Preproceedings of the 19th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 107–117

Benchmarking OpenAI's APIs and other Large Language Models for Repeatable and Efficient Question Answering Across Multiple Documents

Elena Filipovska[†], Ana Mladenovska[*†], Merxhan Bajrami[*†], Jovana Dobreva[*†], Velislava Hillman[‡], Petre Lameski[*†], Eftim Zdravevski[*†]
[*]Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University in Skopje, Macedonia
[†]Magix.AI, Skopje
[‡]Department of Media and Communications, London School of Economics and Political Science, London, UK
{elena, ana, jovana, merxhan.bajrami, petre, eftim}@magix.ai, {ana.mladenovska.1, bajrami.merdzan}@students.finki.ukim.mk, {jovana.dobreva, petre.lameski, eftim.zdravevski}@finki.ukim.mk, v.hillman@lse.ac.uk

***Abstract*—The rapid growth of document volumes and complexity in various domains necessitates advanced automated methods to enhance the efficiency and accuracy of information extraction and analysis. This paper aims to evaluate the efficiency and repeatability of OpenAI's APIs and other Large Language Models (LLMs) in automating question-answering tasks across multiple documents, specifically focusing on analyzing Data Privacy Policy (DPP) documents of selected EdTech providers. We test how well these models perform on large-scale text processing tasks using the OpenAI's LLM models (GPT 3.5 Turbo, GPT 4, GPT 4o) and APIs in several frameworks: direct API calls (i.e., one-shot learning), LangChain, and Retrieval Augmented Generation (RAG) systems. We also evaluate a local deployment of quantized versions (with FAISS) of LLM models (Llama-2-13B-chat-GPTQ). Through systematic evaluation against predefined use cases and a range of metrics, including response format, execution time, and cost, our study aims to provide insights into the optimal practices for document analysis. Our findings demonstrate that using OpenAI's LLMs via API calls is a workable workaround for accelerating document analysis when using a local GPU-powered infrastructure is not a viable solution, particularly for long texts. On the other hand, the local deployment is quite valuable for maintaining the data within the private infrastructure. Our findings show that the quantized models retain substantial relevance even with fewer parameters than ChatGPT and do not impose processing restrictions on the number of tokens. This study offers insights on maximizing the use of LLMs for better efficiency and data governance in addition to confirming their usefulness in improving document analysis procedures.**

***Index Terms*—OpenAI, LangChain, RAG, GPT, QA, LLM, Llama, Large Language Models, Multi-document, one-shot learning, few-shot learning Q&A**

I. Introduction

TRADITIONAL document analysis methods often rely on manual review or simplistic keyword-based searches, leading to significant inefficiencies and limitations. As the volume and diversity of documents continue to expand, the need for innovative approaches that can streamline analysis while preserving accuracy and comprehensiveness arises. In this study, we explore different methods for assisting the analysis of selected EdTech providers' Data Privacy Policy (DPP) documents. On the task at hand, we aim to evaluate the efficacy, consistency, benefits and limitations of various LLMs in assessing DPP documents. The importance for such an assessment is founded on the need for an automated, scalable and reliable way to systematically analyze large bodies of text semantically. We also aim to examine the most optimal way of using LLMs regarding the optimizing factor - whether it is the price, the duration or the format of the answers provided that plays a key role in a technical chore. All of these LLM models are trained on extensive datasets and exhibit remarkable proficiency in generating human-like responses and cognitive reasoning across diverse tasks. LLMs are built to handle various tasks, such as text generation, translation, content summary, chatbot conversations, and more [1].

Our methodology begins with the gathering of documents from a vast repository of online accessible terms of services of popular educational platforms and products. The experimental flow combining expert knowledge and automated processing is shown in Fig. 1. The first step of selecting the EdTech platforms is performed by educational experts. Likewise, the legal team defined 45 questions about key aspects of the GDPR (General Data Protection Regulation). Even though the legal expertise is crucial for other aspects of the methodology, the focus of this article is the automated processing of the documents and questions by LLM models, as shown in the top-right rectangle. The whole methodology is part of a larger study that involves manual expert validation of the AI-generated answers and evaluation of their quality.

Through a scraping and filtering process using the BeautifulSoup library, we refine this collection of edtech products, ultimately obtaining a list of about 800 DPP documents for further processing. These documents are evaluated to ensure their content aligns with our study's focus. For each DPP document, these GDPR-related questions are asked through various LLM methods, aiming to evaluate whether the documents address regulatory requirements.

Topical area: Advanced Artificial Intelligence in Applications

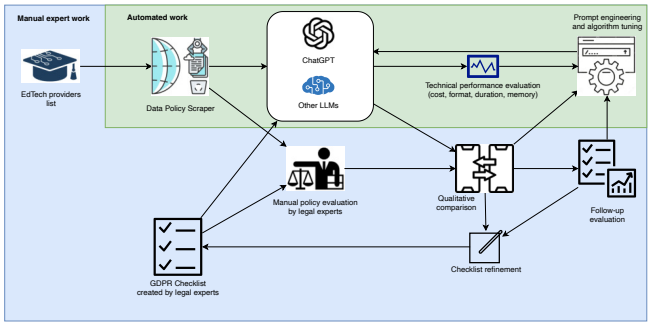

Fig. 1. Methodology combining expert knowledge and automated processing

Furthermore, we explore four distinct approaches for leveraging LLMs to analyze these documents and answer the 45 GDPR-related questions. Specifically, we explore the capabilities of OpenAI's ChatGPT (Generative Pre-trained Transformers) GPT 3.5 Turbo, GPT 4, and GPT 4o. To use them, we evaluate OpenAI's direct APIs and Microsoft Azure's APIs through direct API calls, LangChain, and Retrieval Augmented Generation (RAG) systems. We also evaluate a local deployment of quantized versions (with FAISS - Facebook AI Similarity Search) of LLM models (Llama-2-13B-chat-GPTQ). Each approach offers unique advantages in processing and interpreting documents. All of these methods are systematically evaluated on multiple predefined use cases that differ in the number of documents being processed at once, the number of questions being evaluated at once, the length of the request, the length and format of the response, etc. To score the approaches, we employ a range of metrics, including response format, execution time, and cost.

