

Retreaval Augmented Generation

Thomas Haschka,
Simeon Harrison, Martin Pfister

RAG - Overview

What will you see today...

Theoretical Overview -
Retrieval Augmented Generation:

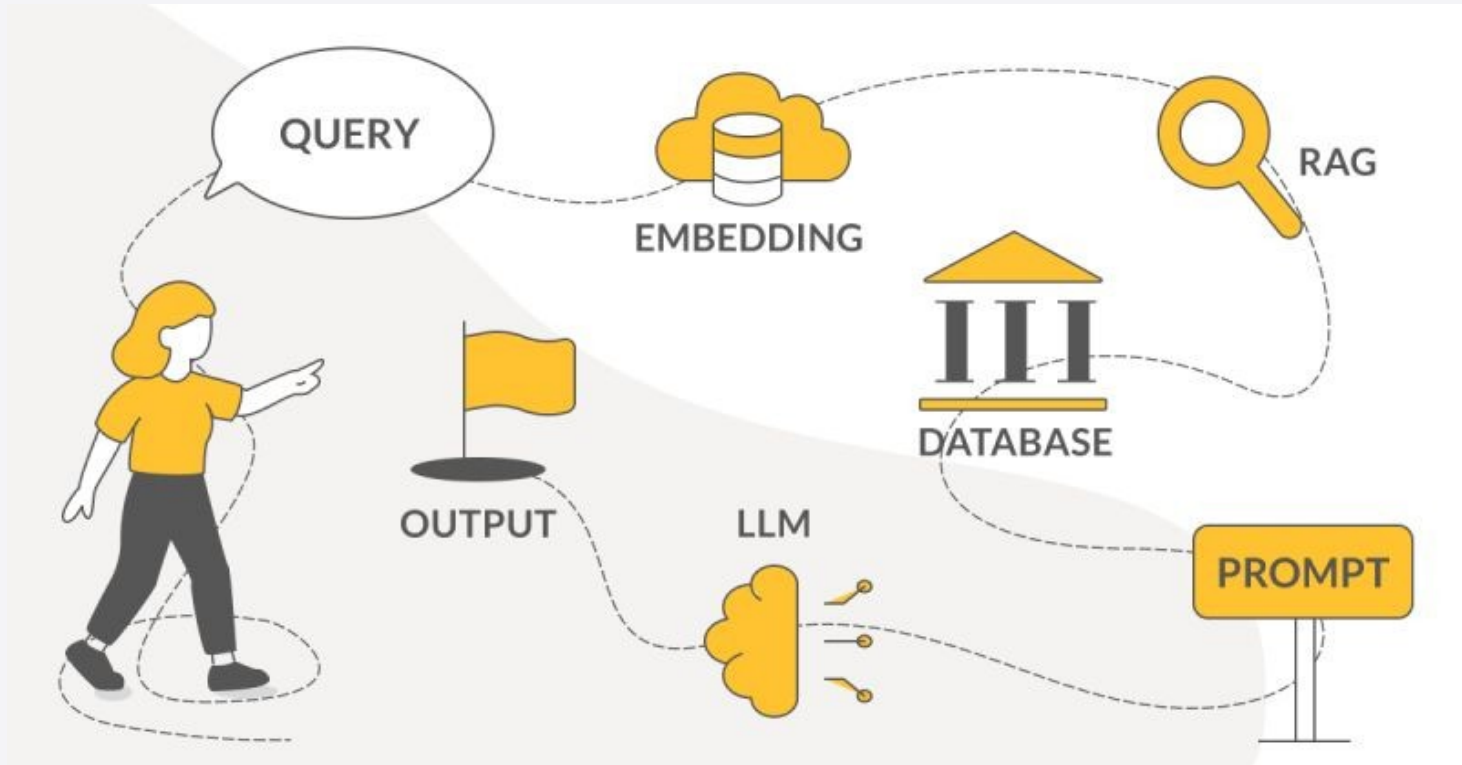
Discussion about the inner workings of
RAG, in order to understand what a:

- An Embedding Model
- A Vector Database
- A Large Language Model Engine
and
- REST

is and how we can orchestrate them.

- Practical I:
Scholarly implementation without 3rd
party tools fostering in depth
understanding.
- Discussion about the implementation
in Practical I and its shortcomings
- Practical II:
Implementation using contemporary
tools such as LangChain, ChromaDB
etc.

RAG Introduction



Why RAG

Include Information Outside the LLM Training Dataset

RAG allows us to augment an LLM prompt with information outside of what was stored / apprehended in the LLMs weights during training.

