ARTICLE

Building high-quality RAG systems

The intricacies of creating high-quality, context-aware generative AI systems

By Jesper Alkestrup
jeal@thetechcollective.eu
The Tech Collective

Nikolaj Purup
nipu@implement.dk
Implement Consulting Group

An introduction to building high-quality RAG systems, including specific recommendations for non-English systems.

RAG stands for retrieval augmented generation and has become one of the hottest terms in search technology and generative AI in 2023 – likely to continue in 2024. And no wonder it has.

RAG is transforming how organisations utilise their vast quantity of existing data to power intelligent chatbots and to automate complex workflows much faster than ever before. A successful RAG system can draw on an organisation's collective knowledge to function as an always-available, in-house expert to deliver relevant answers, grounded in verified data.

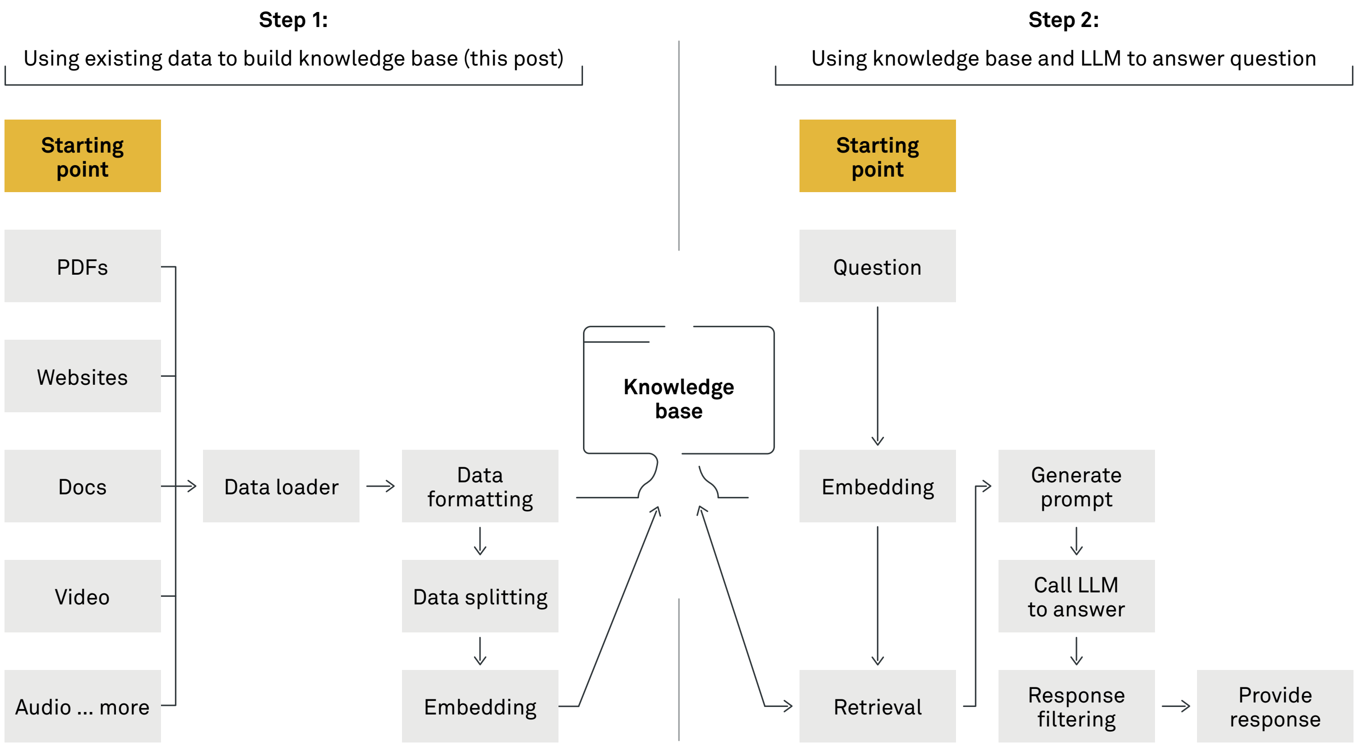
At Implement, we have developed RAG systems both for public and private clients, solutions deployed both for internal and external use and tools that are utilised in multiple languages. In this introduction, we will walk you through the primary steps required to set up a RAG system and especially highlight some of the intricacies when implementing generative AI for non-English languages such as Danish, German or Spanish.

RAG structure, a brief overview

Before we deep dive into the specific steps of setting up a RAG system, a simple way to view a RAG system is by dividing it into two core components:

1. **Indexing phase:** This initial stage involves processing the input data. The data is first loaded, appropriately formatted and then split. Later, it undergoes vectorisation through embedding techniques, culminating in its storage in a knowledge base for future retrieval.
2. **Generative phase:** In this phase, a user's query is input to the retrieval system. This system then extracts relevant information snippets from the knowledge base. Leveraging a large language model (LLM), the system interprets this data to formulate a coherent, natural language response, effectively addressing the user's inquiry.