Our study aims to provide insights into the optimal practices for document analysis using Artificial Intelligence (AI) technologies. By focusing on the documents themselves, we aim to uncover nuanced insights that inform decision-making and optimize outcomes in document governance and compliance. That being said, the central point is on testing AI technology for scaling otherwise manual and highly nuanced specialized literature that typically takes a long time to assess, rather than encouraging or suggesting that AI technology alone can justify if documents present companies' compliance, transparency, or accountability to any rules or conditions in which they are mandated to operate.

This paper is structured as follows. After this introduction, Section II reviews related works, providing context and highlighting the contributions of other researchers in the field. Section III presents the methodology of the paper. Specifically, in subsection III-A we describe the data used in this study, detailing the selection and preparation processes. In subsections III-B and III-C, we outline the specific use cases designed to test the efficacy of different approaches and the evaluation methods, respectively. Subsection III-D discusses the use and implementation of OpenAI models, elaborating on each approach's technical aspects, and Subsection III-E describes the implementation of the quantized Llama-2-13B-chat model with FAISS vector database. Section IV presents and discusses the results, offering a detailed analysis of the findings, explains the challenges faced and the limitations of the current study. Finally, Section V concludes the paper by summarizing the key findings and suggesting directions for future research.

II. Related work

Due to the widespread use of LLMs in general domains and real world applications, various text analysis tasks have been automated and optimized, even for relatively specific tasks. Given the extensive training and massive datasets that serve as the basis for the models offered by OpenAI, their usage covers a wide set of areas. The utilization process undergoes several techniques used across almost all applications. Namely, the reading comprehension abilities of LLMs are one of their primary strengths. Ranging from classifying titles and abstracts against an inclusion/exclusion criteria, as explored in [2], to policy advising in governmental processes as elaborated in [3], the basis of implementing a system containing GPT models is text understanding.

In [4], a system is proposed that aims to address the gap in answering search queries, namely those that lack any previous internal related information about the topic. Their main approach in accomplishing this is to introduce both GPT-3.5 and GPT-4 to novel prompts, as well as instruct the model to provide a step-by-step explanation of the answering process to enhance the responses, reduce the hallucination risks, and ultimately increase the response comprehensiveness.

Authors in [5] present a question-answering chatbot aiming to answer questions related to numerous policies and guidelines set by their organization to dictate the appropriate outfit. Using the National Defence Policies and Standards of Canada as training data to their model, they developed a retrieval-based chatbot – since the responses it produces need to only be related to the aforementioned policies. The encodings of this data are based on the BERT model, so when the question vector is computed, the most similar passage of the entire corpus is retrieved, which represents the base for the answer. This response initiation by the model based on data given as additional information in the prompt is a common practice in modern AI chatbot applications and serves as the ground mechanism for Retrieval Augmented (RAG) Systems. This emerging branch of AI implies heavy reliance on prompt engineering and appropriate frameworks for optimal solutions, as well as a selection of the most relevant model for the problem at hand. Similar to this work, one of the approaches that we evaluate is the use of RAG systems to enhance the LLM responses.

Authors of [6] explore the differences in the abilities and limitations of three GPT models in the case of multiple choice question answering based on three levels of Python courses. The rapid advancements, as can be seen from the obtained results from their experiments, can significantly impact the use of generative models in the educational process. Their qualitative analysis of the results suggests very noticeable

improvements in the latest GPT model releases regarding the validity of the generated responses as well as the manner in which these responses are created.

The performance of both GPT-3.5 Turbo and GPT-4 models concerning a specific medical domain is explored in [7]. Namely, they explore the usage of the GPT models in the standardized orthopedic examination administered by the U.S orthopedic residency programs without prior exposure to similar queries. Their results indicate superior performance of the GPT-4 model over the GPT-3.5 Turbo regarding the quality of the reasoning capabilities produced.

Rapid development of AI branches always introduces security and data privacy concerns. The study [8] explores the evolving security, privacy, and data protection challenges posed by the increasing use of sophisticated digital products and platforms. It discusses how systems like chatbots process and store personal data, the security risks associated with their deployment, and the need for stringent data protection measures in line with regulations like the GDPR. Often, end-users (be that individuals or organisations) rely on data privacy policies and other lengthy texts to be reassured about how digital technologies meet data privacy and security standards and requirements. However, these texts are typically difficult to digest and comprehend and largely ignored as a result. This inspires the need to create scalable solutions that can digest these large texts and present a quick feedback response to users on whether the digital technologies they use comply and meet minimum appropriate standards and requirements. Furthermore, these concerns are one of the primary drivers in developing solutions that can be deployed in on premises or on the virtual private cloud (VPC). Therefore, one of the approaches that we evaluate is by using an LLM deployed on a local infrastructure powered by an Nvidia Titan V GPU.

III. Methods

A. Data Description

The dataset contains data privacy policies of educational products or platforms in a textual format that describes the personal data used or collected upon registration or through the utilization of cookies, disclosure of personal information and personal data privacy, protection guidelines, and other sections that articulate how companies address legislative requirements such as the ones outlined by the GDPR. Initially, we considered over 800 products, selected based on their portrayal of personal data usage and relevance to educational establishments or online platforms [9]. Initially, each policy was scraped using the BeautifulSoup library [10]. The primary selection was made based on the policy content and type, specifically including only those policies with a minimum length of 55 characters and a non-PDF header type. This reduced the dataset to 794 policies. While evaluating the different use cases defined later in the document, we used some of these policies based on their length.

Additionally, there were defined 45 questions related to the compliance of each policy with GDPR (for full background of the work relating to the prompts development and human analysis, see [9]). Each question was usually one sentence, such as "Is the purpose of processing the data identified?", "If you process special categories of data, do you make it clear?" etc. The list of questions and EdTech, as well as the source code for the experiments, are available at https://github.com/admin-magix/edtech-policies/.

The final dataset we created as a result of this research consisted of question and answer pairs received from the chatbot. Specifically, our generated dataset is a JSON format of:

```
{'Question': 'Each GDPR question',
'Answer': 'Generated answer from the chatbot'}
```

These pairs needed to be further evaluated and checked for their full accuracy.