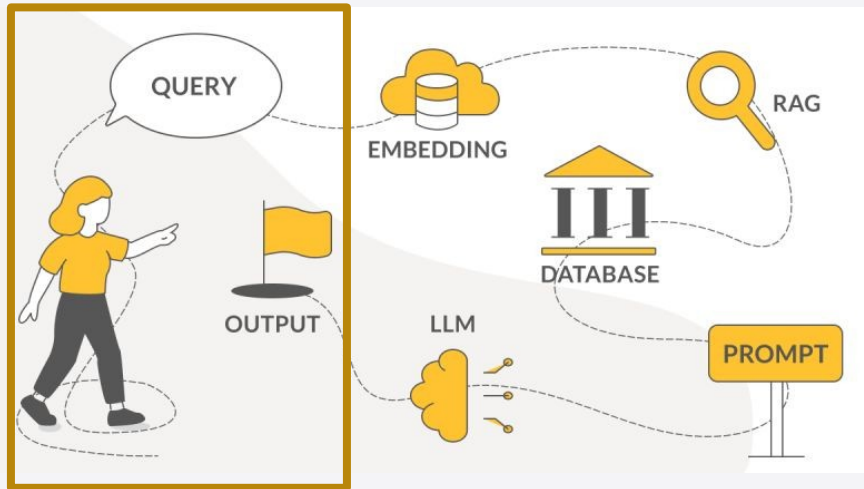
Perform Prompt Engineering on Steroids

It is the perfect tool to combine the power of databases / search engines with the power of LLMs.

It allows one to make ones own data available to the LLM and built custom chatbots with in house data.

Typical RAG Components

User Facing Interface

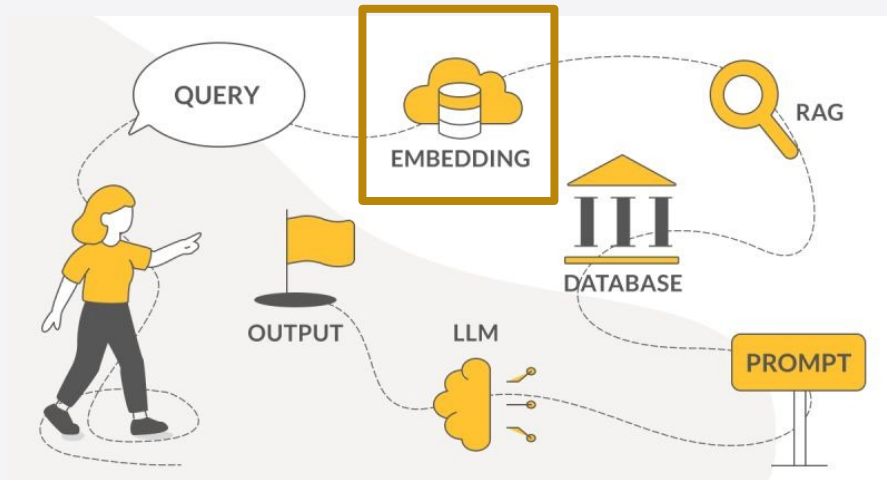


A typical RAG system acts using a **chatbot like interface**. A human is presented with an interface to ask questions, give commands, brief converse with the RAG-LLM machinery.

The importance here is also that the RAG process **keeps track of the questions as well as of the answers** in order to simulate a conversation, and allow the LLM to **«remember»** past questions and answers.

Typical RAG Components

2/ Embedding (Model)



For the sake of fast information retrieval most RAG implementations use so called embedding models.

Each human question asked is transformed into an embedding. This embedding is then used in order to efficiently query data in a database.

The embedding shall encode the semantics of the query. It allows us to find knowledge in a database with similar semantics.

Typical RAG Components

2/ How do Embeddings Work

Text A: The mother searched her kids at school

Embedding A:

2	1	0.5	2	1	1	2	1.5	0.5	1
---	---	-----	---	---	---	---	-----	-----	---

Text B: The mother searched her kids at the kindergarden

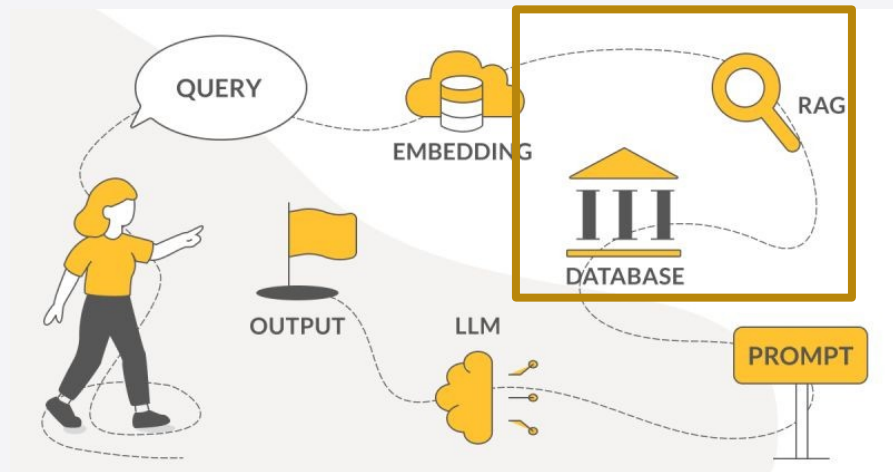
Embedding B:

2	2	4	3	2	1	2	1.5	0.5	1
---	---	---	---	---	---	---	-----	-----	---

Embedding models create vectors from text phrases which shall represent the inherent semantics of the Text. Semantically similar phrases shall be represented by «similar» vectors.

