr_list(l: list):
    """Return list with elements incremented by 1.
    >>> incr_list([1, 2, 3])
    [2, 3, 4]
    >>> incr_list([5, 3, 5, 2, 3, 3, 9, 0, 123])
    [6, 4, 6, 3, 4, 4, 10, 1, 124]
    """
    return [i + 1 for i in l]
```

```python
def solution(lst):
    """Given a non-empty list of integers, return the sum of all of the odd elements
    that are in even positions.

    Examples
    solution([5, 8, 7, 1]) ==>12
    solution([3, 3, 3, 3, 3]) ==>9
    solution([30, 13, 24, 321]) ==>0
    """
    return sum(lst[i] for i in range(0,len(lst)) if i % 2 == 0 and lst[i] % 2 == 1)
```

```python
def encode_cyclic(s: str):
    """
    returns encoded string by cycling groups of three characters.
    """
    # split string to groups. Each of length 3.
    groups = [s[(3 * i):min((3 * i + 3), len(s))] for i in range((len(s) + 2) // 3)]
    # cycle elements in each group. Unless group has fewer elements than 3.
    groups = [(group[1:] + group[0]) if len(group) == 3 else group for group in groups]
    return "".join(groups)

def decode_cyclic(s: str):
    """
    takes as input string encoded with encode_cyclic function. Returns decoded string.
    """
    # split string to groups. Each of length 3.
    groups = [s[(3 * i):min((3 * i + 3), len(s))] for i in range((len(s) + 2) // 3)]
    # cycle elements in each group.
    groups = [(group[-1] + group[:-1]) if len(group) == 3 else group for group in groups]
    return "".join(groups)
```

https://paperswithcode.com/sota/code-generation-on-humaneval

ruHumanEval

Russian **HumanEval (ruHumanEval)** is the Russian analogue of the original HumanEval dataset from MERA benchmark, created to evaluate the ability of language models to generate code in the Python programming language to solve simple problems.

ruHumanEval

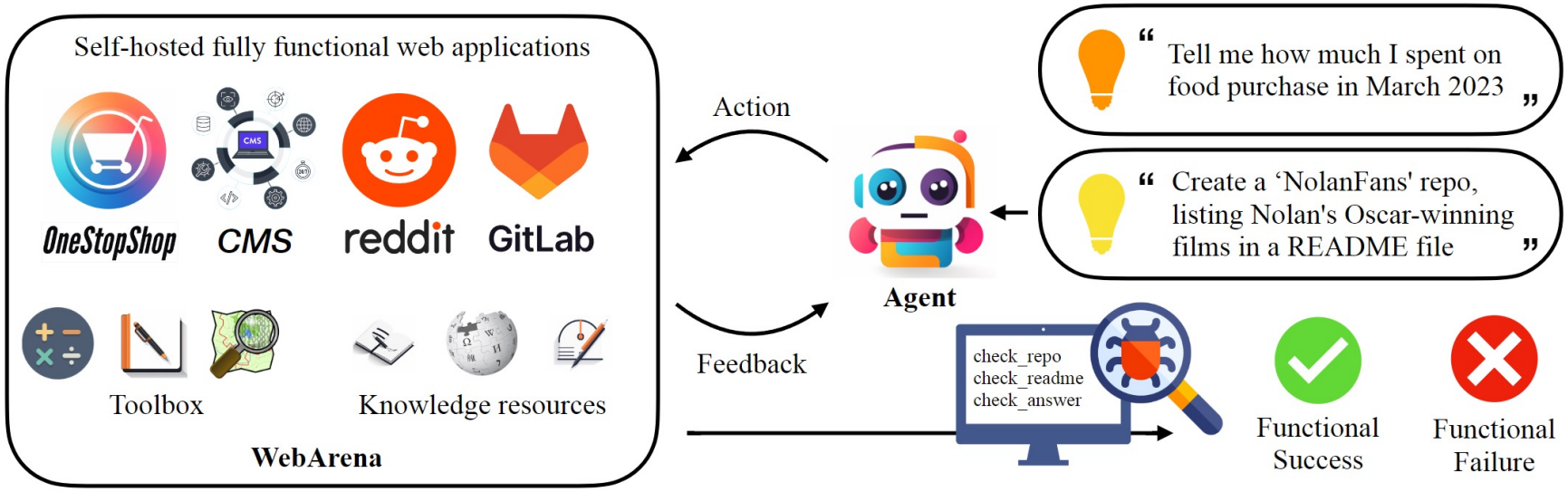
Assessing models' abilities to generate solutions for straight-forward programming problems on Python. The solution is considered to be correct if it passes all test case.

```
  "instruction": "На вход подается функция с описанием в
виде строки docstring. В соответствии с описанием вам
необходимо реализовать функцию на основе
шаблона:\\n{function}"
  "inputs": {
     "function":
          "def greatest_common_divisor(a: int, b: int) ->
int:
         '''Верните наибольший общий делитель двух целых
чисел a и b.
         Примеры: greatest_common_divisor(3, 5)
         1 ''' ",
         "tests": "[{'a': 3, 'b': 7}, {'a': 10, 'b': 15},
{'a': 49, 'b': 14}, {'a': 144, 'b': 60}]"
         },
  "outputs": ["1", "5", "7", "12"]
```

https://arxiv.org/abs/2401.04531v2

WebArena

- WebArena is web environment for building autonomous agents.
- WebArena creates websites from four popular categories with functionality and data **mimicking their real-world equivalents**.
- To emulate human problem-solving, WebArena also embeds tools and knowledge resources as independent websites.