Overview



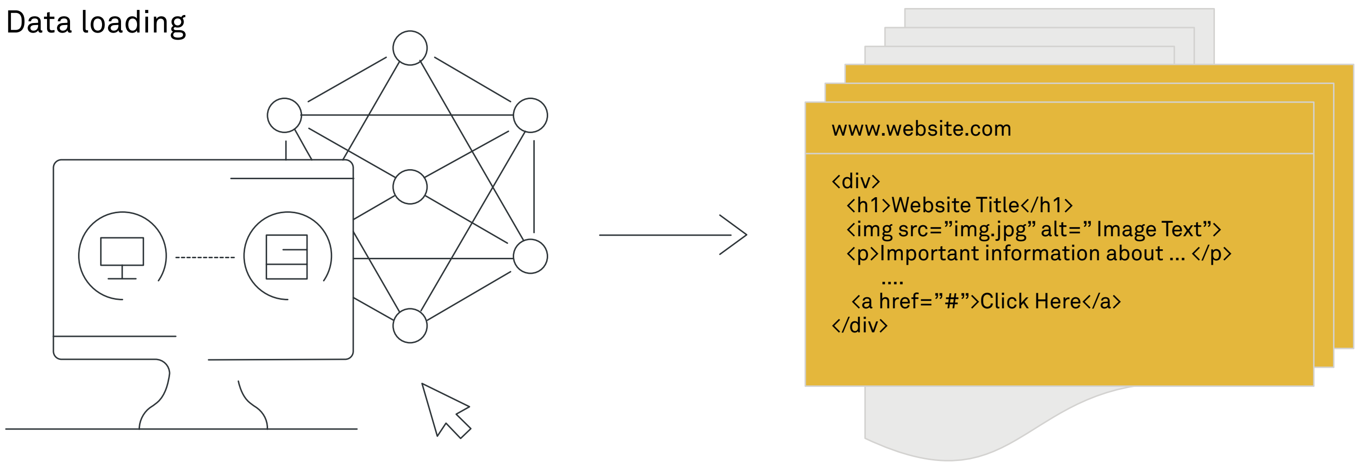
Now, let us walk through the process step by step, from loading the data all the way to generating an answer.

The indexing phase

The initial indexing phase involves processing the input data, ensuring all relevant data is stored in the knowledge base. We then move on to the generative phase.

1. Data loader: the devil is in the details

Data loading



```
www.website.com

<div>
  <h1>Website Title</h1>
  <img src="img.jpg" alt=" Image Text">
  <p>Important information about ... </p>
       ....
   <a href="#">Click Here</a>
</div>
```

Your organisation already possesses a wealth of knowledge, and the goal of a RAG system is to tap into this knowledge. It could be anything from Word documents, PDFs and videos to website content. The first step of setting up a RAG system is, therefore, to make this information accessible to the RAG system by setting up "data loaders". These are tools designed to collect and organise data from various sources, such as SharePoint, Dropbox or your company's website, and prepare it for the RAG system to analyse and utilise.

You need to set up data loaders for each type of file format your RAG system should handle. Default document loaders for various text formats are generally language agnostic. However, it is important that you keep the original format of documents, such as headers and paragraph layouts. This syntactic structure makes information retrieval more effective in the later stages of the system. Without syntactic structure and relevant metadata, locating specific information in a document can become much more difficult. This is particularly true for non-English languages that often do not have robust tools available for splitting text post-data loading in a semantically meaningful way.

When it comes to multimedia content such as videos and audio recordings, there is a significant variability in performance on the ability to convert spoken language into text. We see that technologies such as Whisper v3 from OpenAI show promise in handling multiple languages but are performing less accurately in non-English languages. It is common practice to assess language-specific speech-to-text performance by reviewing standard benchmarks, but when it comes to audio, you will often need to do internal tests to ensure that the model works as intended in your specific use case.

Starting with a standard data loader is generally a practical approach, but it is also important to closely inspect the processed data. This inspection can reveal what data, formatting or structural elements might have been lost in the process. When you understand these gaps, you can make targeted improvements and enhance the ability of the RAG system to search through and retrieve the right information from your organisation's knowledge base.

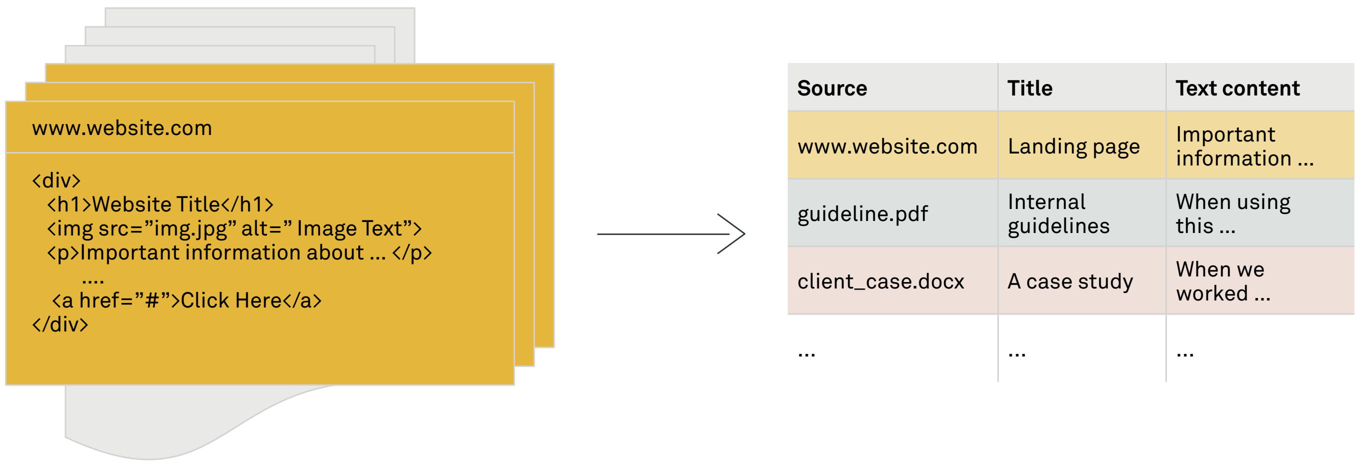
In essence, ***setting up your RAG system begins with gathering all the pieces of knowledge scattered across your organisation into a unified, accessible format.*** This process lays the foundation for leveraging AI to unlock new insights and efficiencies in your business.

