# D-RAG: making the RAG interact with the densely packed data

The idea of RAG  was essentially to provide the way to fight with the hallucinations in an LLM generation flow by providing a real set of documents as a supplement to the generating-process. The RAG itself roughly consists of 4 big steps: (1) Forming a dataset from the available data, (2) Getting it vectorised and packed into some kind of a queriable system, (3) Retrieve the relevant data, judging by the (4) conversation-interaction with user, adapting to the they needs. We first clean the dataset, ranking the websites, raising the most important once to our advanced feature extraction pipeline. Then we create a hierarchical topology of the pages, moving the important features up to the website-level in contrast to the page-level. In the embedding creation step we use a vectorised DB, with additional features concatenated on the website- and page-level. After embedding creation we also create an external knowledge base, tailored to our customer person. During the RAG phase, we, in addition to do the similarity ranking based on the vectorised documents, form a query based on additional ontological information provided by the KB. Finally, during the conversational phase, we employ multi-chain strategies of speaking with the client, switching adaptively from query retrieval to specification Q&A etc.

## Initial idea

The biggest intuition in our problem was to outline the most important things for the user persona (i.e. SCD)  to rely on in the RAG process. Those 6 entities are actually listed in the design document, namely: technology, services, materials, products, industries and regions. We thus try to leverage all the available data on all the essential steps of the RAG agent process. Additional thing was trying to make our data more responsive to the queries by its metadata enrichment — we would like to design several channels of information communication between the conversational and the RAG system and the data.

## Data processing

The initial data is a raw text with some topological properties provided by the website inner structure (which could be possibly applied to the problem). Main features of the dataset include quite uneven length of the data, inclusion of the technical files of the webpages, rich content of the urls, etc. At first we've been trying to process the full dataset, but understood that it is infeasible for most of the components of the RAG — thus we decided to make the dataset smaller (i.e. increase its relevance and density). The other thing was the topology of the data — we would like to use the fact that there is a given topology in the data, and not only one source of it is available. The cleaning stage of the preprocessing is self-explanatory — we'd like to remove everything which is not textual, removing the near duplicates, and non-important page parts. All the aforementioned is making our data denser, thus having more ways to be ranked and retrieved by our RAG.

### Making the dataset smaller

The other important step, forced by the limited resources (practical) and extreme volume of the data (theoretical) was the near duplicate removal, made possible by the LSH. However, due to the chaotic nature of the documents, that can be a problematic approach. An LSH initially has shown good performance, but not a drastic improvement in a dataset reduction size. Thus an alternative solution was developed: URL-based filtering. We extracted a set of features from each url in an umbrella of URLs for each website, and then labeled a small subset of those – using a simple TF-IDF vectoriser, a set of numerical features we then used a logistic regression for that. Thus we got a reduction of 80% size of a dataset (also adding 20% of the data as a random sample from the rest).

## Densifying the data

One of the important ideas was to hierarchialise the feature engineering step, namely dividing the documents into the buckets by the top domain name, and then, by analysing the inner web pages in the each of the buckets, move their features up to the domain. By doing this, we believe that we diffuse the features, which are hidden inside the subpages of a domain through the topology of a website. For the specific features, we use the intuition that the locational-spatial information is the key for most of the supply chain analysis. We designed a pipeline for identifying the exact location names and associating them with the specific pages.

## Spatial location extraction

A combination of US cities and Geonames dataset was used to append additional spatial information to the initial text files, provided by ODS data collection platform. We initially identify the list of city candidates in the text (no NER yet), providing a list of cities associated with the page. Then we evaluate a list of the cities and match them with the candidates, merging a set of features as well; then performing a unification of the features under the top-page level. One of the key things here is geospatial coordinates, which create an accurate geographical representation of the company (top level).

## NER

An important information provider for the important query matching information is surely the named entities (preferably also with a conference resolution module installed): thus we instantaniated two prototypes of NER modules. First one is based on GLiner — near SotA performance and  the open set of NER descriptors (you can even use IOF ontology entities for that!). The big cons of this model is its speed limitation (so even in a large scale production system it may be unusable, however the model has rather small 0.5B parameters for the top one). Thus we developed another pipeline on a less expensive model, the spacy nlp pipeline. It has a bit restricted NER pattern, which has a closed set of entities — however they still provide helpful metadata to the pages, helping to connect those with the KB.

# RAG

Our RAG is based on the combination of the two parallel approaches: KB-Query enrichment and the Vectorised DB query matching. For the first process, by the IOF supply chain ontology file,  we construct a knowledge base. This knowledge base then helps us to form a set of query terms by the user request. The ontology is created using an owlready2, acting as a query ready knowledge base, which adds additional explanation to the user request. All the KB services  are hidden behind a FastAPI server. The query first is decomposed to its KB relationships; then the enriched request is passed to the vectorised db.
For the second db matching, first wecompose a big database by vectorising the provided documents with a sentences based chunking embedding. We then build a relevance query-to-document scoring (based on the vectorised content and metadata), and then retrieve top-k documents to the query to help the conversational agent.

### In-house ontology

We propose an idea to create a self-evolving knowledge base system, by using a combination of YARRRML and Morpg-KGC. One can generate the ontological rules for the relational database, containing the metadata we extracted; then a possibility of an in-house knowledge base (now about companies!) becomes real.

## Conversational Unit

The conversational system is based on the Langchain q&a methods. The chat is separated into several possible chains to follow, judging by the user's needs. Many different user flows are possible, however usually the system is expected to give a competent answer during the first conversation rounds. The chat history is attached and maintained throughout the different chains of conversation. The Vue based user interface for a smooth experience is provided as a frontend. Also, there is a possible extension of the system by a small external knowledge base of Supply Chain related academic and standardisation documents that could be attached – processed similarly to the web pages – (however there are potentional copyright risks, thus it was not included in the final pipeline).

## Conclusion

During the D-RAG development a complex processing system was constructed, which tries to make the chaotic web data be more “dense” and “rich”. We successfully employed several processing techniques to make the RAG system more accurate and powerful. We developed a vectorisation strategy which would leverage both the amount of the meaningful text we use, and the additional metadata we mine from the text. We proposed a chain of retrieval Q&A based user interaction system, which would guide the Supply Chain Specialist through the rich documentation available in our system. Finally, we made the system easily extensible and deployable, either as a set of isolated services or, in some cases, as a processing pipeline stages.