https://webarena.dev/
https://arxiv.org/pdf/2307.13854

Realistic tasks on WebArena

A high-level task that can be fully executed in WebArena.

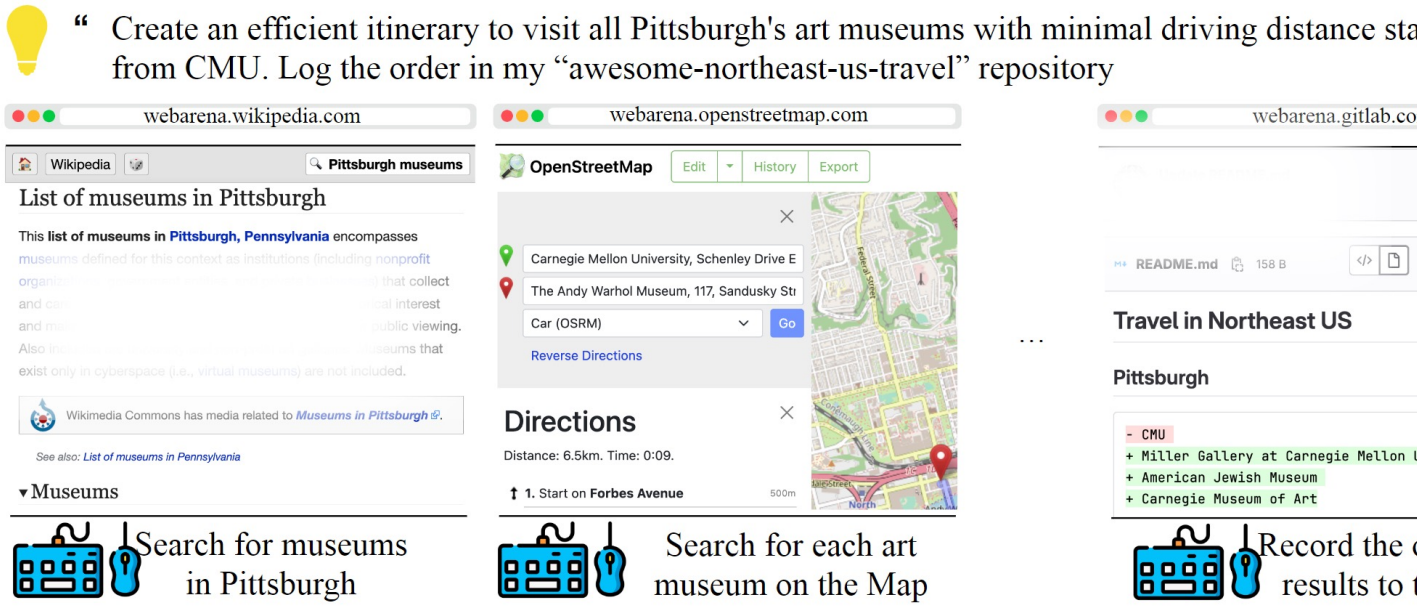
Completing such tasks requires sophisticated, long-term planning and reasoning capability.

Examples:

Subscribe to the newsletter on OneStopMarket

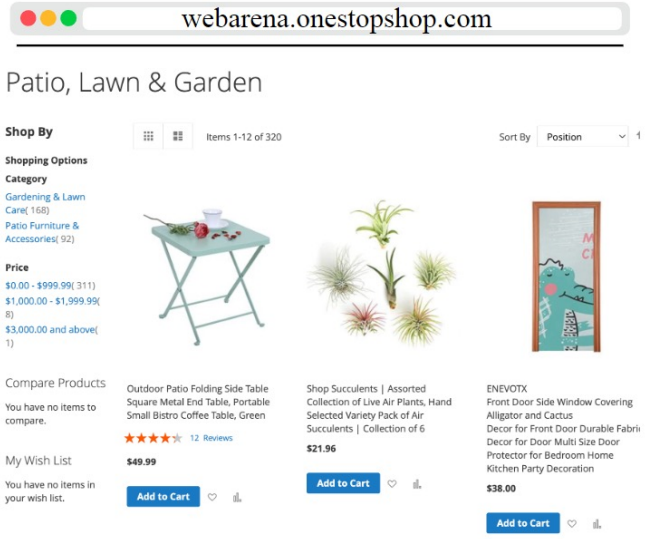
Cancel order 307

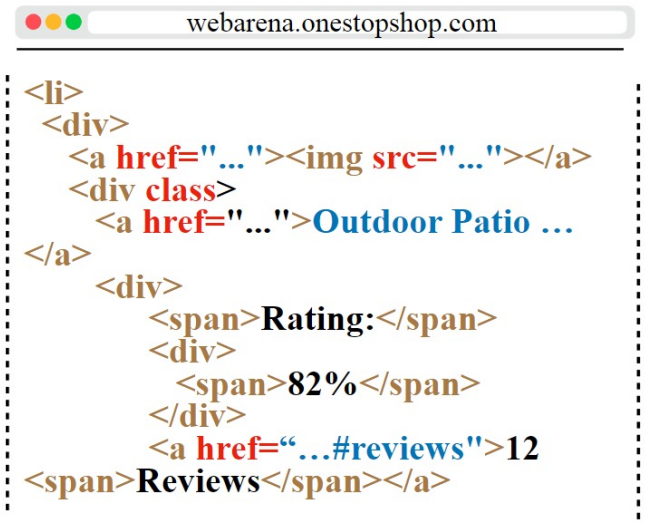
Set my Gitlab status as enjoying life



Observation space

The observation is the URL and the content of a web page, with options to represent the content as a screenshot, HTML DOM tree, and accessibility tree.