In this first step, you also need to be diligent with document access by following best practices in data management. You cannot bypass existing information restrictions in this first step, as bypassing restrictions here might lead users of the final generative product to generate responses that include information that they were not supposed to have access to.

An inconvenient yet necessary insight attained in this step is that we often see clients detecting broken data management processes in this part of the RAG project, leading to necessary but frustrating setbacks to preserve the information security of the company.

2. Data formatting: the key to contextual responses

Data formatting



```
www.website.com

<div>
  <h1>Website Title</h1>
  <img src="img.jpg" alt=" Image Text">
  <p>Important information about ... </p>
      ....
   <a href="#">Click Here</a>
</div>
```

Source	Title	Text content
www.website.com	Landing page	Important information ...
guideline.pdf	Internal guidelines	When using this ...
client_case.docx	A case study	When we worked ...
...	...	...

In the first step, you loaded your organisation's diverse knowledge through data loaders. The next step is data formatting, a process aiming at ***standardising the format of collected data.*** This ensures that the data is in the best shape for the subsequent phase, which involves breaking down the text into smaller, meaningful pieces.

Imagine taking the rich, complex information contained in various formats like HTML or PDF files and simplifying it into plain text. This simplification involves organising the text with basic markers to make the upcoming task of dividing the content more manageable. In sophisticated RAG systems, where multiple answers might fit a given context, it is valuable to organise additional details such as document titles, headers and topics alongside the text. This information, treated as metadata, aids in later stages for refining searches or analyses. For instance, when segmenting the text into chunks, knowing the topic of a particular chunk can significantly enhance the system's ability to provide contextually relevant responses.

After the second step, you have all data in the same format and are now ready to begin text splitting. The third step is crucial for enabling semantic search in your RAG system, allowing it to efficiently analyse and make use of the information from your organisation.

3: Text splitting: size matters

Text splitting

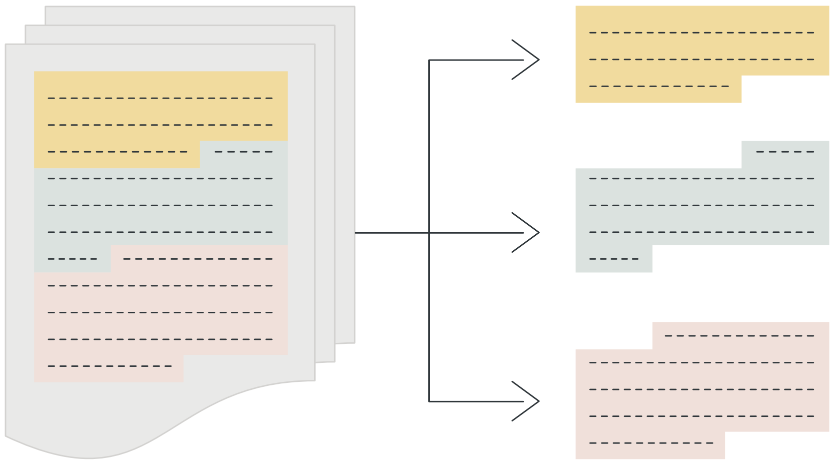


Illustration: splitting documents into smaller chunks

In the process of developing a retrieval augmented generation (RAG) system, ***splitting text into appropriately sized chunks is a necessary third step to allow efficient search.***

The necessity of text splitting stems from the reliance on semantic search to find pertinent pieces of information. Semantic search operates through a vector search mechanism, which requires converting texts of limited length into embeddings.

Embeddings are a way to translate the semantic meaning of text into a numerical format, represented as multidimensional vectors. For those looking to dive deeper into the concept of embeddings, "Getting started with Natural Language Processing" offers a more detailed walkthrough.

Two primary factors influence the ideal text size that existing data needs to be split into: *model constraints* and *retrieval effectiveness.*

Model constraints

Every embedding model has a maximum limit on how much text can be compressed into a single vector, known as the maximum token length. Exceeding this limit means the extra text will be ignored and lost from analysis. This is particularly limiting for multilingual models, which historically have more restrictive limits than their English-only counterparts. For example, a popular multilingual model such as MiniLM-L12-v2 only handles up to 128 tokens (roughly ~ 60 words), whereas the commercial Ada-02 embeddings from OpenAI currently support up to 8,191 tokens.

Additionally, a model's performance can be affected by the length of text it was originally trained on, potentially reducing the model's efficiency on retrieval for longer texts – despite being able to manage long inputs.

Retrieval effectiveness

It might seem alluring to match text chunk length to the model's maximum length for embedding. However, this approach does not typically yield the best retrieval outcomes.

Larger text chunks can offer more context, aiding the semantic search in understanding the broader meaning. However, they can also make it more challenging to identify specific search terms. Smaller chunks, conversely, may improve the accuracy of matching specific queries but risk lacking the context needed for fully informed answers. Employing a hybrid strategy – using smaller chunks for detailed search and integrating additional context for context retrieval – can provide a more performant solution. It is vital that you understand the importance of chunk size no matter if your project involves multiple languages or is solely in English.