Typical RAG Components

3/ Phrase Distance Measure



In order to “retrieve” phrases with similar semantics in a database we have to define a Phrase Distance Measure (PDM):

The PDM is a composition of different factors:

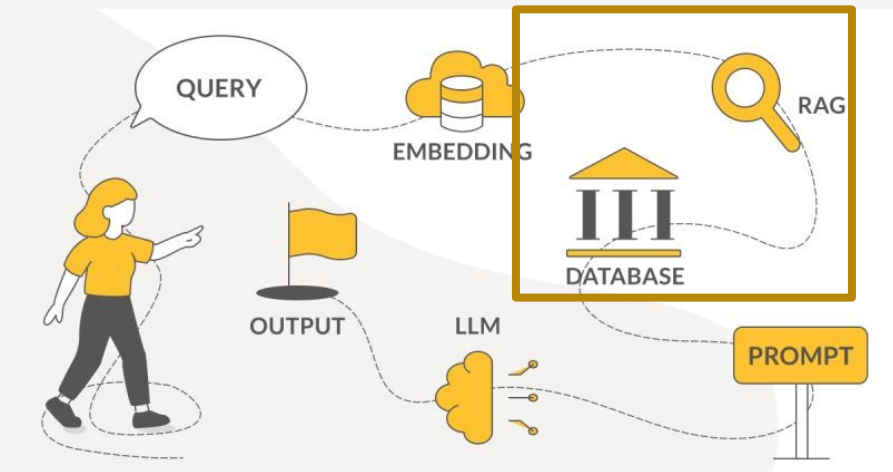
For phrases p_1 and p_2 :

$$D: (p_1, p_2) \mapsto d,$$

the PDM shall yield d , a scalar describing how “semantically” distance p_1 and p_2 are.

Typical RAG Components

3/ Phrase Distance Measure



In general the PDM is a composition of:

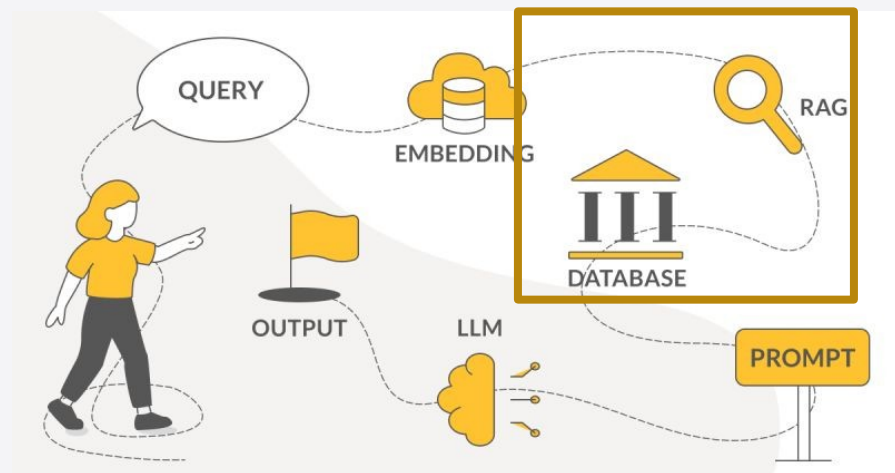
- The Tokenizer
- The Embedding Model
- The Distance Function

The most common Distance Function used is the Cosine Distance:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \cdot \sqrt{\sum_{i=1}^n B_i^2}}$$

Typical RAG Components

3/ Phrase Distance Measure



The Cosine Distance:

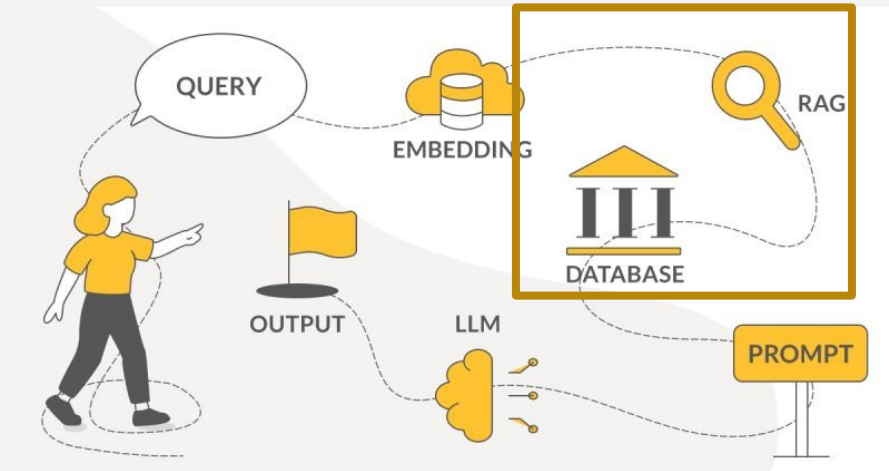
$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \cdot \sqrt{\sum_{i=1}^n B_i^2}}$$

is efficient to compare vectors in high dimensional spaces, which is not the case for the classical Euclidean L2 or Manhattan distance L1.