Evaluation

Two evaluation approaches:

- **The correctness of performing information seeking tasks**. It compares the predicted answer with the annotated reference with three implementations.
- If **the intermediate states during the executions possess the anticipated properties** specified by the intent.

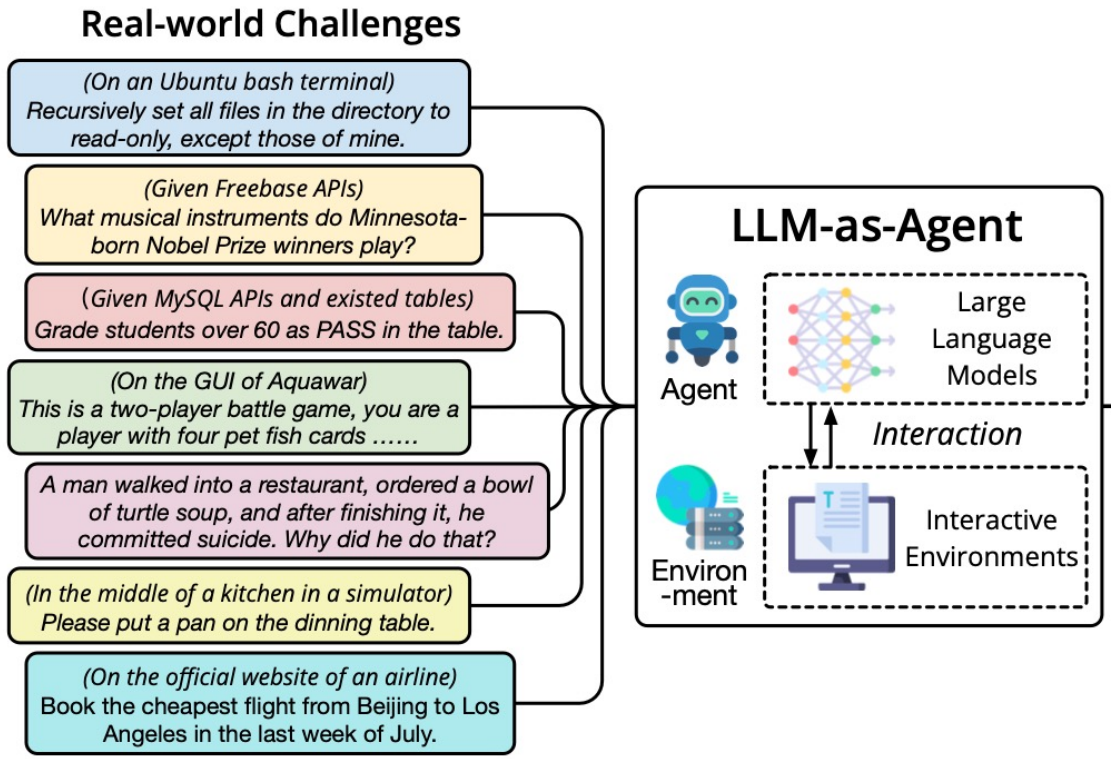
Function	ID	Intent	Eval Implementation
$r_{\text{info}}(a^*, \hat{a})$	1	Tell me the name of the customer who has the most cancellations in the history	`exact_match`($\hat{a}$, "Samantha Jones")
	2	Find the customer name and email with phone number 8015551212	`must_include`($\hat{a}$, "Sean Miller") `must_include`($\hat{a}$, "sean@gmail.com")
	3	Compare walking and driving time from AMC Waterfront to Randyland	`fuzzy_match`($\hat{a}$, "Walking: 2h58min") `fuzzy_match`($\hat{a}$, "Driving: 21min")
$r_{\text{prog}}(\mathbf{s})$	4	Checkout merge requests assigned to me	`url = locate_last_url(s)` `exact_match(url, ''gitlab.com/merge_requests?assignee_username'' =byteblaze'')`

	5	Post to ask "whether I need a car in NYC"	`url = locate_latest_post_url(s)`
			`body = locate_latest_post_body(s)`
			`must_include(url, ''/f/nyc'')`

https://webarena.dev/
https://arxiv.org/pdf/2307.13854

AgentBench

- **AgentBench** is a multi-dimensional benchmark for **LLMs as agent** evaluation in **interactive environments**.
- It assesses LLMs' **reasoning** and **decision-making abilities** across **eight distinct environments** of **three main types:** Code-Grounded, Game-Grounded, and Web-grounded.



