Using AI Models to Generate Knowledge Bases in the Context of IT Service Management

Rim Messaoudi[a], Alexandru Alexandrescu and Francois Azelart
Akkodis Research, Akkodis, France

Keywords: Generative AI, Large Language Models (LLM), Knowledge Base (KB), IT Service Management.

Abstract: Artificial intelligence (AI) models have fundamentally improved its capacity to treat several topics and domains. The purpose of this paper is to use these methods especially Large Language Models (LLMs) to generate knowledge databases in the context of Information Technology Service Management (ITSM). The work starts by extracting information from documents and perform analysis on these documents using Natural Language Processing (NLP) techniques to group them into rigorous knowledge base articles that are easy to classify and search. This paper presents a method that applies generative AI for building a knowledge database in the context of ITSM (Information Technology Service and Management).

1 INTRODUCTION

A knowledge base (KB) is defined as the underlying set of facts and actions which a computer system has available to solve a problem. It can be written as a document that provides information, guidance, or solutions on a specific topic, issue, or question (Wang, Q., 2017). A ticket is a record of a specific issue, request, or inquiry submitted by a user or customer to a support team. It typically includes details about the problem, the user's contact information, and any relevant context (Li, 2014). AI models are included to manage this type of problems, the goal is a achieve a performant resolution. A description of specific AI tasks can be found in this site.[2] In this paper, we try to present a use case that use LLM models to generate a resolution for several problems related to the ITSM topic (Mo, Y.,2024).

2 PROPOSED METHOD DESCRIPTION

One way to approach this project is to index the information from the documents and perform a Question Answering task to extract the text that corresponds to a specific label. The labels are usually organized into a template for all corresponding documents. For example, we can extract the document type, the objective, or the detailed procedure, etc., into specific tickets.

Next, a text summarization task can be used to extract a short description. Once each document is arranged in articles based on a template, a classification task can be performed on every article to group similar tickets into knowledge base articles.

2.1 Applied Data

We use in this project real data, that are extracted from Service now[3] and GLPI[4] tools. They presented textual descriptions related to incident tickets. The applied data contain also documents with different format. These documents contain procedures that guide administrators and agents through the resolution process.

2.2 Architecture

The project architecture, presented in Figure 1, contains two parts the frontend part and the back-end part: the main frontend holds the Streamlit tool and calls the functions from the backend that aggregates all the processors. All the developed files contain only

[a] https://orcid.org/0000-0001-5311-0375
[2] https://huggingface.co/task
[3] https://www.servicenow.com/fr/
[4] https://glpi-project.org/fr/

Messaoudi, R., Alexandrescu, A. and Azelart, F.
Using AI Models to Generate Knowledge Bases in the Context of IT Service Management.
DOI: 10.5220/0013741600004000

In *Proceedings of the 17th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2025) - Volume 2: KEOD and KMIS*, pages 136-139
ISBN: 978-989-758-769-6; ISSN: 2184-3228

functions. The config.yaml file holds the project configuration, such as the models and paths used.

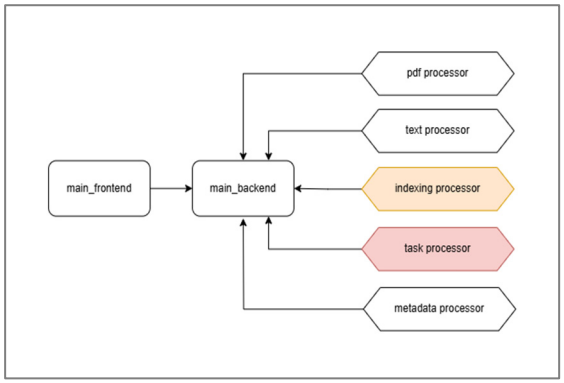

Figure 1: Proposed method architecture.

2.3 Method Steps Description

The following steps are used to read PDF documents that contains resolution notes and procedures used to resolve problems or tickets. Once the PDF processing is approved, other document types can follow the same steps:

1. Read PDF documents, and split them into text chunks, and store these document text chunks, indexes, embeddings, and metadata. Since free and downloadable models like the ones available on the site of Hugginface [5] are limited to 512 tokens, we cannot process all the text from one document at the same time. For example, an average document contains about 16,000 characters, which averages 2,000-3,000 tokens. So, the text must be split into chunks. The splitting of the text must be done by preserving full sentences, and for that, an NLP (Natural Language Processing) toolkit like SpaCy, pysbd, NLTK and other packages, specialized for French language, is necessary such as:
   - Mistral-7B can handle 8k tokens but needs a powerful GPU like A100 with 40 GB of VRAM (Siino, M, 2024);
   - Mixtral can handle 32k tokens, but it is not downloadable and not free (Lermen, S, 2023);