Choosing the right size for text chunks in a retrieval augmented generation (RAG) system is not a one-size-fits-all solution. The ideal chunk size varies depending on your specific use case, including the file formats you are working with and the nature of your search tasks. There is no universal answer, but through experimentation and testing, you can determine the most effective chunking strategy for your needs.

Text splitting: techniques explained

You can split text into manageable segments using different methods, each suited for specific needs and contexts. The methods generally fall into two main categories: *rule-based and machine learning (ML)-based* approaches.

Rule-based splitting: The rule-based method relies on analysing the characters in the text to find logical places to split, such as punctuation marks or specific string patterns. It is straightforward and does not require the text to be understood in its semantic context, making it a quick and resource-efficient option. It does, however, require that syntactic structure be preserved when loading the data.

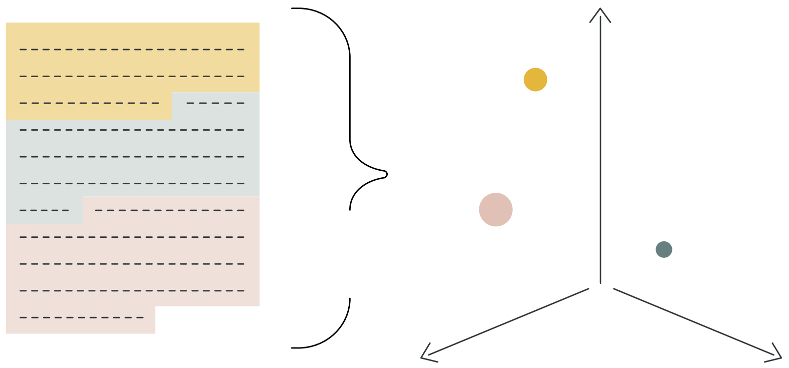
Machine learning-based splitting: ML-based methods leverage pre-trained transformer models, typically trained in English, to perform tasks such as sentence splitting or identifying semantic sections in the text. While some models support multiple languages, their accuracy outside English is currently not great, and they require significantly more computational resources than their rule-based par.

Given the challenges with ML-based splitters for non-English languages, a simple rule-based approach is often recommended for starters. If the structure of the original data is maintained and formatted correctly, rule-based splitting can yield high-quality results.

Through the third step, you have segmented the text into semantically meaningful chunks using either rule-based or ML-based methods. Now, you are in a position to proceed with embedding these chunks for efficient storage and retrieval. The fourth step is foundational in preparing your data for a RAG system, ensuring that the system can access and analyse the information effectively.

4. Embedding models: navigating the jungle

Text embeddings



Embedding models convert text to vectors in a multidimensional space

As a crucial fourth step in developing an efficient retrieval augmented generation (RAG) system, you must now select the appropriate embedding model. The choice can be more complex for languages other than English. The model you choose should be either multilingual to cover a wide range of languages or specifically designed for the particular language you are focusing on. It is worth noting that the most advanced models are typically optimised for English and may not perform as well with other languages.

When you compare embedding models, you can use the massive text embedding benchmark (MTEB) as a helpful tool for comparison. MTEB evaluates models across more than 100 languages, providing a broad overview of their capabilities. Language-specific benchmarks can be valuable for tasks like classification, offering insights into the best models for languages from Danish to Spanish. However, keep in mind that these benchmarks might not fully reflect a model's effectiveness in a RAG system, where the goal is different from tasks like classification.

A practical tip is to start with the highest-rated multilingual model in the MTEB retrieval benchmark, keeping in mind that its retrieval score is based on English. Therefore, it is necessary for you to conduct your own tests in your target language to accurately gauge performance.

What is the MTEB benchmark?

The massive text embedding benchmark (MTEB) is the largest benchmark for text embeddings, combining 56 datasets to assess performance across tasks such as classification, clustering, retrieval and more. This open-source benchmark includes test datasets in over 112 languages, offering a comprehensive overview *of text embedding performance across a diverse range of tasks and languages.

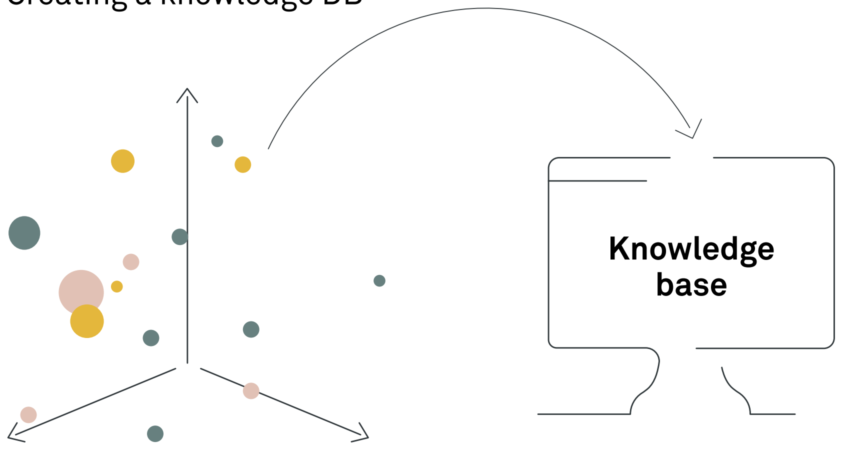
- Retrieval benchmark: Retrieval, one of the eight types of tests in MTEB, evaluates the embeddings' ability to retrieve relevant content in asymmetric search scenarios – specifically, matching short queries with longer texts. This benchmark is particularly crucial for building RAG systems.