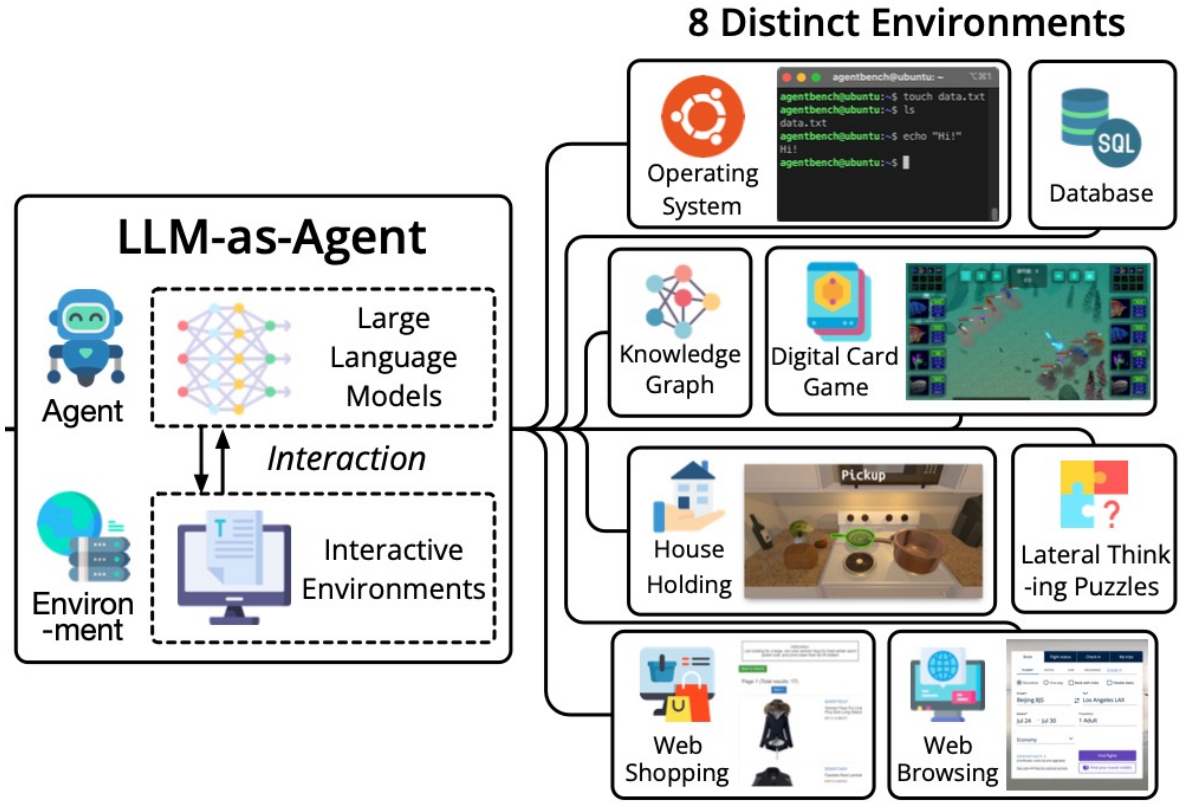
https://arxiv.org/abs/2308.03688

AgentBench Environments

Three Types of Groundings:

- **Code-Grounded Environments** coding and reasoning abilities.
- **Game-Grounded Environments** require no expertise in coding but more integral grasping of commonsense and world knowledge.
- **Web-Grounded Environments** interfaces for people to interact in the real world => **critical and valuable for following development**.

Each environment simulates real-world challenges where LLMs operate as autonomous agents.

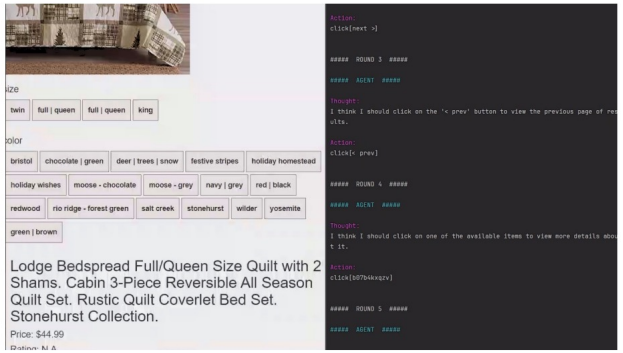


https://arxiv.org/abs/2308.03688

Evaluation methodology

- AgentBench uses a **Chain-of-Thought (CoT)** approach, a standard reasoning strategy for LLMs.
- The evaluation is structured as an **Interactive Evaluation.**
- LLMs are assessed on their ability to follow instructions, perform coding tasks, and engage in multi-turn interactions.
- **Metrics:** Success Rate (SR), F1 score, and Game Progress, among others.
- The **overall score** is calculated by resizing each task's average score to 1 and then averaging across all tasks for each model.

(d) **Digital Card Game (DCG)**
`Task:` "Compete against another player using four 'fish' cards in 'Aquawar' game."
`Action space:` Four 'fish' cards and Assertion
`Observation:` Battle process, status of 'fish'



(g) **Web Shopping (WS)**
`Task:` "Looking for a queen size bedspread set in the color redwood, and price lower than 70."
`Action space:` Search (generate keywords) and Click (choose from all clickable buttons)
`Observation:` Products' descriptions; the webpage

https://arxiv.org/abs/2308.03688

Experiments and results

The AgentBench framework was tested using 27 different LLMs, including both API-based commercial models and open-source alternatives.

Table 1: AGENTBENCH evaluates 27 API-based or OSS LLMs on LLM-as-Agent challenges.

Model	#Size	Form	Ver.	Creator	Model	#Size	Form	Ver.	Creator
gpt-4 (OpenAI, 2023)	N/A	api	0613	OpenAI	llama2-70b (Touvron et al., 2023)	70B	open	chat	Meta
gpt-3.5-turbo (OpenAI, 2022)	N/A	api	0613		llama2-13b (Touvron et al., 2023)	13B	open	chat	
text-davinci-003 (Ouyang et al., 2022)	N/A	api	-		llama2-7b (Touvron et al., 2023)	7B	open	chat	
text-davinci-002 (Ouyang et al., 2022)	N/A	api	-		guanaco-65b (Dettmers et al., 2023)	65B	open	-	Meta
claude-2 (Anthropic, 2023b)	N/A	api	-		guanaco-33b (Dettmers et al., 2023)	33B	open	-	
claude (Anthropic, 2023a)	N/A	api	v1.3	Anthropic	vicuna-33b (Chiang et al., 2023)	33B	open	v1.3	
claude-instant (Anthropic, 2023a)	N/A	api	v1.1		vicuna-13b (Chiang et al., 2023)	13B	open	v1.5	LMSYS
chat-bison-001 (Anil et al., 2023)	N/A	api	-	Google	vicuna-7b (Chiang et al., 2023)	7B	open	v1.5	
chatglm-6b (Zeng et al., 2022; Du et al., 2022)	6B	open	v1.1	Tsinghua	openchat-13b (Wang et al., 2023a)	13B	open	v3.2	Tsinghua
codegeex2-6b (Zheng et al., 2023)	6B	open	-	& Zhipu	wizardlm-30b (Xu et al., 2023)	30B	open	v1.0	Microsoft
codellama-34b (Rozière et al., 2023)	34B	open	instruct		wizardlm-13b (Xu et al., 2023)	13B	open	v1.0	
codellama-13b (Rozière et al., 2023)	13B	open	instruct	Meta	koala-13b (Geng et al., 2023)	13B	open	-	UCB
codellama-7b (Rozière et al., 2023)	7B	open	instruct		oasst-12b (LAION, 2023)	12B	open	sft-4	LAION
dolly-12b (Conover et al., 2023)	12B	open	v2	Databricks					

https://arxiv.org/abs/2308.03688

LLMArena

- **LLMArena** a dynamic evaluation benchmark specially designed for multi-agent interaction.