2. Using Transformers:

Use a Sentence Transformer model to index each PDF document. The choice of a sentence transformer and not a usual transformer is crucial since we are interested in the meaning of each sentence in context and not a word or group of words. Some sentence transformers for French extracted from Hugging Face include:

a) Lajavaness sentence-flaubert-base ,
b) Lajavaness/sentence-camembert-large ,
c) paraphrase-multilingual-mpnet-base-v2 ,
d) paraphrase-multilingual-MiniLM-L12-v2
e) all-MiniLM-L6-v2

For the Question Answering task, we use: FAISS index search to search sentences that correspond to a question. We use also cosine similarity to find the text chunk indexes that correspond to a question, we extract the top k sentences, and perform a re-ranking of these sentences using a transformer specialized in QA to extract the response. We use the cloud model openai to perform indexing and question-answering.

3. Based on the tasks described above, we move to the process the generated PDF document and retrieve data according to a specific template.
4. Find a description for one document by performing successive summarization tasks using a specialized transformer model.
5. Perform classification on each description to find similar topics. For example: (a) Softmax and feedforward from Torch; or (b) FlauBERT.
6. Use the same process for QA as above, but this time store the position of each image relative to the text from each PDF document. When constructing the KB based on the template, insert the images at the same positions relative to the text.
7. Develop a streamlit tools that contain three tabs: (a) Indexing documents; (b) Testing different models and techniques; and (c) Batch for batch processing a complete folder;

2.3.1 Text Splitting Task

The purpose of this phase is to read the text from a PDF file, clean it by removing extra whitespaces, recurrent dots, etc., and then split it into full sentences using an NLP toolkit like SpaCy (Kumar, M.,2023), pysbd, NLTK (Yao, J. (2019)). Then the sentences can be grouped into chunks based on the total number of characters or total number of words. The latter requires some extra processing using NLPs for splitting the text into words and counting them. The result of this operation is a list of text chunks.

2.3.2 Indexing Task

The purpose of this task is to retrieve the embeddings for each chunk and save them as FAISS indexes,

[5] https://huggingface.co/

NumPy embeddings, and each chunk list. This step is basically retrieving the embeddings, but since some of the operations involved take time, saving them is necessary to avoid redundancy.

Some metadata is also saved, such as the file name, the models, and the options used. The next time the same file is selected, if it is found in the metadata with the same parameters, the operation is avoided since its embeddings are already saved. However, if even one parameter is different, a new operation will start with the new parameters.

It is very important to save the embeddings and the sentences chunks in the same order. At the ranking phase, we will retrieve the lost relevant indices from the chunks list so the embeddings and the chunks list must be aligned.

2.3.3 Question Answering Task

For this task, the first step is to retrieve the top-K most relevant sentences from FAISS vectors or using cosine similarity between the question and the chunk embeddings. This step is an optimization to avoid performing Question Answering on the whole text. Now that we have a reduced set of top-K sentences, we'll use the QA model to score these sentences based on how well they answer the question.

2.3.4 Text Classification

For this task use text classification techniques to group the sentences to groups based on the same label like for ex: ["description", "procÃ©dure", "rÃ©sultat", "sans rapport"]. Use a fine-tuned transformer for text classification or zero shot classification[6].

The problem here is defined as a lot of noise between the sentences due to date, names, page numbers and other metadata. A solution is needed here to eliminate noise and group sentences together.

2.3.5 Group Sentences Dynamically Based on the Similarity Factor

This phase aims to group extracted sentences based on similarity factor:

1. Consider a group of K_max consecutive sentences, K_max < N (under 40). Calculate the pairwise cosine similarity matrix for k_max sentences. Tip: Use sklearn [7]from scipy for CPU, and pytorch if a GPU is available.
2. Take a subgroup of minimum K_min and maximum K_max sentences using a dynamic size sliding window and do some operation on the corresponding minor from the similarity matrix to calculate a similarity score.
3. Extract the sentence group with maximum score, preserving the sentence order.
4. Repeat the process until all sentences are grouped.

2.3.6 Observation

The similarity matrix is a symmetrical matrix having the same scores in the upper left and the lower right and 1 on the diagonals since every sentence is perfectly similar to itself.

Calculating similarity score: Use the similarity matrix EIGENVALUES. Calculating the eigenvectors of the similarity matrix gives insight into the dominant semantic direction and the dominant eigenvalue will define the dominant direction (similarity).