- Lack of multilingual retrieval benchmark: As of early 2024, MTEB's retrieval benchmarks only include datasets in English, Polish and Chinese. For languages lacking specific retrieval benchmarks, such as Norwegian, German or Swedish, the choice may involve selecting a top-performing model from classification benchmarks or opting for a versatile multilingual model with strong retrieval capabilities in English.

Embedding is a key process in organising your documents in a vector database, and it might require further customisation to meet your specific needs. Thankfully, with large language models (LLMs), it is now possible to generate training data from your existing content and fine-tune an embedding model to enhance performance significantly. This approach allows for a tailored solution that can improve retrieval accuracy by 5-10%, ensuring your RAG system efficiently connects queries to the correct information.

5. Creating the knowledge base: vector management

Creating a knowledge DB



Creating the knowledge base for a RAG system typically involves indexing the data in a vector database (DB), where the embeddings created in the previous step are stored.

Vector DBs are specifically designed to efficiently manage and retrieve vector embeddings based on semantic meaning rather than mere keyword matches. This capability is what empowers a RAG system to search through vast amounts of embeddings and retrieve the most pertinent pieces of information in response to user queries. It should be noted that although not as common, the information backbone could be another data structure and DB technology; it is not unusual to combine semantic vector search with a knowledge graph or existing relational databases for more complex organisations.

An essential step is devising a strategy for the ongoing data management of the newly created knowledge base. This step is crucial for ensuring that the system's information remains up to date and synchronised with the broader database infrastructure of your organisation.

Vector databases, such as Qdrant, Milvus and Chroma DB, are specialised storage solutions designed to handle these numerical representations. These platforms offer the advantage of being optimised for the quick retrieval of complex data patterns. In practice, though, the driving force when it comes to selecting the right vector DB comes from the need to integrate the storage with the organisation's existing databases, which are often relational systems such as SQL databases.

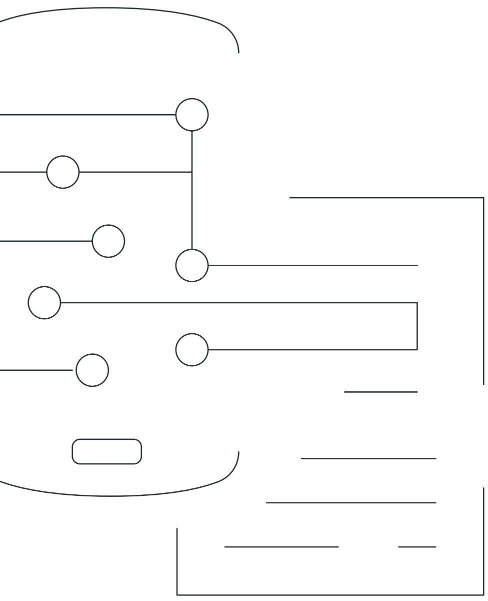
When you create a process for the integration between the storage and the organisation's existing databases, it involves setting up automated workflows or protocols that regularly update the RAG knowledge base with new data entries, modifications or deletions occurring in the company's databases. This ensures that the knowledge base remains an accurate and up-to-date reflection of the organisation's information landscape. ***By focusing on the synchronisation and management of data between the vector database and existing databases, companies can leverage their accumulated knowledge more effectively.***

This alignment is not just about maintaining data integrity; it is about enhancing the accessibility and utility of information across the organisation. Again, as part of setting up these data flows, it is absolutely key that you preserve access restrictions to sensitive data in order not to leak information as part of generating answers to end users of the system that they were not intended to see.

Having a strategic approach to data management underpins the success of the RAG system, ensuring it becomes a powerful tool in the organisation's information ecosystem and that it stays relevant over time.

The generative phase

Once the indexing phase has ensured that all relevant data is stored in the knowledge base, we move on to the part of the system responsible for handling user inputs and generating answers to a given query.



6. Information retrieval: finding the right data

In the generative phase of a retrieval augmented generation (RAG) system, the core function that determines the quality of the response is the retrieval of relevant information from a user's query. This process typically begins with semantic search, which leverages the vector embeddings crafted during the indexing phase.

The user's question (query) is transformed into a vector using the same embedding model that processed the knowledge base's text, ensuring the query and the stored information share a common vector space. This alignment enables a semantic similarity search, aiming to find content that closely aligns with the user's question in meaning, not just in shared keywords.

In its simplest form, the retrieval step consists of a top-k vector retrieval that may return the 3-10 most similar matches to the search query. In an ideal scenario, the system would retrieve a single passage that precisely answers the query. Yet due to the often overlapping and contextually rich nature of information, it is required to sample multiple text chunks to provide fulfilling context for the large language model (LLM) to answer the question.

The precision of this retrieval process is critical. If the system retrieves passages that are irrelevant or wrong in the context related to the query, the generated response risks being false. Where RAG systems today typically come short is not by the inherent capability of the LLM to generate a meaningful response but in the database-and-retrieval design that occasionally provides contextually wrong data if not designed properly.

From our project experience, almost 9 out of 10 "wrong" or "low-quality" answers are caused not by the lacking abilities of the LLM itself but instead by the vector database not being able to return the right context for the LLM to generate its answer on top of. This also often points towards poor data management processes, e.g. by having duplicated documents describing the same standard operating procedure or other company information in different ways (where at least one is typically an outdated version of the document).