Semantically similar texts shall have a distance close to +/-1, while dissimilar texts have a distance close to 0.

Typical RAG Components

4/ The Vector Database



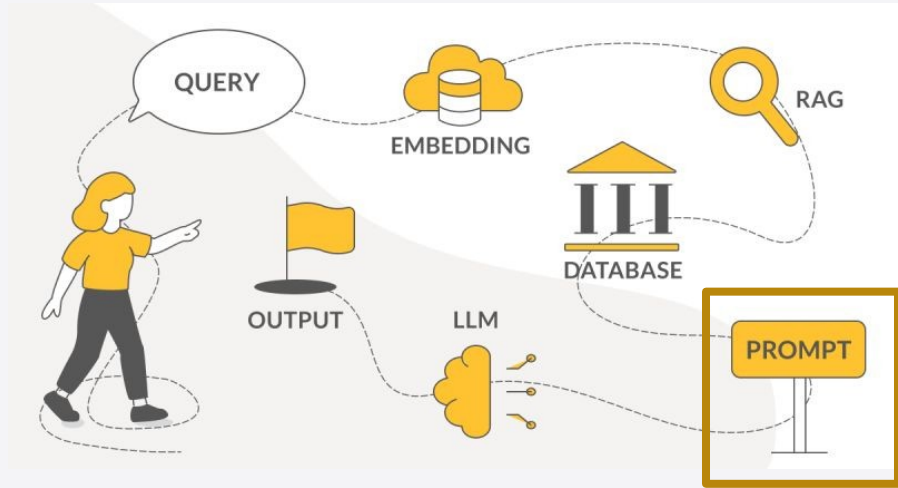
LLMs have limited context sizes. As such we can not deliver the whole information at once, (even if the data is stored in single large document) to the LLM.

As such we create slices of adequately sized texts. For each of these text slices we create embeddings.

Semantically similar slices, in general a bunch of them, shall be selected and added to the prompt, to be later processed by the LLM together with the query.

Typical RAG Components

5/ The Prompt Assembly



The prompt is key to every RAG system:

The RAG process modifies the prompt, by injecting data into it, and delivering it as such to the LLM.

RAG can therefore be considered as prompt engineering on steroids.

Typical RAG Components

5/ The Prompt Assembly – What is a LLM ?

- A LLM is large language model, a ChatGPT like construct.
- It automatically generates text, correctly trained this text generation can result in conversation like chats.
- A large language model can predict the next “Token” from a series of prior “Tokens”.
- A “Token” is a piece of word, or has some special conversational properties.
i.e. end-of-turn Token which hands over the conversation to the enquirer.
- LLMs are trained against a special conversational “Token” construct, described in the Chat Template.

Typical RAG Components

5/ The Prompt Assembly – Chat Template

```
<|begin_of_text|>  
<|start_header_id|>system<|end_header_id|>  
{{ system_prompt }}  
<|eot_id|>
```

```
<|start_header_id|>user<|end_header_id|>  
{{ user_msg }}  
<|eot_id|>
```

```
<|start_header_id|>assistant<|end_header_id|>  
{{ model_answer }}  
<|eot_id|>
```

User / RAG
Client Supplied

LLM generated
until <|eot_id|>

Typical RAG Components

5/ The Prompt Assembly – Finished Prompt

<|begin_of_text|>

<|start_header_id|>system<|end_header_id|>

You are a kind chatbot trying to answer to the users input as accurate as you can. Take the following contexts found in our database into account:

1. A new REST service for corporate weather data.
The API works as follows . . .
2. Cafeteria proposes new services and menus.

. . .
<|eot_id|>

<|start_header_id|>user<|end_header_id|>

Please write a report about our new services.
<|eot_id|>

<|start_header_id|>assistant<|end_header_id|>

A report on our new services:

The cafeteria offers from tomorrow onwards eggs and bacon for breakfast. . .
<|eot_id|>

Engineered Prompt to enforce LLM behaviour

RAG Data injected into the prompt

Human question asked to the LLM

User / RAG Client Supplied

LLM generated until <|eot_id|>

Typical RAG Components

5/ The Prompt Assembly – Enforce Factual Answers using RAG

```
<|begin_of_text|>
```

```
<|start_header_id|>system<|end_header_id|>
```

```
You are a kind chatbot trying to answer to  
the users input as accurate as you can.  
Take the following contexts found in our database  
into account:
```

```
1. A new REST service for corporate weather data.
```

```
    The API works as follows . . .
```

```
2. Cafeteria proposes new services and menus.
```

```
    . . .
```

```
IF THE ANSWER CAN NOT BE DERIVED FROM THE ABOVE  
CONTEXT, REPLY THAT YOU DO NOT KNOW THE ANSWER.
```

```
<|eot_id|>
```

```
<|start_header_id|>user<|end_header_id|>
```

```
Please write a report about our new services.
```

```
<|eot_id|>
```

```
<|start_header_id|>assistant<|end_header_id|>
```

```
A report on our new services:
```

```
The cafeteria offers from tomorrow onwards  
eggs and bacon for breakfast. . .
```

```
<|eot_id|>
```