- Covers **seven different types** of dynamic, **multi-agent game environments.**
- Focusses on assessing LLMs in scenarios that require **spatial reasoning, strategic planning, communication, opponent modeling**, and **team collaboration.**

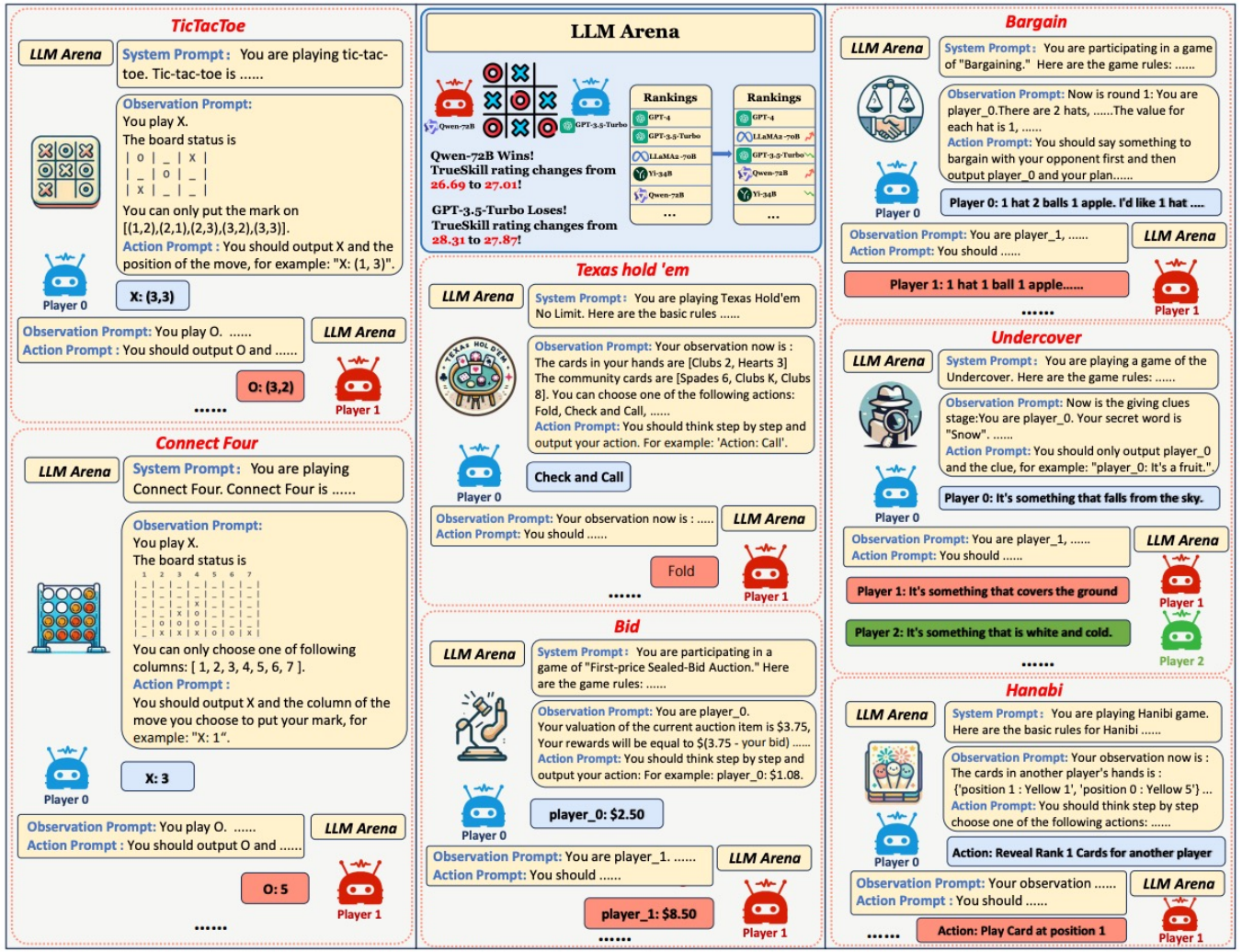


Figure 1: Examples of seven different game environments in LLMARENA.