It is often necessary to implement advanced retrieval strategies, such as reranking and filtering, to refine the search results. Reranking involves adjusting the initial search results based on additional criteria or algorithms to ensure that the most relevant passages are prioritised from a larger pool. Filtering further refines these results by removing any information that does not meet certain predefined standards or relevance thresholds, either with rule-based systems or a layer of LLM prompts.

However, there is only so much that can be done at the retrieval stage. If the knowledge base itself has simply been created by dumping all company data into the vector database without further consideration, the performance of the RAG system will likely never be satisfactory for a production deployment. Therefore, the construction of the knowledge base requires careful planning to organise the existing data in a way that enhances the system's ability to discern and retrieve the precise information needed to answer user queries effectively.

7. Generating an answer

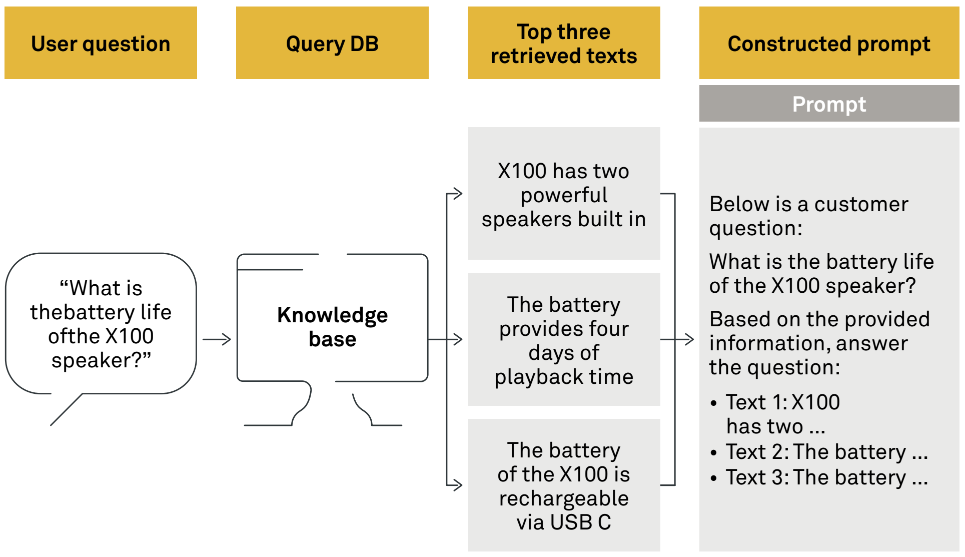
You have now finally reached the point where you generate an answer using a large language model (LLM). This part of the process uses generative AI to turn the information you have retrieved into a clear and relevant response to the user's question.

It is a simple yet effective approach.

After retrieving the text passages that closely match the user's query, the system utilises an LLM to generate an answer. This is done by sending the LLM a text instruction, called a prompt, which includes the user's question along with the selected text snippets, instructing the model to base its response on this specific context.

An example of a simple prompt and generated answer might be:

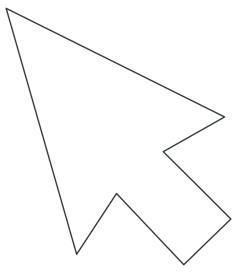
Retrieval and generation


User question
Query DB
Top three retrieved texts
Constructed prompt
Prompt
"What is thebattery life ofthe X100 speaker?"
Knowledge base
X100 has two powerful speakers built in
The battery provides four days of playback time
The battery of the X100 is rechargeable via USB C
Below is a customer question:
What is the battery life of the X100 speaker?
Based on the provided information, answer the question:
• Text 1: X100 has two ...
• Text 2: The battery ...
• Text 3: The battery ...

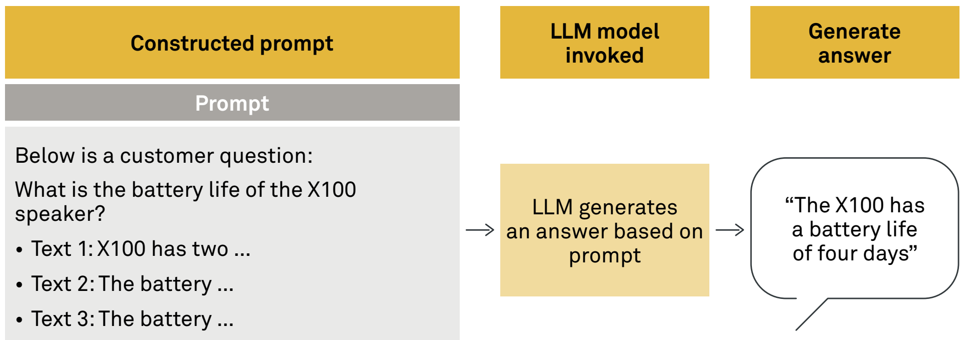
Large language models at a glance

Briefly looking at the LLM, these models are sophisticated statistical models that predict subsequent words based on vast amounts of training data. The rapid advancement in LLM capabilities ensures that the state-of-the-art models are more than equipped to handle both customer-facing roles or internal applications in a RAG system, delivering high-quality outputs that meet the demands of real-world applications.