Engineered Prompt to
enforce LLM behavior

RAG Data injected
into the prompt

Human question
asked to the LLM

User / RAG
Client Supplied

LLM generated
until <|eot_id|>

Typical RAG Components

5/ The Prompt Assembly – Perform Self Reflection

<|begin_of_text|>

<|start_header_id|>system<|end_header_id|>
You are a kind chatbot trying to answer to
the users input as accurate as you can.
Take the following contexts found in our database
into account:

1. A new REST service for corporate weather data.
The API works as follows . . .
2. Cafeteria proposes new services and menus.
. . .

IF THE ANSWER CAN NOT BE DERIVED FROM THE ABOVE
CONTEXT, REPLY THAT YOU DO NOT KNOW THE ANSWER.
<|eot_id|>

<|start_header_id|>user<|end_header_id|>
Please write a report about our new services.
<|eot_id|>

<|start_header_id|>assistant<|end_header_id|>
A report on our new services:

The cafeteria offers from tomorrow onwards
eggs and bacon for breakfast. . .
<|eot_id|>

Engineered Prompt to
enforce LLM behavior

RAG Data injected
into the prompt

Human question
asked to the LLM

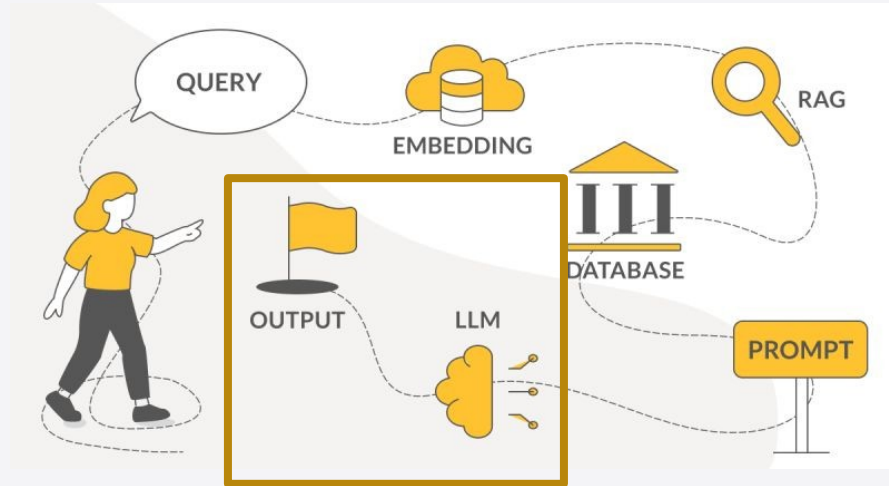
Reinject first LLM Answer into
the Prompt and ask for self
correction. Only yield 2nd pass.

User / RAG
Client Supplied

LLM generated
until <|eot_id|>

Typical RAG Components

5/ The LLM



Finally the large language model processes the supplied prompt and generates its output.

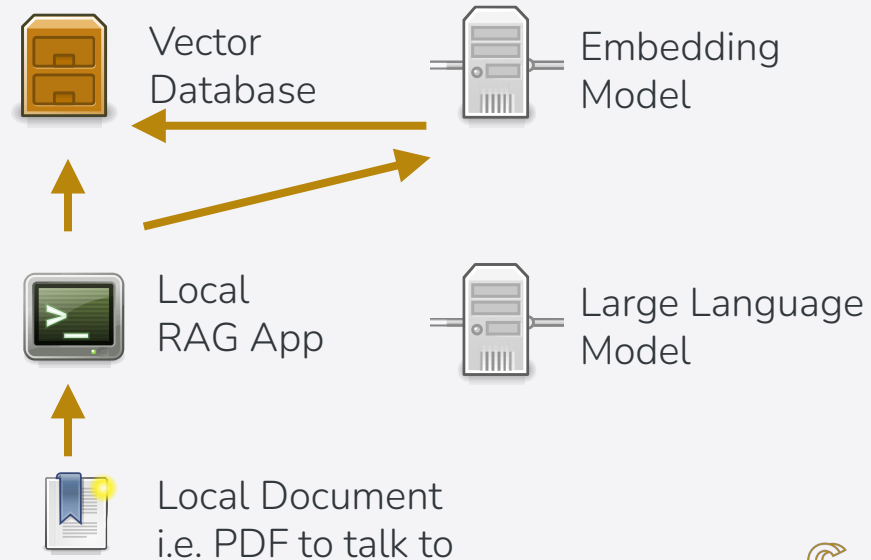
The result is in generally appended to the next prompt, in order to provide history to the conversation.