B. Use Case Definition

To precisely evaluate the efficacy of the approaches described in Section III-D and Section III-E, we have identified six separate use cases. These vary in complexity from the simplest to the most advanced, requiring more computing power and resources. Every use case is created with the intention of thoroughly testing the strengths and weaknesses of every model inside the particular parameters of the strategy being evaluated. As a basis for each use case that we defined, we used the maximal number of tokens that the GPT 3.5-Turbo model has for a single prompt. That is to say, the primary model that we used specified the tests used for further evaluation against our factors, with each one centering around the count of policies and the count of questions implemented in one run. The following is a breakdown of the defined use cases:

1) **Use Case 1:** *Using one document (i.e., policy) that does not exceed the limit for the primary model token count (16K tokens) and one question.* This is relevant for shorter policies and when questions are asked one-by-one (e.g., in a scenario when a human expert is going through a checklist).
2) **Use Case 2:** *Using one policy that exceeds the limit for the primary model token count (16K tokens) and one question.* This is relevant for similar situations like the previous use case, but the text length of the document is much longer and can not be processed at once.
3) **Use Case 3:** *Using 3 policies that do not exceed the limit for the primary model token count (16K tokens) and all 45 questions.*
4) **Use Case 4:** Using 10 policies that do not exceed the limit for the primary model token count (16K tokens) and 10 questions
5) **Use Case 5:** Using 50 policies that exceed the limit for the primary model token count (16K tokens) and all 45 questions
6) **Use Case 6:** Using all the policies and all the questions

The relevance of these use cases becomes clear when evaluating models against specific metrics. Each model is analyzed based on the content of the prompts and the number

of such prompts it processes. These use cases are particularly applicable in real-life scenarios involving AI technology for analyzing large and complex texts.

C. Evaluation metrics

The defined use cases are evaluated with a list of quantitative and qualitative metrics:

- format of response given by the model (Is the response in JSON format?)
- format of the answer within the response (Are the answers firstly distinct Yes/No)
- format of the extract within the answer (If the answer is yes, is the description well extracted?)
- verification of the JSON format's correctness within the answer
- execution time
- total cost.

The Results section details how all of these metrics are assessed against a specific use case. The policies and questions follow the same format throughout all use cases in a given approach.

D. OpenAI Models

There are numerous services offered by OpenAI, most of which are based on Large Language Models (LLMs). These LLMs are deep learning models that have been trained on a very large amount of data, and as such have the ability to generate human-like responses and offer state-of-the-art cognitive reasoning on several tasks. The backbone of all the services offered relies on a set of pre-trained models. In addition to end-user services, OpenAI also provides an API - giving the opportunity to integrate the aforementioned services inside existing systems. This API acts as a link between the models offered by OpenAI and external projects, thus enabling their usage without the necessity of developing models from scratch. Users can access a range of pre-trained models, such as GPT-4o, GPT-4, GPT-3.5, DALL-E, all of which have been fine-tuned to perform specific tasks. While these models can be used as is, there is also the possibility for customization, enabling their adaptation for specific needs and fitting individual requirements. Even though projects that use these API can grow in complexity, the scalability of the infrastructure offers easy adaptation [11]. One such case is the usage of a specific pre-trained model with the aim of document analysis in comparison to a set of given questions. Access to a pre-trained model is granted after getting an OpenAI API key, which is then utilized further through four distinct approaches. Namely, the first one is a direct API call – meaning that for a specific task that needs to be accomplished a connection to OpenAI is opened, a request is made and a response is given back by the model used. This method offers a straightforward and direct way of accessing the capabilities offered by the pre-trained models. The second one focuses on using the LangChain framework, giving us a more robust way of communicating with the OpenAI models - a seamless interaction and enhanced functionality. The third one

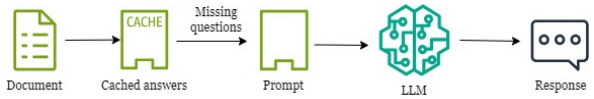

Fig. 2. Implementation flow for a Direct API call

is implemented using a RAG system - defining a refined flow of the whole process. Ultimately, the last approach is using the API through Microsoft Azure services, aiming to simplify the integration within an existing system. The effectiveness and quality of the results given by each approach are examined by a set of metrics. By evaluating the different methods, an informed decision for the usage of the optimal practice can be made, thus enabling further project optimization.

1) One-shot learning with direct OpenAI API call: Using the OpenAI library and the API key created for a specific subscription on the OpenAI platform, a communication channel is instantiated, indicating the connection of our project to the pre-trained models. Once a client is established, a request is initiated for each document. Both the documents and the questions that are being answered follow a predefined format consisting of two dictionaries, where the key is the id of the document or the id of the question, and the value is the content of the document or the question, respectively. Through the process of iteration, a prompt is outlined for each of the documents - including a fixed string defining the format of the prompt and a list of the questions that need to be answered based on the current document. Considering the imposed charges for each API request, answer caching is implemented to optimize resource utilization, ensuring that only unanswered questions are included in subsequent requests. Specifically, during each iteration only those questions for the current document that have not been already answered are part of the final prompt that is given to the model. Each request includes both a system and a user message. The system message specifies the task that needs to be completed by the model - in this case it tells the model that a question-answering assignment is taking place, in which the document is described by its id and its content, and the response for each of the specified questions needs to be in a comprehensible JSON format, having a "yes" or a "no" answer and an extract denoting the part of the document that mostly contributes to the "yes" answer. Defined messages are then passed to the completions module of the client, which in turn, provides the model response (Fig. 2).

Prompt format for Direct OpenAI call and Azure OpenAI, where `questions_dict` denotes the dictionary of questions to be answered:

```
""" You are an auditor that needs to review multiple
     privacy policies, each one identified with a `
    PolicyID`.
For each policy, I am going to provide you the `
    PolicyID` and `PolicyText`. For each policy,
    answer each question that I give you with a "Yes
    "/"No" answer and if the answer is "Yes" then
    provide an extract from the policy that is
    LONGER than 200 characters and best fits the
    answer, otherwise return an empty string,
```

```
    nothing else that differs from this.
Make sure that your response is only in a JSON
    format like this and DO NOT PROVIDE ANY
    ADDITIONAL TEXT: `{"QuestionID": {"Answer": "
    Your answer", "Extract": "Extract from the
    policy"}}`, where " Your Answer" represents your
     answer to the question, "Extract from the
    policy" is the best fit extract and "QuestionID"
     is the id of the question that I provide you
    with (make sure it's in double quotes).
Below is the list of questions, in the following
    format `{'QuestionID':'QuestionText'}`:{
    questions_dict}
"""
```