https://arxiv.org/pdf/2402.16499

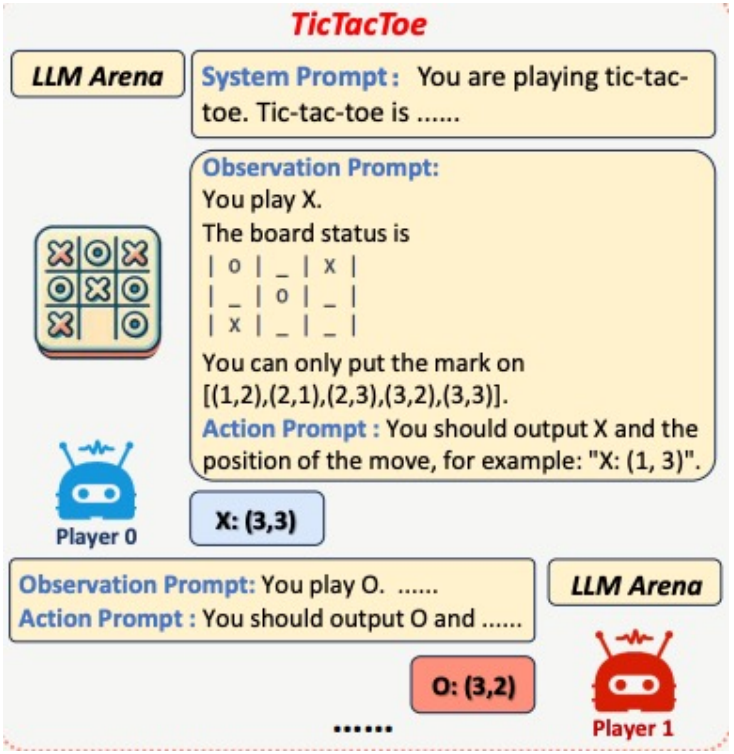
Prompt design

- **System Prompt.** System prompts are special messages used to steer the behavior of LLM. They allow to prescribe the agent's style and task within certain bounds, making it more customizable and adaptable for various use cases.

  In LLM Arena, we stipulate the role of the agent and the rules of the game in the system prompt.

- **Observation Prompt.** Observation Prompt provides the necessary state information for the agent to interact with the environment, including the opponent's actions, current game status, legal actions, etc. We use {} to frame these changing information.

- **Action Prompt.** Action Prompt guides the Agent to choose from legal actions and output regularized actions. In addition, to stimulate the agent's reasoning ability, in some environments, we also add CoT Prompt.



https://arxiv.org/pdf/2402.16499

Experiments and results

- As the scale of the model parameters increases, there is a noticeable improvement in the capabilities of LLMs.
- However, exceptions can arise in specific environments.
- Significant disparity between GPT-4 and other models. GPT-4 achieved SOTA across all evaluated tasks.

LLMs	TicTacToe	ConnectFour	Texas Hold'em	Bid	Bargain	Undercover	Hanabi	Average
GPT-4	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
GPT-3.5-Turbo	82.81	96.30	81.23	84.17	99.26	92.59	84.13	88.64
Qwen-72B-Chat	90.08	80.30	94.88	84.42	92.95	62.96	89.01	84.94
Llama-2-70B	82.21	95.90	98.42	94.22	88.63	85.19	80.35	89.27
Agentlm-70B	81.44	83.62	77.35	86.10	98.33	92.59	79.58	85.57
DeepSeek-67B-Chat	68.00	88.88	96.64	20.30	89.02	74.07	68.69	72.23
SUS-Chat-34B	79.75	96.30	84.38	66.17	74.53	88.89	86.33	82.34
Yi-34B-Chat	78.23	95.36	86.06	88.77	74.18	96.30	34.30	79.03
Qwen-14B-Chat	86.40	72.67	90.56	86.82	91.13	88.89	71.70	84.02
WizardLM-13B	79.14	83.35	74.26	35.28	91.28	48.15	54.23	66.53
AgentLM-13B	78.89	89.23	86.85	30.39	94.93	62.96	22.09	66.48
DeepSeek-7B-Chat	81.17	73.06	96.10	14.53	92.56	74.07	0.00	61.64
AgentLM-7B	68.29	90.35	85.14	0.04	83.53	66.67	98.20	70.32
Vicuna-7B	64.95	80.12	90.46	68.94	86.33	62.96	87.37	77.30
Average	80.10	87.53	88.74	61.44	89.76	78.31	68.28	79.16

Table 1: Normalized scores of 14 different LLMs in 7 environments.

https://arxiv.org/pdf/2402.16499

AppWorld: A Controllable World of Apps and People for Benchmarking Interactive Coding Agents*

Autonomous agents that address **day-to-day digital tasks** (e.g., ordering groceries for a household), must not only operate **multiple apps** (e.g., notes, messaging, shopping app) via APIs, but also **generate rich code with complex control flow** in an iterative manner **based on their interaction with the environment**.

AppWorld
=
AppWorld Engine is a high-quality **execution environment** of 9 day-to-day apps operable via 457 APIs and populated with realistic digital activities simulating the lives of ~100 fictitious users.
+
AppWorld Benchmark is a suite of 750 autonomous agent tasks requiring rich and interactive code generation.