2.4 Results: UI Description

The indexing page aggregates all the text preprocessing, embeddings generation and data saving like faiss vectors, numpy embeddings, text chunks and metadata. We can choose a model for computing embeddings, for text preprocessing using NLP, the level between words and characters and the chunk size is expressed in units of words or characters depending on the level chosen.

2.4.1 Developed Tools

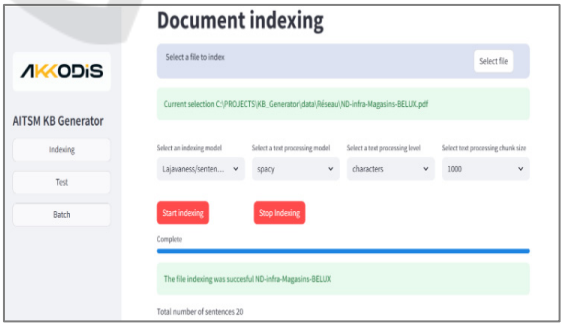

Figure 2: Document indexing interface.

The tests page is used to perform tests using the last embeddings calculated in the indexing page. We can

[6] https://huggingface.co/tasks/zero-shot-classification

[7] https://scikitlearn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html

choose between a JSON template or simply putting a question related to the currently embedded document.

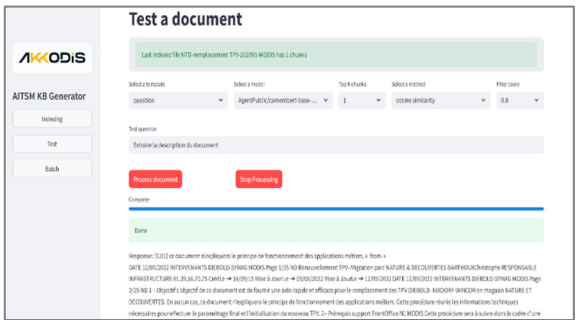

Figure 3: Testing a document interface.

The batch page will be used to batch process a folder of documents based on the options, models and tools from the previous pages.

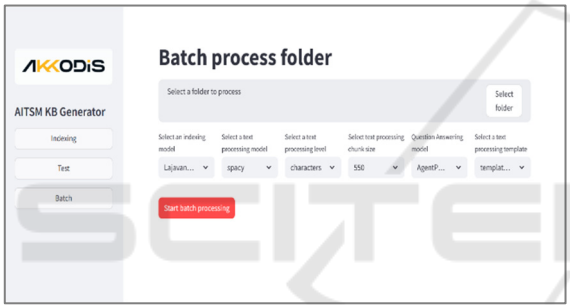

Figure 4: Batch process floder interface.

3 CONCLUSIONS

We present in this paper the use case that discusses the subject of how to generate a knowledge base by using AI models. The purpose of this paper is to use these methods especially Large Language Models (LLMs) to generate knowledge databases in the context of Information Technology Service Management (ITSM). The proposed method starts by extracting information and procedures from documents and try to perform analysis on these documents using several AI techniques. In our future work, we will continue to test on other databases, and we will move to automatise the process of ticket resolution.

ACKNOWLEDGEMENTS

This work is a part of our research project named AITSM (Artificial Intelligence applied on IT Service Management) realised in the Akkodis Research company.

REFERENCES

Kumar, M., Chaturvedi, K. K., Sharma, A., Arora, A., Farooqi, M. S., Lal, S. B., ... & Ranjan, R. (2023). An algorithm for automatic text annotation for named entity recognition using Spacy framework

Lermen, S., Rogers-Smith, C., & Ladish, J. (2023). Lora fine-tuning efficiently undoes safety training in llama 2-chat 70b. arXiv preprint arXiv:2310.20624.

Li, T. H., Liu, R., Sukaviriya, N., Li, Y., Yang, J., Sandin, M., & Lee, J. (2014, June). Incident ticket analytics for it application management services. In 2014 IEEE International Conference on Services Computing (pp. 568-574). IEEE.

Mo, Y., Qin, H., Dong, Y., Zhu, Z., & Li, Z. (2024). Large language model (llm) ai text generation detection based on transformer deep learning algorithm. arXiv preprint arXiv:2405.06652.

Siino, M., & Tinnirello, I. (2024). Prompt engineering for identifying sexism using GPT mistral 7B. Working Notes of CLEF.

Wang, Q., Zhou, W., Zeng, C., Li, T., Shwartz, L., & Grabarnik, G. Y. (2017, June). Constructing the knowledge base for cognitive it service management. In 2017 IEEE International Conference on Services Computing (SCC) (pp. 410-417). IEEE.

Yao, J. (2019, April). Automated sentiment analysis of text data with NLTK. In Journal of physics: conference series (Vol. 1187, No. 5, p. 052020). IOP Publishing.