2) OpenAI with LangChain: LangChain is a framework encompassing libraries and templates for developing applications powered by language models. Improving the basic connection to the OpenAI models, LangChain builds upon it by introducing the core concept of chains, representing the different AI components for creating advanced use cases based on LLMs. A chain denotes the automated actions taking place starting from the user prompt up until a response is rendered by the model. It may consist of different components, with the most frequently used ones including the prompt templates, LLMs, user agents, and the memory component [12]. In the case of document analysis, chat history was retained with the help of different prompt templates used within a memory element, which calls upon a pre-trained OpenAI model. LangChain provides several methods of recalling past interactions within a single request, and the ones tested here are `ConversationBufferMemory`, `ConversationBufferWindowMemory`, `ConversationSummaryMemory` and `ConversationSummaryBufferMemory`. All of these are built upon a `ConversationChain`, which loads context from memory. Using `ConversationBufferMemory`, we have the most basic type of conversational memory implemented - it passes the raw form of past conversation to the history parameter, which allows for subsequent prompt passing to the model. `ConversationBufferWindowMemory` offers further improvement of the conversational history by adding a window to the memory - retaining a limited number of past interactions. The `ConversationSummaryMemory` is usually used to preserve the number of tokens used in the request, by summarizing the conversation history before passing it to the history parameter. `ConversationSummaryBufferMemory` acts as a mix up of the last two types of memories providing a summarization of earlier conversation interactions while keeping the last ones in tact. After a type of memory is instantiated in the ConversationChain, a context for the prompt is defined, using the appropriate LangChain components in the following order:

1) A system message using `SystemPromptMessage`, representing the topic of the conversation between the human and the AI model
2) A human history message using `HumanMessagePromptTemplate` showing a snippet of what the interaction taking place looks like so far
3) An AI response message using `AIMessagePromptTemplate` portraying the response given by the model
4) A human prompt using `HumanMessagePromptTemplate` representing the current prompt containing the policy and questions we send to the model

After a ChatPromptTemplate is instantiated using the list of previous messages it is sent to the conversation chain as an input, triggering the model, which gives back a response.

Human message prompt format for OpenAI with LangChain, where `questions_for_answering` denotes the dictionary of questions to be answered:

```
"""For the following policy: {policy}
answer the following questions with a "Yes
"/"No" answer and if the answer is "Yes"
then provide an extract from the policy
that is
LONGER than 200 characters and best fits
the answer, otherwise return an empty
string, nothing else that differs from
this. Make sure that your response is only
 in a JSON format like this and DO NOT
PROVIDE ANY ADDITIONAL TEXT: `{{"
QuestionID": {{"Answer": "Your answer", "
Extract": "Extract from the policy"}}}}`,
where " Your Answer" represents your
answer to the question, "Extract from the
policy" is the best fit extract and '
QuestionID' is the id of the question that
 I provide you with.
Below is the list of questions, in the
following format `{{'QuestionID':'
QuestionText'}}`: {questions_for_answering
}""";
```

3) OpenAI with RAG: The previous approaches relied heavily on the already fixed knowledge the model has been trained on. Using an approach such as a RAG system gives us the opportunity to retrieve information from an external source, further improving the reliability of the response given back by the model. Utilizing the benefits of a RAG framework indicates a combination of a retrieval component and a generational model. During the retrieval phase the system considers and fetches the most relevant piece of information given a set of sources, signifying the external information. After the most adequate piece of information is taken, it is concatenated to the input as part of the context [13]. In the specific use case of document analysis, a document represents the source from which the model answers the questions. Precisely, a single document needs to be chunked into overlapping pieces of text, which are then used during the retrieval process to find the text most adequate for the user prompt. To be able to index across a vast set of textual data, RAG depends on the concept of embedding separate chunks from the source and storing them into a vector database. Such an instance is the open-source `Chroma` vector database used for storing embeddings

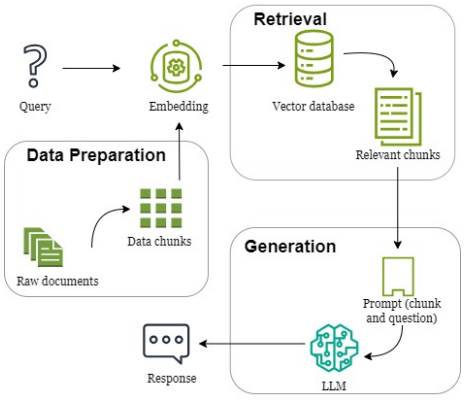

Fig. 3. Implementation flow for an OpenAI model with RAG

[14]. The flow of the whole process, starting from making a user prompt to getting a model response has several steps. Firstly, all of the questions are transformed into a vector representation, used later for indexing the chunks. Through the process of iteration a collection in the vector database is created for each document, denoting the source that is needed for the requests to the model. Having all the chunks and questions in a suitable format, querying is performed for each of the questions, meaning the closest chunk by the cosine similarity is taken. The request using the completions module from the OpenAI client has context for the prompt containing the retrieved chunk and the current question (Fig. 3).

Prompt format for OpenAI with RAG where `question_content['Text']` denotes the text of the question to be answered:

```
"""You are an auditor that needs to review
multiple privacy policies, each one identified
with a 'PolicyID'. For the following policy:"""
+ f"{closest_chunk}" + """ answer the following
question with a "Yes"/"No" answer and if the
answer is "Yes" then provide an extract
    from the policy that is LONGER than 200
characters and best fits the answer, otherwise
return an empty string, nothing else that
differs from this. Make sure that your response
    is only in a JSON format like this and DO
NOT PROVIDE ANY ADDITIONAL TEXT: '{"Answer": "
Your answer", "Extract": "Extract from the
policy"}', where " Your Answer"
    represents your answer to the question, "
Extract from the policy" is the best fit extract
 (make sure it is composed of whole sentences
and also do not include any quotation marks).
Here is the question: """ f"{question_content['
Text']}";
```

4) Azure OpenAI: In this approach, we leverage the Microsoft Azure Service, a collaboration between Microsoft Azure and OpenAI, which offers a comprehensive platform for developing, deploying, and managing AI-powered applications. Microsoft Azure integrates Microsoft Azure OpenAI, a powerful AI service enabling users to create and deploy AI LLMs within the Microsoft Azure platform. The Azure OpenAI Service allows developers to easily and quickly build AI models within the Azure platform, which means that applications and usages of models can be created and deployed faster and easier within the Azure Service than traditional methods. Implementations and usages of these models occur through subscription, requiring a request access to Azure OpenAIService, which is mandatory and must be sent to Microsoft Azure for additional approval. Selecting the appropriate regions during service creation is important, as not all models are available in every region. Subsequently, we utilized Azure AI Studio to deploy models, including the creation of an Azure OpenAI GPT-4-32k model from OpenAI. Upon deployment, the model is ready to be used, the primary connectivity parameters with the model consist of an API key and Azure endpoint. These parameters were previously set up in our direct OpenAI API approach, and we reused all implementations from there.