Following the process of the retrieval step, the process can be visualised as below:



Retrieval and generation


Constructed prompt
LLM model invoked
Generate answer
Prompt
Below is a customer question:
What is the battery life of the X100 speaker?
• Text 1: X100 has two ...
• Text 2: The battery ...
• Text 3: The battery ...
LLM generates an answer based on prompt
"The X100 has a battery life of four days"

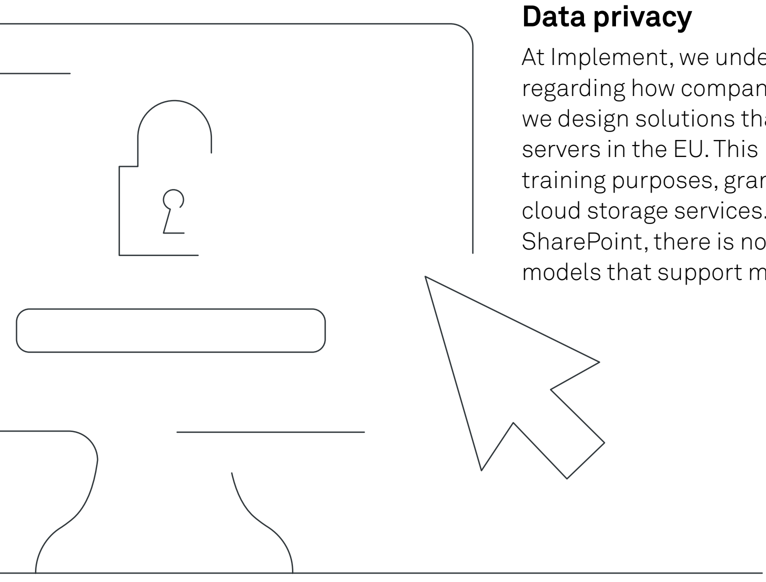
The reliability of responses generated by large language models (LLMs), such as those in a chat interface like ChatGPT, significantly underperforms when compared to a retrieval augmented generation (RAG) system that is grounded in contextual data. While simple chat interfaces can sometimes produce factually wrong answers, known as hallucinations, ***a RAG system significantly reduces the risk of factually wrong answers by anchoring responses in contextually relevant data snippets.*** This grounding mechanism sets RAG systems apart, making them particularly suited for business environments where the credibility of information is paramount.

Hybrid RAG systems for exact answers

It is important to note that in a RAG system, the exact wording of the generated response cannot be controlled and is determined by the LLM's generative function, which can pose challenges in situations that require responses with absolute precision. To address this, integrating RAG systems with conventional NLU chatbot technologies, which utilise predefined responses for specific queries, can offer a comprehensive solution. This hybrid model combines the depth and adaptability of generative AI with the reliability of scripted responses, catering to the diverse needs of businesses seeking to maintain the highest levels of answer accuracy and trustworthiness, such as clients in industries such as finance, insurance and healthcare.

Data privacy

At Implement, we understand that data privacy remains a critical concern, especially regarding how company data might be used for model training. Addressing this, we design solutions that host the generative LLM models, such as GPT4, on secure servers in the EU. This IT infrastructure ensures that no data is utilised for model training purposes, granting businesses full control over their information, akin to other cloud storage services. For organisations that already utilise cloud storage, such as SharePoint, there is no hindrance to utilising generative AI and RAG using commercial models that support more than 100 languages fluently.



If, for security reasons, working with data in the cloud is not an option for your organisation, open-source models like Mistral MoE can be hosted on-premises, ensuring that data never leaves your closed network. State-of-the-art open-source models perform almost on par with models such as GPT4 in the context of RAG systems but only do so in English.

As of early 2024, there is a significant gap in the performance of non-English languages between closed-source models, like GPT4, and open-source models. We expect that multilingual open-source models with performance sufficient for customer-facing on-premises RAG solutions will emerge during 2024.