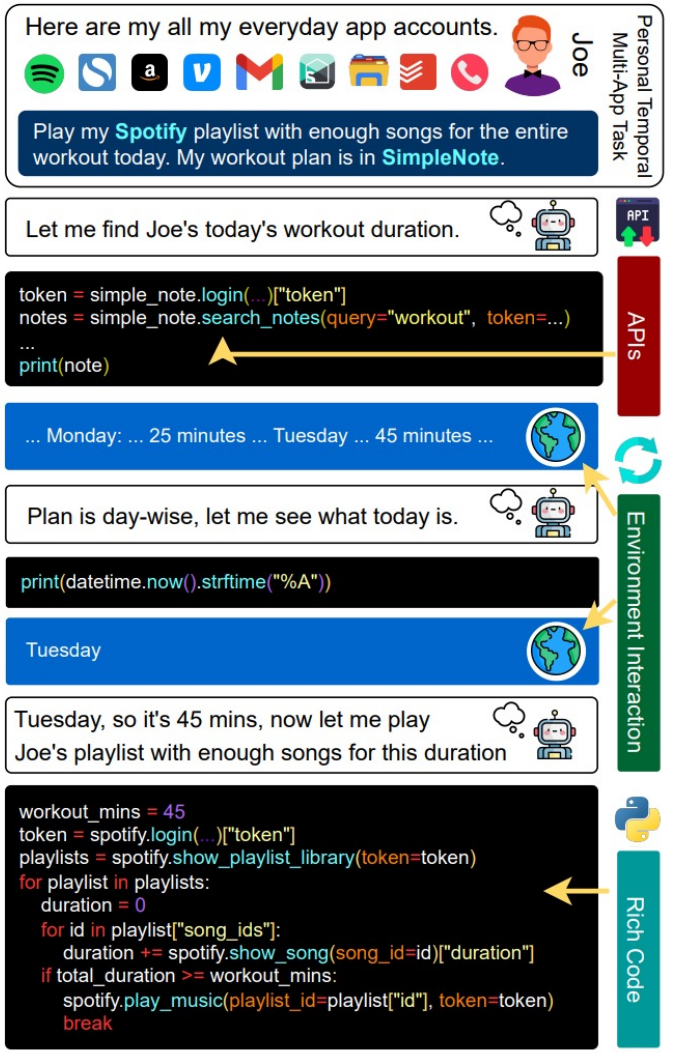


Figure 1: A (shortened) day-to-day task from AppWorld Benchmark requiring **rich code** with **environment interaction** and **API calls**.[2]

https://arxiv.org/pdf/2407.18901v1

AppWorld =

AppWorld Engine is a high-quality simulator of real-world apps and people using these apps, controlled via APIs:

a fully controllable, stable, and reproducible execution environment of simulated apps and people, where agents can operate apps via APIs without any real-world consequences (e.g., spamming emails) or resource usage (e.g., money for Venmo).

+

AppWorld Benchmark provides complex tasks and their evaluation suites, built using AppWorld:

750 complex tasks covering everyday scenarios using the apps in the AppWorld Engine. The tasks are natural, challenging, diverse, and carefully designed with distractors and hurdles so as to require thorough reasoning. Tasks cover multiple apps and rely on using many APIs in an intricate flow.

+

Programmatic and Robust Evaluation :

AppWorld Engine allows fine-grained control of the database underlying the apps

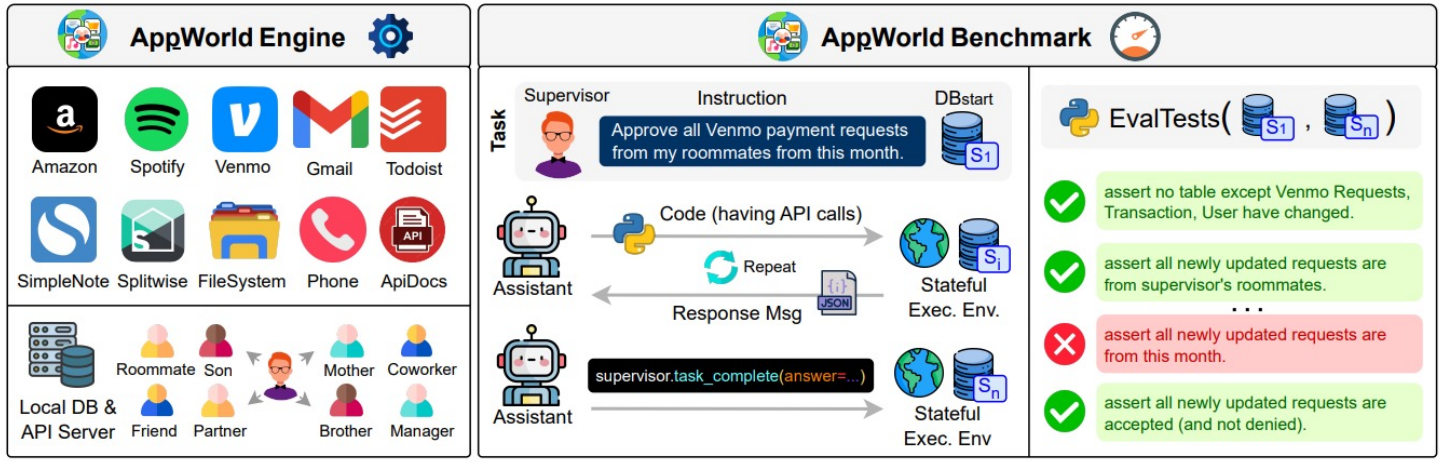


Figure 2: (Left) AppWorld Engine is a feature-rich API-based app simulator based on 9 real-world day-to-day apps populated with data simulating the digital activities of fictitious people living in this world. (Right) AppWorld Benchmark is a collection of complex everyday tasks requiring interactive coding with API calls, which come with reliable programmatic state-based evaluation suites robustly capturing goal completion criteria.