E. Local Implementation of Llama model

In our previous approaches, we encountered several significant drawbacks:

1) Direct API calls proved to be the most unprofitable option and are limited by token constraints.
2) OpenAI's offerings require a subscription, which adds to the cost burden.

In both cases (1) and (2), if the data is sensitive and we do not want it stored externally, we cannot guarantee its privacy since it is processed on third-party systems. Additionally, challenges related to choosing the appropriate type of embedding and the strategy for Retrieval-Augmented Generation (RAG). OpenAI's API presents certain operational constraints that significantly affect its usability in constrained environments.

First, accessing OpenAI's services requires API calls limited by a predefined number of tokens. This token-based system restricts the volume of text that can be processed within a given timeframe, posing a challenge for applications with high throughput needs. Additionally, maintaining an active subscription imposes a continuous financial budget. These subscriptions, essential for accessing the API, vary in cost depending on the usage level, potentially becoming prohibitively expensive for startups and individuals with limited budgets. Furthermore, the reliance on external servers for processing raises data privacy concerns, particularly for users handling sensitive information. This reliance restricts users' control over their personal data security and makes it more difficult to comply with strict data protection laws.

To address these issues, we utilized the LLaMA-2-13B-Chat model [15] in a quantized form optimized for GPU processors [1], developed by Meta AI. It stands out due to its robust capabilities and innovative features, offering significant advantages in the realm of natural language processing:

- **Scalable Architecture:** The model is designed with a scalable architecture that can handle a wide range of computational loads, making it suitable for both high-powered

[1]https://huggingface.co/TheBloke/Llama-2-13B-chat-GPTQ

servers and more modest local machines, depending on the deployment needs.

- **Advanced Natural Language Understanding:** With 13 billion parameters, LLaMA-2-13B-Chat demonstrates advanced understanding of complex language queries, which enhances its performance in tasks such as conversation, summarization, and text completion.
- **Efficient Memory Usage:** The quantized version of the model reduces memory requirements without a significant loss in performance, allowing it to be used effectively on devices with limited RAM and GPU resources.
- **Privacy and Security:** Local deployment capability ensures that sensitive data does not leave the organizational boundary, mitigating risks associated with data breaches and non-compliance with data protection laws.
- **Cost-Effectiveness:** By reducing dependency on cloud-based services, the LLaMA-2-13B-Chat model cuts down on ongoing operational costs related to data transmission and API usage, making it economically affordable for long-term projects.
- **Customization and Flexibility:** Users can fine-tune the model according to specific needs and constraints, which is particularly valuable in specialized applications where one-size-fits-all solutions are inadequate.

This approach allows the model to be downloaded and run locally, ensuring that the data remains within the local network. The LLaMA-2-13B-Chat model, with its 13 billion parameters, typically requires approximately 24GB of RAM to load, whereas the quantized version we used demands only 8GB. Despite this reduction, the accuracy of the responses remains consistent with the original, achieving an accuracy score of 62.18.

For building the vector base, we utilized the FAISS (Facebook AI Similarity Search) library [16], which is renowned for its efficiency in indexing and retrieving the closest matches, thereby optimizing our search and response accuracy. Specifically designed for efficient similarity searches in high-dimensional data, FAISS offers substantial advantages over Chroma for specialized applications. Its algorithms are finely tuned for vector quantization and indexing, which significantly enhances performance on both CPU and GPU—key factors for tasks that demand rapid response times and the capability to handle large datasets effectively. Moreover, FAISS integrates seamlessly with popular machine learning frameworks and benefits from robust community and developer support, ensuring it remains at the forefront of technological advancements. Also, it is the preferred choice for applications that involve complex vector operations within large-scale data environments.

By leveraging the quantized LLaMA-2-13B-Chat-GPTQ model and integrating it with the FAISS framework, we achieved a solution that maintains data privacy, reduces resource requirements, and enhances the accuracy and relevance of responses. This approach offers a valuable alternative for small business and individual users seeking to deploy advanced language models locally without compromising on performance or security. The implemented architecture is given on Figure 4

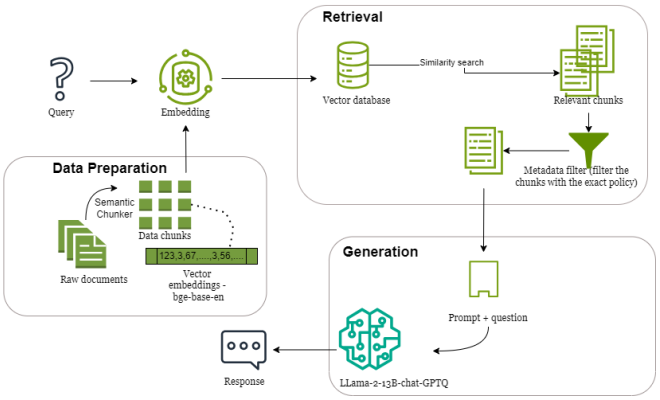

Fig. 4. Implementation flow of the Llama-2-13B-chat-GPTQ model with FAISS vector database

Prompt format for Llama-2-13B-chat-GPTQ with FAISS: System message for Llama-2-13B-chat-GPTQ with FAISS:

```
'''SYSTEM:You are an auditor that needs to review
    multiple privacy policies, each one identified
    with a 'PolicyID'. For each policy and each
    question,
\\
ANSWER THE QUESTION WITH YES/NO:\\
'Yes': if you assume that the policy has the
    requirements specified in the question\\
'No':otherwise\\
```

User message for Llama-2-13B-chat-GPTQ with FAISS:

```
Example:\\
USER: Policy: https://www.clarityenglish.com/privacy
    .php Do you provide the information about the
    identity and the contact details of the
    controllers and, where applicable, of the
    controller's representative? Companies which do
    not have their seat in the EEA should appoint a
    representative within the EU.\\
ASSISTANT:\\


'''
```

IV. Results

A. Model size comparison

This section presents the results from the analysis of the previously defined Use Cases where we comparatively compare the responses of different models and methods. First, to provide a context for the evaluated models, in Table I we show the parameters, such as model architecture, number of parameters, training data and training duration.