8. Filtering and guardrails

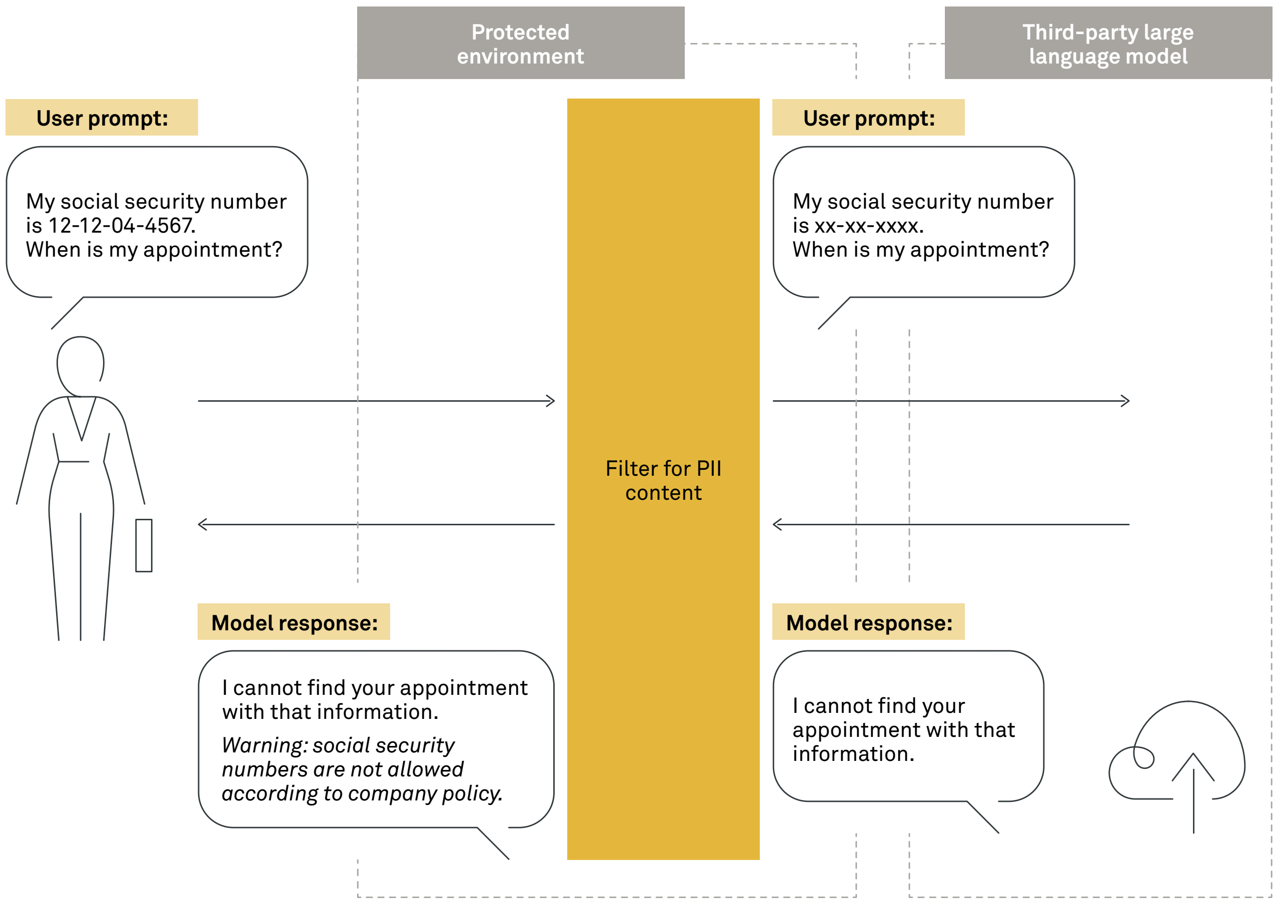
Following the generation of an answer in a retrieval augmented generation (RAG) system, one last step remains before presenting the response to the user: filtering and applying guardrails.

This final review process ensures that the generated answer not only meets the standards expected in a professional setting but also aligns with company guidelines and is free from any harmful content. The depth of filtering required can vary significantly based on the application’s context. In some cases, minimal filtering may suffice, especially in controlled environments and for tools designed for internal purposes. However, in more open-ended applications, extensive review mechanisms are essential to maintain the integrity and reliability of the responses provided.

The review often involves additional calls to LLM models, designed to assess the appropriateness of the generated answer in the given context. These models can evaluate whether the response adheres to predefined content guidelines, ensuring it is free from bias, inaccuracies or any form of sensitive or inappropriate content that could harm the user or tarnish the company's reputation.

In addition, you can also apply content filtering at the initial stage of user query input. This preemptive measure helps to screen out any inappropriate, sensitive or harmful queries from the outset, further safeguarding the system from generating unsuitable responses. Examples of such filtering include identifying and blocking profanity, discriminatory language or requests for information that violates privacy regulations.

Guardrail filtering


Protected environment
Third-party large language model
User prompt:
My social security number is 12-12-04-4567. When is my appointment?
User prompt:
My social security number is xx-xx-xxxx. When is my appointment?
Filter for PII content
Model response:
I cannot find your appointment with that information. *Warning: social security numbers are not allowed according to company policy.*
Model response:
I cannot find your appointment with that information.

Implementing these filtering and guardrail mechanisms is not just about preserving the factual accuracy of the answers. It is also about ensuring that every interaction reflects the organisation's values, adheres to legal and ethical standards and contributes to a safe and respectful user experience.

Through careful review and refinement of both the questions and the generated answers, businesses can leverage the transformative potential of RAG systems while maintaining the highest levels of trust and integrity in their digital communications.

Now let's get started!

Thank you for following along with our overview of retrieval augmented generation (RAG) systems. We hope this guide has shed light on the intricate workings of RAGs and demonstrated their potential to revolutionise information retrieval and response generation in various settings.

At Implement Consulting Group, we pride ourselves on our extensive experience in bringing RAG systems to life in production environments. Our expertise spans assessing organisational needs to deploying robust, high-performing RAG solutions tailored to your specific requirements.

If you are considering the integration of a RAG system in your company's operations and need professional guidance to ensure success, we are here to assist, helping you navigate this exciting technology landscape and unlock the full potential of your organisation's collective knowledge in new ways using retrieval augmented generation (RAG).

References:

- Evaluating the Ideal Chunk Size for RAG System using Llamaindex
- Building RAG-based LLM Applications for Production
- How to chunk text a comparative analysis
- Massive Text Embedding Benchmark (MTEB)
- SBERT on Asymmetric retrieval
- Finetuning embeddings using LlamaIndex
- Finetuning embeddings using SBERTs GenQ approach
- All You Need to Know About Vector Databases and How to Use Them to Augment Your LLM Apps
- 10 ways to improve the performance of retrieval augmented generation systems
- Improving Retrieval performance in RAG pipelines with Hybrid Search
- Evaluating retrieval performance of RAG systems using LlamaIndex

Fast facts about Implement

Founded: 1996
Number of employees: 1,500+
Headquarters: Copenhagen
Offices: Stockholm, Malmo, Gothenburg, Oslo, Zurich, Munich, Hamburg, Aarhus, Düsseldorf and Raleigh
implementconsultinggroup.com