https://arxiv.org/pdf/2407.18901v1

Experiments and results

Base LLM	Method	Test-N TGC | SGC	Test-C TGC | SGC
GPT4^{O}	ReAct	**48.8 | 32.1**	**30.2 | 13.0**
	PlanExec	44.6 | 23.2	19.7 | 7.9
	FullCodeRefl	33.9 | 26.8	19.2 | 12.2
	IPFunCall	32.1 | 16.1	18.0 | 10.1
GPT4^{Trb}	ReAct	26.8 | 12.5	**17.5** | 5.8
	PlanExec	**32.7** | 16.1	11.0 | 3.6
	FullCodeRefl	25.6 | 19.6	12.5 | 7.2
	IPFunCall	30.4 | **21.4**	14.6 | **9.3**
LLaMA3	ReAct	20.8 | 8.9	3.4 | 0.0
	PlanExec	8.9 | 1.8	2.4 | 0.7
	FullCodeRefl	**24.4 | 17.9**	**7.0 | 4.3**
DeepSeek	ReAct	7.1 | 1.8	2.9 | 0.7
	PlanExec	1.8 | 0.0	0.7 | 0.0
	FullCodeRefl	**13.1 | 8.9**	**5.8 | 2.9**
Mistral-7B LLaMA	CodeAct	0.0 | 0.0	0.0 | 0.0
	ToolLLaMA	0.0 | 0.0	0.0 | 0.0

Table 3: Main Results: Task and scenario goal completion scores (TGC | SGC), demonstrating that AppWorld Benchmark is highly challenging for current models.

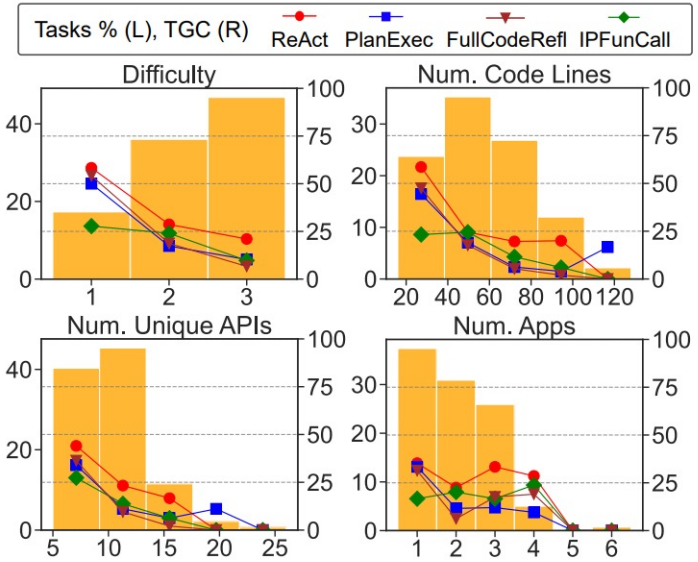
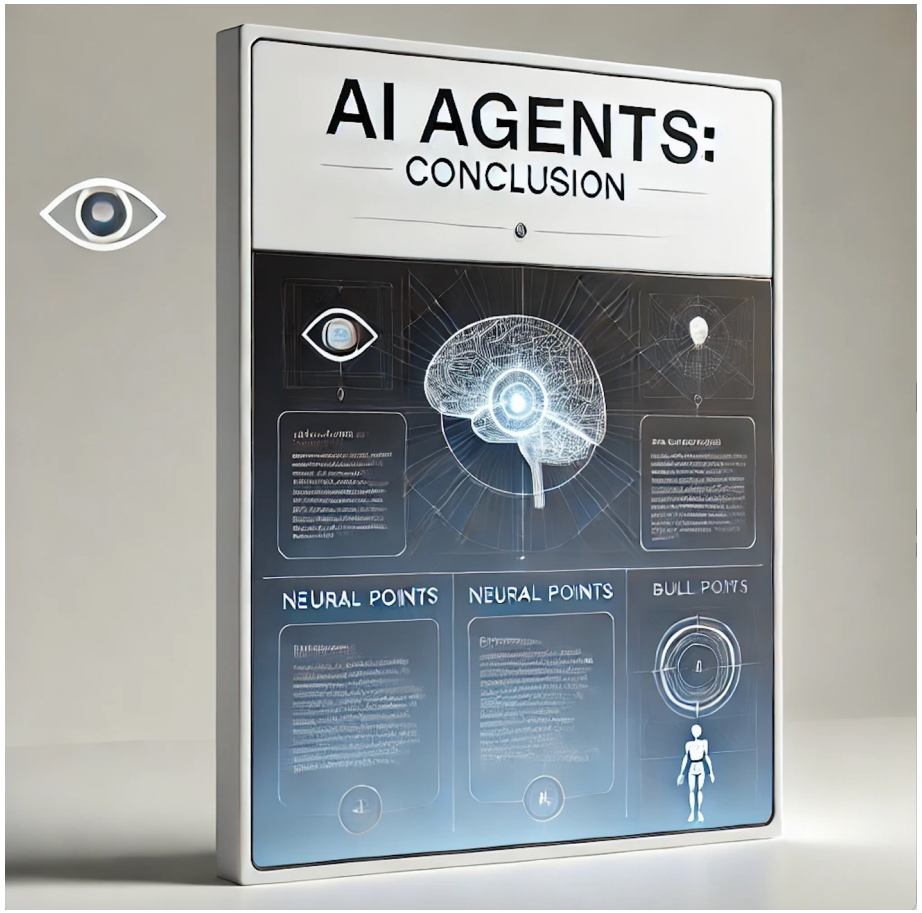


Figure 4: Percentage of Test-C tasks for hardness indicators (left axis) and TGC scores (right axis) for GPT4^{O}. All methods perform worse as difficulty increases.

All methods perform substantially worse as the difficulty increases.

https://arxiv.org/pdf/2407.18901v1



AI agents: conclusion

- New era of AI agents based on LLMs.
- A promising approach for multi-component, complex systems and tasks that a single model cannot handle
- Possibility of real-time actualization of knowledge without additional training.
- Tools for complex problems.