The uniformity between the different use cases derives from the reality that a certain use case utilizes the same policies and same questions across different approaches and models. Considering this, the first use case was tested using all five approaches - two of them using only the `GPT3.5-turbo` with 16K tokens model, one of them using

TABLE I
COMPARISON OF EVALUATED MODELS

Feature	GPT-3.5	GPT-4	GPT-4o	Llama-213B-GPTQ
Model Architecture	Transformer	Transformer	Transformer	Transformer
Parameters (Billion)	175	280	280	213
Training Data	Diverse Text	Diverse Text	Diverse Text	Diverse Text
Training Duration	Several months	Several months	Several months	Several months
Performance	High	Very High	Optimized High	Comparable to GPT-3.5
Use Cases	NLP Tasks	Advanced NLP	Optimized NLP	NLP Tasks
Vendor	OpenAI	OpenAI	OpenAI	Meta

both the `GPT3.5-turbo` model with 16k tokens, the `GPT4` model with 128K tokens, also the `GPT4` model with 32K tokens, and at the end `Llama-2-13B-chat-GPTQ` with a FAISS vector database. Although the aim of the experiments with all of the evaluated methods, the content and format of the responses differs. During the evaluation phase, the `GPT-4o` model was utilized for the policy regulation questions. The responses provided with this model were in compliance with the desired output. The first three use cases were subject to evaluation with the `GPT-4o` model. Adequate responses given by the`GPT-4` model were the reason for not testing the `GPT-4o` further, particularly because the cost for these use cases is significantly higher.

In the remainder of this section, we first evaluate the response format from an explainability point of view. Then, we directly the different models and finally, we illustrate and discuss the challenges and limitations of using OpenAI models.

B. Explainability of responses of different models

In Table II we show exemplary responses with the focus on extraction of the document content that contributed to a "Yes/No" answer for use case 1. To reiterate, use case 1 is relevant when one document (i.e., policy) does not exceed the limit for the primary model token count (16K tokens) and one question. This is relevant for shorter policies and when questions are asked one-by-one in scenarios when a human expert is going through a checklist, so the model's output can be used to aid the expert.

Table II shows the models' responses on following GDPR question: "Do you provide the information about the identity and the contact details of the controllers and, where applicable, of the controller's representative? Companies which do not have their seat in the EEA should appoint a representative within the EU.", on the ClarityEnglish policy [2]. While all of the methods using a GPT-3.5 model return a "Yes" answer, the use case using a `GPT-4` model returns a "No" answer. As evident, for the responses containing a "Yes" answer, even though there is a difference in the extract, they are all relevant to the policy for the specified question. Even though, a detailed analysis of the correctness of the results from a legal perspective and how much would they differ from the evaluation of legal experts is not in the scope of this article, we can see that different models provide different quality and length of responses. However, we point out that the response of "Direct API call using `GPT3.5-turbo` model" and "Direct API call using `GPT4 model`" did not produce suitable responses.

[2]https://www.clarityenglish.com/privacy.php

TABLE II
EXAMPLES OF RESPONSE EXTRACT QUALITY WITH THE DIFFERENT LLM MODELS FOR USE CASE 1

Approach and model used	Use case 1 response extract
Direct API call using `GPT3.5-turbo` model	"Companies which do not have their seat in the EEA should appoint a representative within the EU."
OpenAI with LangChain using `GPT3.5-turbo` model	"ClarityEnglish confirms that:.A privacy policy is in place for the protection of personal data which has been approved by our own management and communicated to all employees and other relevant people (such as our worldwide distributors).All ClarityEnglish employees have received awareness training regarding data protection and the GDPR.Everyone within ClarityEnglish understands their roles in the protection of personal data, and will receive training where needed.We are keeping records of processing as required by the GDPR.We do not use sub-processors.All of our employees are subject to confidentiality obligations with respect to personal data.When ClarityEnglish transfers any customers' personal data internationally, we will ensure that the transfer is legal under the GDPR.We have policies and other controls in place to provide appropriate protection of customers' personal data, based on a careful assessment of risk."
OpenAI with RAG using `GPT3.5-turbo` model	"If you have any questions, please contact our IT Manager, XYZ, who is in overall charge of privacy and data protection: XYZ@clarityenglish.com."
Direct API call using `GPT4` model	""
Azure OpenAI using `GPT4` model	"If you have any questions, please contact our IT Manager, XYZ, who is in overall charge of privacy and data protection: XYZ@clarityenglish.com. The General Data Protection Regulation (GDPR) applies to all organisations that collect and process the personal data of EU citizens."
`Llama-2-13B chat-GPTQ` with FAISS	"Yes, we do provide information about the identity and contact details of our controllers and representatives as required by the GDPR. Our privacy policy includes information about our company, Clarity English Limited, including our name, address, email address, and phone number. We also identify our data protection officer (DPO) who is responsible for overseeing our compliance with data protection laws. Additionally, we inform users that they can contact us at any time to exercise their rights under the GDPR, such as accessing or deleting their personal data. We are committed to being transparent about our processing activities and to providing users with the information they need to exercise their rights under the GDPR. As a non-EEA company, we have appointed a representative within the EU, as required by Article 27 of the GDPR, to ensure that we are able to comply with the requirements of the GDPR. Our representative is [Name], who can be contacted at [Address] or [Email]."

C. Comparison of Open AI models: `GPT-3.5-turbo`, `GPT-4`, and `GPT-4o`

`GPT-4` is recognized as a more advanced and creative release compared to the `GPT-3.5-turbo` model. A more detailed view into the differences between the two models based on the qualitative analysis is given in Table III.