If the prompt, and conversation gets to large, (larger then the LLMs context window) one has to think about solutions about how to «compress» the prompt. i.e. by asking the LLM to summerize the preceding conversation.

Practical Setup

1/ Building A Vector Database

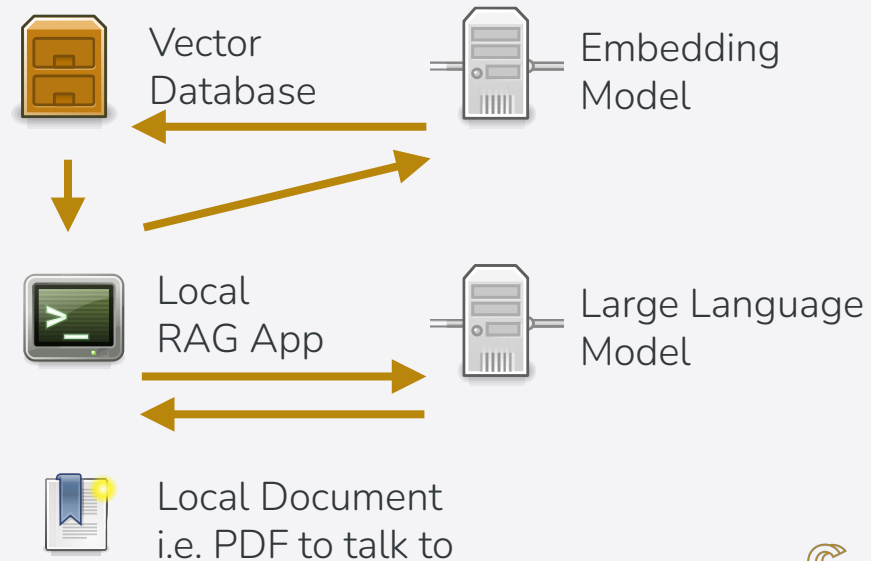
- A local RAG Application reads documents and transforms them to text.
 - In simple cases: direct transfer with tools like «pdftotext»
 - More elaborate chains can be built. i.e. with image extraction and AI models to build descriptions from tables / graphs.
- The extracted text has to be split in tiny parts, to fit the context window. The split size, and also the overlap of the text passages, is to be parameterized.
- The Embedding Model is used create Embedding Vectors for each text passage to be stored in the Vector Database.



Practical Setup

2/ Performing a Query

- A local RAG Application reads in a human generated query.
- An Embedding Vector, is created using an Embedding Model.
- The Vector Database is queried for “semantically close” text passages.
- The local RAG Applications builds from the user query and the returned documents the Prompt.
- The prompt is sent to the Large Language Model which provides an Answer.



Practical Setup

Bill of Materials – What do we Need

- LLM server, answering to prompts [through a REST API]
- A server that generates embeddings [through a REST API]
- A program that follows the conversation, queries the embedding server as needed and yields a prompt to the LLM comprising of:
 - Conversation History
 - Retrieved Data (from a vector database)



Vector
Database



Embedding
Model



Local
RAG App



Large Language
Model



Local Document
i.e. PDF to talk to

Practical Setup

Bill of Materials – Typical

- Engine for Embedding and LLM model
 - Llama.cpp
 - And/or run the model in python with torch, huggingface api. Implement REST by hand.
- A Vector Database
 - Use ChromaDB, Picone, etc.
- A program that follows the conversation, queries the embedding server as needed and yields a prompt to the LLM comprising of:
 - Use Langchain
 - And/or Implement by Hand



Vector
Database



Embedding
Model



Local
RAG App



Large Language
Model



Local Document
i.e. PDF to talk to

The Art of RAG

Main Issues

- Even with vector databases getting the right information is hard.
- The prompt has to be carefully crafted for the proposed usecase.
- The system has to be tested and validated.

The core challenge in building a performing RAG system is to optimize the retrieval process and to achieve a high accuracy in fetching the right documents for a given query.

The Art of RAG

Evaluating a RAG system:

- There is no clear evaluation path for RAG and what works for you is definitely dependent on your use case.
- Nevertheless we give you some hints on how to evaluate the performance of your RAG system, and its document retrieval process.

Document retrieval primarily is linked to the quality of the phrase distance measure introduced earlier.

$$D: (p_1, p_2) \mapsto d,$$

The PDM is a cascade of the following steps:

- Tokenization of phrase
- Building embeddings from these tokens
- Evaluating a distance measure

The Art of RAG

Evaluating a RAG system:

- The PDM is evaluated in a multistage process

$$T(p_1) = \vec{v}_{p_1},$$

$$T(p_2) = \vec{v}_{p_2},$$

$$V(\vec{v}_{p_1}, \vec{v}_{p_2}) = d,$$

Embedding vectors are created.

Embedding vectors are compared and a distance is returned.