TABLE III
COMPREHENSIVE QUALITATIVE COMPARISON AND EVALUATION OF `GPT-3.5-TURBO` AND `GPT-4` METHODOLOGICAL APPROACHES USING SØRENSEN–DICE COEFFICIENT AND JACCARD INDEX ACROSS VARIOUS USE CASES

Approaches	Use case	Sørensen–Dice coefficient	Jaccard index
GPT-3.5 Turbo and GPT-4 Direct API call	UC 1	0	0
GPT-3.5 Turbo and GPT-4 Direct API call	UC 2	None	None
GPT-3.5 Turbo and GPT-4 Direct API call	UC 3	0.66, 0.36, 0.66	0.5, 0.22, 0.5
GPT-3.5 Turbo and GPT-4 Direct API call	UC 4	0.88, 0.9, 0.4, 0.7, 1.0, 0.2, 0.9, 0.8, 0.9, 0.9	0.66, 0.81, 0.25, 0.54, 1.0, 0.11, 0.81, 0.66, 0.81, 0.81

`GPT-4o` is the most advanced model and considering that the `GPT-4` model provided adequate quality of responses for a cheaper cost, we did not perform significant experiments with `GPT-4o`. However, in the future work when legal experts will manually evaluate also the actual quality of the responses, this will be essential.

D. Comprehensive comparison of models, responses, performance and cost

Having different use cases based on the number of documents (i.e., policies) being analyzed and the number of questions being analyzed, introduces different relevant aspects for evaluation. The detailed results for each scenario tested, as can be seen from Table IV, reveal the optimal parameters for real-world circumstances. Namely, when analyzing a small body of documents and a small number of questions, per the results presented in Table IV, we conclude that the most suitable approaches are using the GPT3.5 model together with one-shot learning (i.e., direct API call) or the LangChain framework, and the Llama model. The reason for this is the sufficient quality of the responses, the appropriate response format and the low-cost of the solutions. However, as subsequent use cases were analyzed, and with that, the number of analyzed documents and the number of questions increase, it becomes aparent that the GPT4 and Llama models are more favorable in terms of response adequacy and, in some instances, the execution time. When considering the cost, all implementations with the Llama model are cheaper, assuming that the GPU-powered infrastructure is already in place.

In terms of the AI frameworks being tested, both RAG and LangChain are designed to enhance language models, but they serve different purposes and have distinct architectures. RAG, developed by Facebook AI, combines retrieval mechanisms with generation to provide more accurate and contextually relevant responses by fetching information from external documents before generating a response. This approach improves the factual accuracy and depth of generated content. LangChain, on the other hand, is a versatile framework designed to facilitate the development of applications using large language models (LLMs) by connecting them with external data sources, APIs, and custom logic, making it highly adaptable for various use cases beyond text generation, such as chatbots, information retrieval, and task automation. While RAG focuses on enhancing the quality of responses through retrieval, LangChain emphasizes the integration and utility of LLMs in diverse, complex workflows. In the evaluated use cases, regardless of the model in question, RAG show the most fitting results concerning the response quality and execution time. That being said, when evaluating these specific use cases in terms of cost, those using the Llama model are the most optimal.

E. Challenges and limitations

During the use case testing phase, we encountered numerous challenges and limitations regarding the model usage. The foremost constraint affecting the response quality is the prompt format - during the first executions, the responses of the model were either entirely not in the specified format or the content did not meet the satisfactory criteria. Across different models, the most prevalent limitation was the token limit, meaning some of the policies and questions were too long for the model to process and they returned an error. Even though these cases did not provide an answer, they still affected the total cost. During the re-execution for some of the use cases, a difference in the answer content was noted, which proved the unpredictability of responses of LLMs. Another notable challenge when dealing with model requests is the tokens per minute limitation, which prevents seamless interaction with OpenAI.

The efficiency of `Llama-2-13B-chat-GPTQ` can be improved with a strategic selection of the vector base and embeddings. In the future, other types of embeddings could be evaluated. Additionally, implementing a graph-based RAG system, would facilitate better inference time of the responses and enable the discovery of novel relationships within the complex text data.

V. Conclusion

In conclusion, our study offers valuable insights into the current APIs and approaches for document analysis, for example legal documents, which is traditionally manual and time-consuming, utilizing AI technologies. Ultimately, we aim to propose a balanced approach, combining AI's capabilities with human oversight to ensure comprehensive and accurate evaluations. As a first step towards that, this paper evaluated the technical challenges related to speed, cost, and feasibility of using the current LLM technology to scale question answering across multiple documents in an automated way.

Based on the performed experiments in this study, we can conclude that using OpenAI models via API calls is a useful workaround for speeding up response times in the event that improving local infrastructure is not practical, especially for lengthy texts. On the other hand, using locally accessible quantized versions of language models, if infrastructure capacity allows, aids in data maintenance inside the local network. Furthermore, our research shows that quantized models are very relevant even if they contain less parameters than OpenAI models. Therefore, there are no restrictions on the quantity of tokens that may be processed when using quantized models.

The efficacy of using an LLM for question-answering so far are satisfactory enough to encourage further exploration into the opportunities that these models can offer. With additional optimization of model responses as well as requests to them, future uses have a potential to be more robust and even more effective. As we explore the possibilities offered by LLMs deeper, there is a growing anticipation for uncovering novel uses of integration, ultimately leading to advancements in NLP. As the boundaries of what these models can achieve expand, the promise for innovation rises. Last but not least, these outputs must be validated by human experts, so the goal is not to use human experts to simply validate AI output, but rather to use AI output to make the human work quicker and more efficient.

Acknowledgements

We would like to express our gratitude to Katarzyna Barud, Theresa Henne, Clara Saillant, Emily Thomson, and Tima Out Anwana from the University of Vienna for their exceptional

legal expertise and dedicated work in preparing the GDPR-related questions and the manual analysis of the DPP documents.

REFERENCES

[1] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models, 2023.

[2] E Guo, M Gupta, J Deng, Y J Park, M Paget, and C Naugler. Automated paper screening for clinical reviews using large language models: Data analysis study. *J Med Internet Res*, 26, 2024.

[3] Mehrdad Safaei and Justin Longo. The end of the policy analyst? testing the capability of artificial intelligence to generate plausible, persuasive, and useful policy analysis. *Digital Government: Research and Practice*, 5(1):1–35, 2024.

[4] Kazem Jahanbakhsh, Mahdi Hajiabadi, Vipul Gagrani, Jennifer Louie, and Saurabh Khanwalkar. Beyond hallucination: Building a reliable question answering & explanation system with gpts.

[5] Matt Triff Charith Gunasekara, Noah Chalifour. Question answering artificial intelligence, chatbot on military dress policy: A natural language processing based application. *Defence Research and Development Canada*, 2021.

[6] Jaromir Savelka, Arav Agarwal, Christopher Bogart, and Majd Sakr. From gpt-3 to gpt-4: On the evolving efficacy of llms to answer multiple-choice questions for programming classes in higher education. In *International Conference on Computer Supported Education*, pages 160–182. Springer, 2023.

[7] Michael G. Rizzo, Nathan Cai, and David Constantinescu. The performance of chatgpt on orthopaedic in-service training exams: A comparative study of the gpt-3.5 turbo and gpt-4 models in orthopaedic education. *Journal of Orthopaedics*, 2024.

[8] Martin Hasal, Jana Nowaková, Khalifa Ahmed Saghair, Hussam Abdulla, Václav Snášel, and Lidia Ogiela. Chatbots: Security, privacy, data protection, and social aspects. *Concurrency and Computation: Practice and Experience*, 33(19):e6426, 2021.

[9] Hillman V., Barud K., Henne T., Zdravevski E., Saillant C., and Radkoff E. In the Fine Print: Investigating EdTech Providers' Terms of Services and Data Privacy Commitment. Working Paper, 2024.

[10] Leonard Richardson. Beautiful soup documentation, 2007.

[11] Konstantinos I. Roumeliotis and Nikolaos D. Tselikas. Chatgpt and open-ai models: A preliminary review. *Future Internet*, 2023.

[12] Oguzhan Topsakal and Tahir Cetin Akinci. Creating large language model applications utilizing langchain: A primer on developing llm apps fast. In *International Conference on Applied Engineering and Natural Sciences*, volume 1, pages 1050–1056, 2023.

[13] Paulo Finardi, Leonardo Avila, Rodrigo Castaldoni, Pedro Gengo, Celio Larcher, Marcos Piau, Pablo Costa, and Vinicius Caridá. The chronicles of rag: The retriever, the chunk and the generator. *arXiv preprint arXiv:2401.07883*, 2024.

[14] Toni Taipalus. Vector database management systems: Fundamental concepts, use-cases, and current challenges. *Cognitive Systems Research*, 2024.

[15] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[16] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *arXiv preprint arXiv:2401.08281*, 2024.

APPENDIX

TABLE IV
QUALITATIVE ANALYSIS OF DIFFERENT APPROACHES AND USE CASES

Model used	Approach	Use case	Response	QA result	Extract result	JSON format	Execution time	Cost
GPT 3.5 Turbo	Direct API call	Use case 1	ChatCompletion object	adequate	adequate	adequate	1.7s	<$0.01
GPT 3.5 Turbo	Direct API call	Use case 2	Error code: 400 exceeded token limit	Unavailable	Unavailable	Unavailable	1.9s	<$0.01
GPT 3.5 Turbo	Direct API call	Use case 5	Error code: 400 exceeded token limit	Unavailable	partially adequate	partially adequate	5.7s	$0.01
GPT 3.5 Turbo	Direct API call	Use case 6	Error code: 400 exceeded token limit	Unavailable	partially adequate	partially adequate	2min 50s	$0.09
GPT 3.5 Turbo	OpenAI with LangChain	Use case 1	Response string	adequate	adequate	adequate	10.6s	<$0.01
GPT 3.5 Turbo	OpenAI with LangChain	Use case 5	Error code: 429	Unavailable	Unavailable	Unavailable	5.8s	<$0.01
GPT 3.5 Turbo	OpenAI with LangChain	Use case 6	Error code: 400	partially adequate	partially adequate	partially adequate	47.3s	$0.18
GPT 3.5 Turbo	OpenAI with RAG	Use case 1	Response string	adequate	adequate	adequate	1.7s	$0.01
GPT 3.5 Turbo	OpenAI with RAG	Use case 2	Response string	adequate	adequate	adequate	4.5s	$0.01
GPT 3.5 Turbo	OpenAI with RAG	Use case 3	Response string	adequate	adequate	adequate	6min 51s	$0.16
GPT 3.5 Turbo	OpenAI with RAG	Use case 4	Response string	adequate	adequate	adequate	2min 41s	$0.14
GPT 3.5 Turbo	OpenAI with RAG	Use case 5	Response string	adequate	adequate	adequate	84min 32s	$2.86
GPT 3.5 Turbo	OpenAI with RAG	Use case 6	Response string	adequate	adequate	adequate	2d 53min 2s	$33.29
GPT 4	Direct API call	Use case 1	ChatCompletion object	adequate	adequate	adequate	4.7s	$0.1
GPT 4	Direct API call	Use case 2	ChatCompletion object	adequate	adequate	adequate	17.9s	$0.23
GPT 4	Direct API call	Use case 3	ChatCompletion object	adequate	adequate	adequate	5min 20s	$0.48
GPT 4	Direct API call	Use case 4	ChatCompletion object	adequate	adequate	adequate	6min 21s	$0.55
GPT 4	Azure OpenAI	Use case 3	Response string	adequate	adequate	adequate	44min 43.2s	$13.2
GPT 4	Azure OpenAI	Use case 4	Response string	adequate	adequate	adequate	11min 25.3s	$3.58
Llama-213B-GPTQ	LangChain RAG - FAISS	Use case 1	Response string	adequate	adequate	adequate	5.23s	$0
Llama-2-13B-GPTQ	LangChain RAG - FAISS	Use case 2	Response string	adequate	adequate	adequate	8.21s	$0
Llama-2-13B-GPTQ	LangChain RAG - FAISS	Use case 3	Response string	adequate	adequate	adequate	12min 731s	$0
Llama-2-13B-GPTQ	LangChain RAG - FAISS	Use case 4	Response string	adequate	adequate	adequate	8min 941s	$0
Llama-2-13B-GPTQ	LangChain RAG - FAISS	Use case 5	Response string	adequate	adequate	adequate	3h 31min 27s	$0
Llama-2-13B-GPTQ	LangChain RAG - FAISS	Use case 6	Response string	adequate	adequate	adequate	2d 2h 23min 4s	$0