The Art of RAG

Evaluating a RAG system – Finding the optimal PDM:

Finding the optimal PDM is hence a process of finding:

- the optimal **Tokenizer**.
- the optimal **Embedding Model**.
- the optimal **Distance Measure**.
- Better PDMs should yield better document retrieval and hence, better LLM answers.

The Tokenizer is often tied to the Embedding Model.

The Embedding Model is where we have the largest choice. How can we evaluate if we have a performant - **in terms of compute power/retrieval accuracy** - embedding model ?

The optimal distance measure is in most cases **the cosine distance**, but other solutions do exist.

The Art of RAG

Evaluating a RAG system – Hand Annotated Datasets

Hand Annotated Datasets:

- MTEB, BEIR, etc. Dataset
Specially crafted to yield documents to supplied questions.
- Better PDMs should yield better scores.
- Hugging Face leaderboard
<https://huggingface.co/spaces/mteb/leaderboard>

In practice new models incorporate such datasets in their training process, yielding the right embeddings for optimal retrieval on the BEIR dataset, but probably failing in your specific usecase.

Hence, **always validate your usecase** using in house experts.

The Art of RAG

Evaluating a RAG system

The Silhouette / Distance Matrix Correlation Method

You need a known dataset of **semantically close** and **semantically distant** phrases

i.e. articles about the same topic written by different authors. (semantically close = same meaning)

Articles about different topics (semantically distant = different meaning)

You verify that:

Same meaning is close in distance

Different meaning is far in distance

The advantage of this method is that the dataset to be used for this method can be relatively small.

You can use your in House data.

i.e:

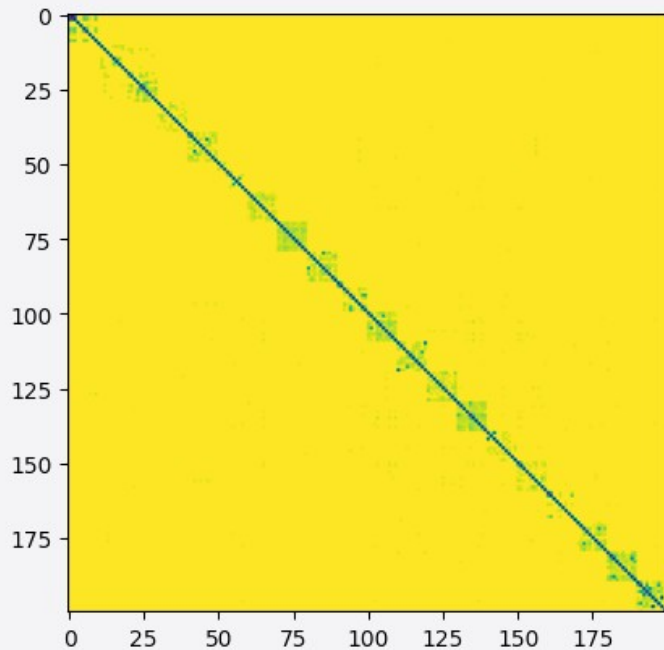
Medical records with the same outcome written by different doctors (semantically similar)

vs.

Medial records with different outcomes (semantically different)

The Art of RAG

Evaluating a RAG system - Distance Matrix Correlation



Optimal if you see something like green squares in yellow:

Green is semantically close texts.

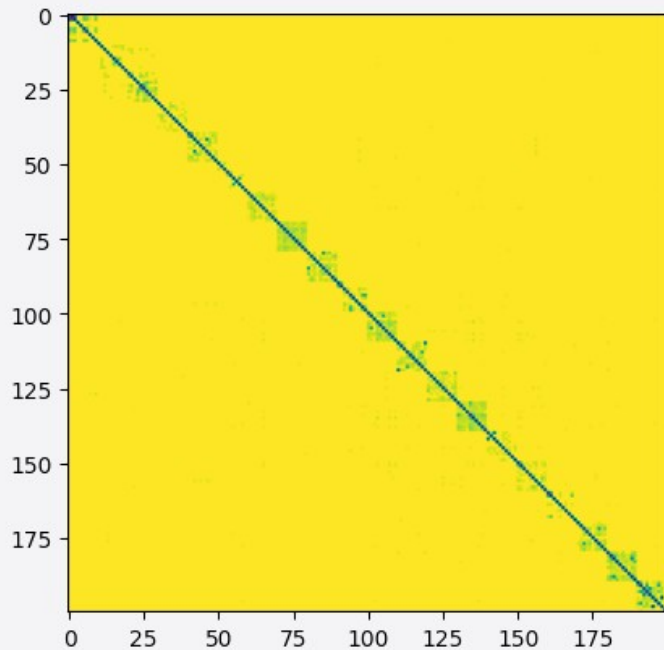
Yellow are semantically different texts.

Here printed for 20 different news articles, written by 10 different outlets.

Hence,
10 times the same semantic content, for
20 different writing styles.

The Art of RAG

Evaluating a RAG system - Distance Matrix Correlation



You can now create an artificial matrix,
and set distances for

semantically similar articles = 0
semantically different articles = 1

And calculate the Pearson correlation
between the distance matrix for a model
in question, and the optimal created one.

The closer the correlation is to 1 the
better your semantic differences are
represented by the embedding model.

The Art of RAG

Evaluating a RAG system – Silhouette Index

$$k_{j,a} = \frac{1}{n_a} \sum_{i=1}^{n_a} D(p_a(i), p_a(j)),$$

Normalized sum of inter cluster distances.
Distances between phrases of same meaning.

$$k_{j,b} = \frac{1}{n_b} \sum_{i=1}^{n_b} D(p_b(i), p_a(j)),$$

Normalized sum of intra cluster distances.
Distances between phrases of different meaning.

$$s_j = \begin{cases} k_{j,a} < k_{j,b} : & 1 - \frac{k_{j,a}}{k_{j,b}} \\ k_{j,a} = k_{j,b} : & 0 \\ k_{j,a} > k_{j,b} : & \frac{k_{j,b}}{k_{j,a}} - 1 \end{cases}$$

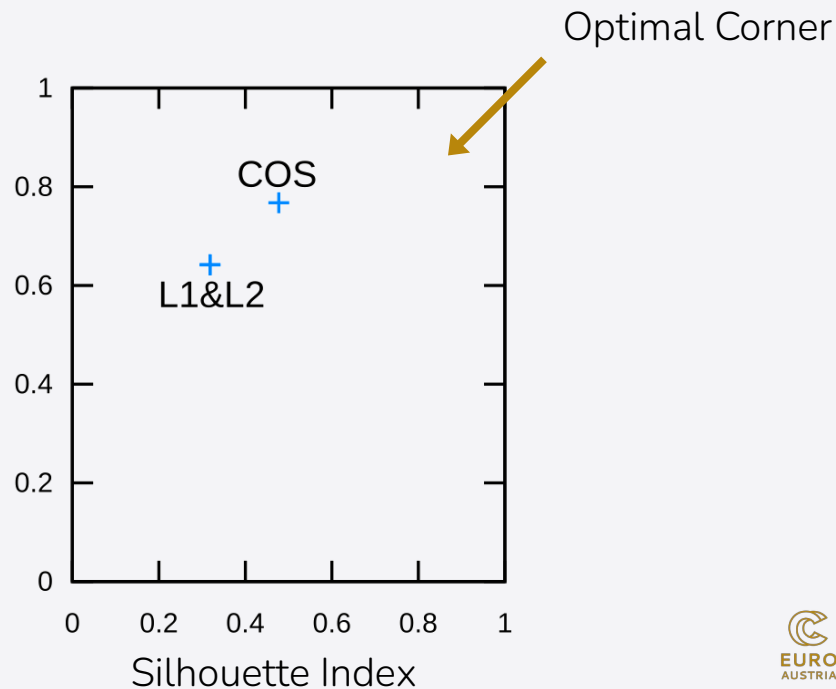
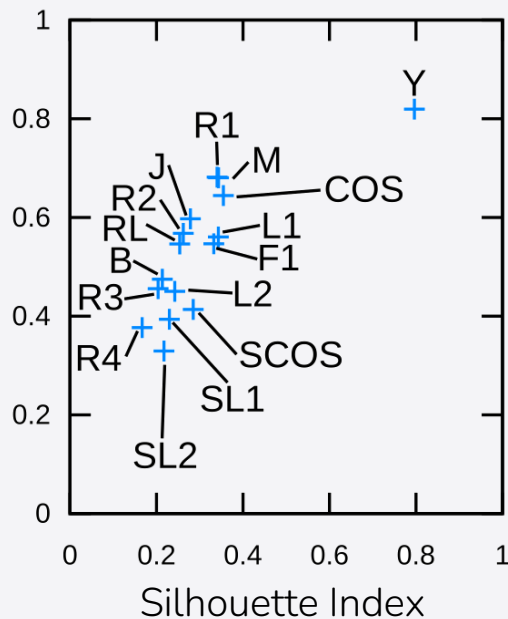
$$S = \frac{1}{n_a} \sum_{j=1}^{n_a} s_j,$$

If intra-cluster distances >> inter-cluster distances, which is what we want S shall be close to 1

The Art of RAG

Evaluating a RAG system – Using Both Indices

Distance Matrix Comparison
Pearson Correlation – Element Wise
with Optimal Distance Matrix



Further IDEAS

Things that you can further do with your RAG system:

- Incorporate texts from all kinds of databases, with other retrieval systems than vector databases:
 - SQL based search
 - GRAPH based search
- Use search engine based search: i.e. using Apache Lucene: <https://lucene.apache.org>
- Perform web searches from queries and include their results in the prompt.
- What you include in the prompt is limited only by the context size of your LLM

STAY IN TOUCH



eurocc-austria.at



vsc.ac.at



THANK YOU



This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia