Designing LLM agent tools for due diligence in financial instruments:

CMBS securitisation and equity structured notes

LSEG

Executive summary

This document outlines our approach to developing applications utilizing Large Language Models (LLMs). The focus of this proposal is narrowed to a specific application: question-answering due diligence inquiries. We have chosen this particular use case due to its significant potential to deliver tangible business benefits to our client base.

Investors need to dissect and understand the pertinent nuances of each document, whether it is for securitised prospectuses often exceeding 350 pages with highly specialised language [1] or structured notes which are shorter documents that require bulk processing. Although LLMs are well-suited for sophisticated Q&A tasks, further investigations are required to ascertain their reliability in a financial domain context. To ensure accuracy we are investigating the application of a retrieval-augmented generation (RAG) system.

Our goal is to examine and optimise the configuration of the RAG system and test its limits for the above tasks using generally available LLMs. Our conclusions are grouped in the following categories:

- the amount of context we need to provide to the LLM to enable it to effectively answer queries
- the impact of the novel GPT-128k long context window from the Nov 2023 release alongside cost and scalability trade-offs that arise from it
- whether LLMs can correctly identify supporting evidence for a query
- the performance of the answer generation given the target responses: whether the answer is a paragraph or specific quantity such as notional amounts, issuer names, dates, lists, etc.

We also pinpoint areas where the proposed system's performance could be improved. Finally, we look beyond a question-answering (RAG) tool and highlight the steps needed to transform it into an interactive system via conversational LLM agents capable of actively engaging with the user.

1 They cover various deal types - examples include residential mortgage-backed security (RMBS), commercial mortgage-backed securities (CMBS) and asset-backed securities (ABS).

Overview of the approach and outcomes

At London Stock Exchange Group (LSEG), our mission is to optimise our clients' efficiency throughout the entire trade lifecycle. The complexity of securitisation documents, with their intricate legal details and term specifications, can often make them seem overwhelming. Investors, traders and salespeople must meticulously analyse various aspects of a security, including its overall structure, individual loan mechanics and seniority structures, as part of their due diligence. Similarly, equity structured notes require a precise understanding of nuances in term definitions from varying issuers and with variation in the use of lexicon. While these documents are shorter, customers need to quickly and at scale identify the mechanics of guarantees/protection, pay-out formulas, governing laws, etc. The primary tool at the investor's disposal is PDF keyword search, which can often be time-consuming and inefficient in locating precise answers and all the relevant context.

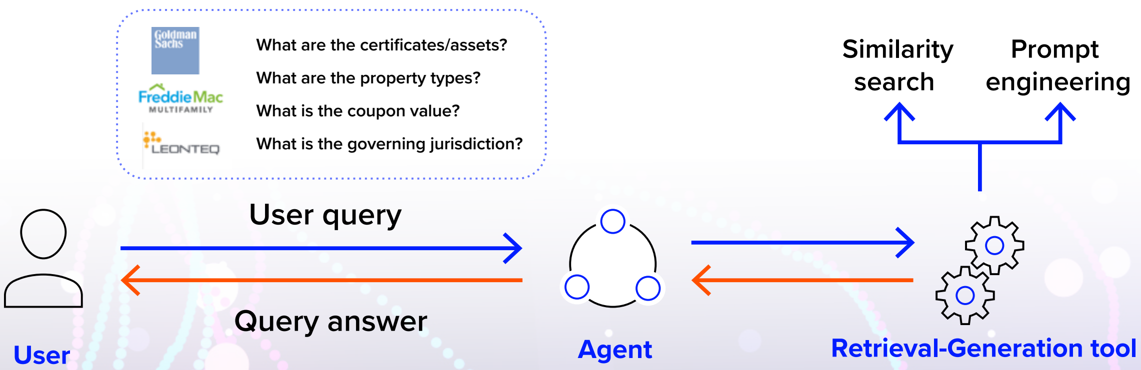
Large Language Models (LLMs) are ideally suited to tackle this challenge, offering a natural language interface capable of delivering contextually relevant responses. However, the obstacle lies in the fact that LLMs alone cannot "learn" specific deal documentation accurately through fine-tuning – and the resulting answers can easily be "hallucinated". A prevalent solution to this problem is the implementation of a **retrieval-augmented generation (RAG) system**. This system combines efficient document storage and retrieval using vector databases to select relevant text snippets. After that, an LLM is employed alongside prompt engineering methods to generate an accurate answer to the user query from the associated retrieved snippets.

To ensure scalability, it is crucial to maintain both repeatability and precision within these experiments. While the RAG method has been extensively researched for a variety of general use cases, in deep domain-specific context, particularly in finance, it merits further investigation. Consequently, the objective of this paper is to identify the optimal setup of ML systems for such use cases.
We approach this in the following ways:

– Identifying the right metric by measuring ourselves against the right questions
– Considering the trade-offs between long context LLMs and a RAG solution for our use case (i.e. by analysing the recently released 128k context GPT4 by OpenAI)
– Finding the optimal setup of such a system by individually analysing the following components: vector database similarity search, LLM context comprehension and the quality of the LLM-generated answers.
– Identifying further components needed for an optimal system setup, such as UI & UX components, LLM approaches, etc.

To evaluate the model's capabilities, subject-matter experts (SMEs) selected a set of high-value questions for the investment due diligence process. These questions target key features of the security, like the offered assets and their principal allocation/nominal value, the identity of the relevant entities, geographical spread and more.

Figure 1: General view of document question-answering system



Source: LSEG, Nov 2023

In addition to focusing on the main details from the provided documentation, these questions were designed to test a range of language comprehension challenges for the LLMs, including understanding names, dates, locations, lists and tables. This diverse questioning aims to reveal the model's strengths and limitations. We have divided our experimentation into the three primary components of a functional RAG tool:

- **Experiment 1: similarity search –** we aim to identify sections of the document containing relevant information for answering our query. We discovered that typically up to **five search results are sufficient to build a representative context for the model**. This approach has an efficiency component as it reduces the volume of information sent to the LLM, thus reducing operational costs and system latency.
- **Experiment 2: context comprehension –** we evaluate the LLM's ability to correctly identify supporting evidence within the text snippets returned from the similarity search. In some cases, we may find it useful to return a direct quotation from the source document or reinforce an LLM-generated answer with the original text. For these cases, it will be sufficient for the model simply to identify the correct supporting text. On average, the **model accurately identifies the text snippet containing the answer 76%** of the time and effectively disregards paragraphs lacking relevant information to the user's query 91% of the time.
- **Experiment 3: answer quality –** we analyse responses for queries with two distinct purposes: **value extraction** (where the answer is a specific value, e.g., notional amount, date, issue size, etc.) and **textual answers** (where the answer is textual and is contained within a sentence or a paragraph). For both tasks we compare the performance of both GPT3.5 and GPT4 models, with the latter consistently demonstrating superior results. For value extraction tasks, GPT4's **accuracy ranges between 75-100%**, whereas for textual information extraction, the **quality of the generated answers ranged between 89-96%**, depending on the complexity of the task. The 128k context window tends to perform here largely on par or slightly worse than the traditional shorter windows.

Based on these observations, we can confidently state that the proposed methodology is suitable for integration into workflows that require handling of substantial workloads involving extensive documentation sources. However, it is important to recognise that such systems tend to not be fully accurate, hence we also discuss potential ways to mitigate this, for instance by referencing source materials and suitable UX components.

Data processing

In this chapter, we delve into the data used and the measures we took to prepare it for Retrieval Augmented Generation (RAG) system. Our first challenge presents itself in the format of the available PDF documents, which can span more than 350 pages. LLMs typically have token length restrictions, and that limits the amount of context that can be provided. Using longer context windows comes with a trade-off of slower response times and lower quality of results [1]. In Nov 2023, OpenAI introduced the 128k context window version of GPT4, which offers an option to choose from two approaches: one with short context and one with long context. The first is using short text chunks which offer a lot of reuse, the challenge is in picking the right text chunks. The second method is simpler and involves feeding larger portions of the document into the long context model and try to answer all questions relying on the full scope of text. For the first, we will outline the three steps employed to select the right snippets of text for the RAG system's context: document splitting, text snippets embedding and vector database choice. Our goal is to pinpoint pertinent information within the document by utilising the cosine similarity metric to extract the most relevant text snippets from the vector database.

In the remainder of the document, we will refer to this as "short context" setup or simply GPT3.5 and GPT4 and when referring to the new 128k context window usage we will refer to as "long context" or GPT4-128k.

Data in scope

For this study, we chose Freddie Mac Structured Pass-through Certificates (SPCs) [2] and structured notes term-sheets (SNs) from various providers. Our choice was guided by our domain experts, our clients' needs and the challenging structure of the documentation. The SPCs are notably detailed and filled with valuable information, while the SNs are more concise – however, information is presented in multiple formats that typically vary by provider.

Given the extensive and complex nature of these documents, it is vital to identify methods of accessing their content more efficiently. In most instances, clients do not need to comb through the entire document to locate relevant details; they can focus on specific sections or receive an LLM-generated answer. The data contained in these documents is found in various formats – tabular data merges with textual paragraphs, ordered lists, simple diagrams or key-value pairs. The research dataset comprises 95 SPCs documents and 18 SN term-sheets.

Choosing which questions to ask

We aim to statistically determine the reliability of questions answered by LLMs based on specialised documents. Our team has meticulously selected a set of queries that serve two crucial purposes:

- they are business-relevant, representing the high-value questions that our customers are most likely to ask from the documents.
- each question poses a unique challenge to our retrieval-augmented generation system.

Some questions necessitate succinct and accurate answers while others call for a comprehensive summary of the source content. There are also complex questions that involve multiple nested queries and others that impose specific restrictions on the LLM. The comprehensive list of questions and additional details are provided in Appendix 1.

Document splitting - short context LLMs

When dealing with the shorter-context models GPT3.5 and GPT4, we face limitations in the length of our document inputs and prompts. This limit varies across different GPT models, with the longest prompt length offered by OpenAI [3] being a variant of the GPT4, which accepts an input of 32k tokens. To illustrate, the documents in the dataset have an average page length of 900 tokens, with the SPCs documents containing more than 350k tokens per document. Consequently, it becomes essential to divide the document into smaller, manageable chunks that fit within the token limit of the model used and are efficient for a scalable system.

To that end, the documents are divided at the sub-page level, ensuring that the informational context is preserved for each snippet through overlapping. Text snippets known to contain the answers to a predefined list of queries are labelled accordingly, while most contain information outside of the defined investigations, acting as “noise” for the proposed system. On average, each processed document snippet ranges between 150 and 1,000 tokens in size. The proposed document splitting method shows no context loss in the yielded text snippets, yet an issue of efficiency arises due to redundant information being present across multiple text snippets, which translates to inefficient API usage. There is more work to be done to identify the optimal method of splitting documents.

For the system’s analysis we utilised the GPT3.5-4k context and the GPT4-8k context, two readily accessible models with sufficient capabilities and input token lengths for the task at hand. We recognise that there are several longer context models [4], however, there have been observations of rapidly declining performance with context lengths longer than 16k tokens. Additionally, we found GPT4-32k to be very cost inefficient – twice as expensive than GPT4-8k and 40 times more expensive than GPT3.5-4k as of the publishing date.

Long context LLM

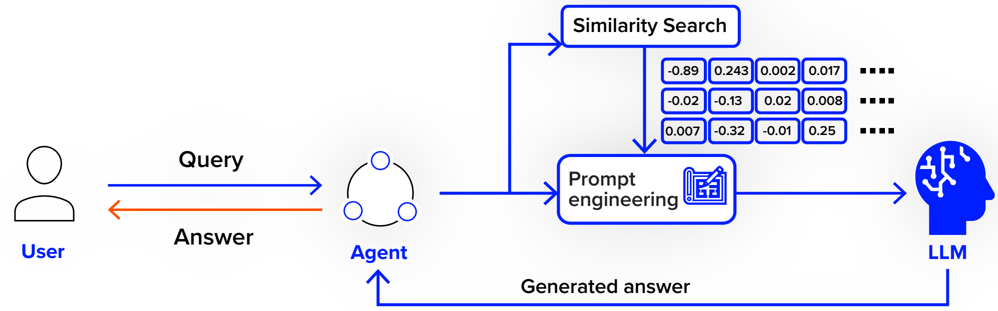
With the latest advancements in GPT technology we are progressively able to use longer context windows for providing documentation to the LLM. One such model that we investigate is GPT4-turbo-128k which provides enough input capacity to use the testing documents in full, without the need of splitting. It is a matter of ongoing research as to how much retention can long context models actually offer, with some results available here [4]. There are many factors behind the decision of splitting the available documents and utilising them as a whole. The main ones that drive the design decisions of the system are: the documents themselves, as they can be larger than the context window of the LLM, and the practical application of the Q&A tool.

Retrieval-augmented generation setup

This is the component with which the end-user interacts with the model through the lense of the LLM agent, performing multiple ordered steps to achieve the final goal of providing a reliable answer. The method relies on extracting the most similar document snippets from the database through similarity search while making use of LLMs with prompt engineering methods to get a final answer from the retrieved context.

The subsequent sections will delve into the methodologies employed, key findings from testing and an evaluation of the system's performance.

Figure 2: Retrieval-generation methodology

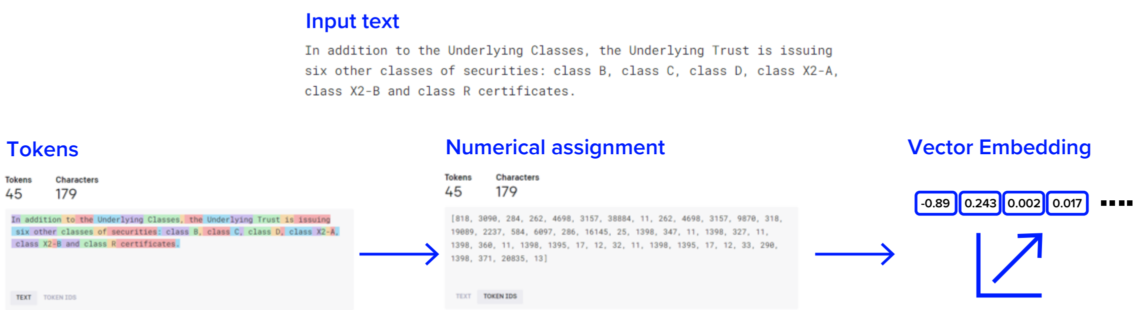


Source: LSEG, Nov 2023

Embeddings

The first step is tokenisation, where we convert human-readable text into a machine-understandable numerical format. Each word, sub-word or punctuation mark in the English language is assigned a unique numerical value. This list of values is then used to create a fixed-size vector embedding, capturing the contextual meaning of the tokenised text as depicted in Figure 3 below. Maintaining the vector size independent of the input text length is an important aspect of the embedding process as it allows for efficient semantic comparison between a user query consisting of a few words and paragraphs with hundreds of words.

Figure 3: Transforming text to machine-readable data: text gets tokenised (middle); each token is represented by a number (right)



Source: LSEG, Nov 2023

There are various embedding models available, with our focus being towards assessing the effectiveness of the following state-of-the-art choices: OpenAI text-embedding-ada-002 (OpenAI) [5], MpNET-v2 [6] (Sentence-BERT) and all-MiniLM-L6-v2 [7] (Sentence-BERT [8]). The OpenAI's alternative is generally regarded as the default choice, primarily due to the performance and the increased maximum input token count of 8k tokens, despite being a paid API solution. The other two Sentence BERT derived models are open-source alternatives, with the observed performance of MpNET-v2 generally being on par with the text-embedding-ada-002 [9], yet the maximum token input length is capped at 1k tokens.

Vector databases and similarity search

The vector database is the main component in the document retrieval workflow, as the user queries ultimately resort to search for the relevant context within the documents. In the previous section we demonstrated how each document snippet extracted from the original document is assigned a vector representation. This vector representation is optimised to make semantically similar pieces of text appear mathematically similar. Vector databases are specialised databases for handling high-dimensional vector data while efficiently allowing vector searching at scale. This technology greatly improves the efficiency of the RAG system as we can now narrow search results to the closest vectors (and therefore the semantically closest equivalent document snippets). The LLM is then presented with these most relevant document snippets, thus considerably reducing the number of tokens the LLM needs to analyse while maximising the relevance of the context of the output to the user query.

Several vector database frameworks are available, including locally hosted open-source options like Facebook AI Similarity Search (FAISS), Milvus and Redis, as well as managed databases provided by Pinecone or Weaviate. For this investigation we used the FAISS (IndexFlatL2) **[10]** local database, prioritising response quality over production scale efficiency compared to alternative databases.

Prompt engineering and model output management

In the process of designing an engineered prompt we identified three main components: task description, context (document snippet) and the user query. The focus of the task description is to inform the LLM of the context it will be provided with, the task it is supposed to accomplish, the output format and what to do in case no answer can be found. However, there is a balance to strike – the challenge is finding the right length of task instructions to give the model without overwhelming it, as excessively long described tasks can make the LLM output deviate from the intended scopes of accurate and consistent results. Generally, the instruction sentences need to be concise, with each new important instruction preferably placed on a new row, as observations showed that long and complex instructions tend to not be fully taken into consideration.

Our previous work **[11]** revealed that one-shot and few-shot notably improves LLM performance on information extraction tasks, specifically topic and sentiment. We have performed the following prompt engineering experiments:

– The first application of prompting helps us quantify the LLM's ability to understand each piece of context. The scope of the prompting method required a yes/no answer from the LLM if the query can be confidently answered from the given context. This methodology was used to analyse the LLM's context comprehension ability, as shown in **Experiment 2** below.
– The second prompting methodology builds on the previous one to direct the LLM to generate final answers from the given context and format the output accordingly. In **Experiment 3** below we analyse the answer quality across multiple domain specific tasks.

At the end of both prompts, we included a reminder to the LLM to be aware of the context provided and to inform the user if a reliable answer could not be provided. This has been found to help greatly in maintaining a consistent output structure.

LLM agent tools

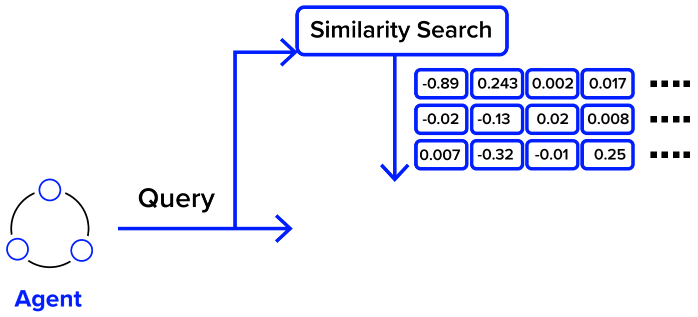
Having decided on the design choices behind the system, we need to combine them in a packaged solution – i.e., a retrieval-augmented generation tool **[12][13].** While this tool can function independently as a simple question-answering product, it lacks several components necessary for a satisfactory user experience. The proposed retrieval-augmented generation tool is best employed as one component of an LLM agent, which enhances the workflow by incorporating additional relevant components into an integrated AI system – such as reasoning components (i.e., memory, reasoning logic) – and allowing connectivity to external data sources. For a system based on a RAG tool, an LLM agent can be used to expand the interaction into a chat-like system that allows for a seamless user interaction by including a plan-and-execute component that decides the necessary chaining of steps and actions and other components such as conversational memory.

Including such an interactivity within the end-product facilitates an efficient dialogue between the user and the LLM, significantly increasing the quality of the overall experience and customer workflow effectiveness. We will explore some of these aspects in upcoming work.

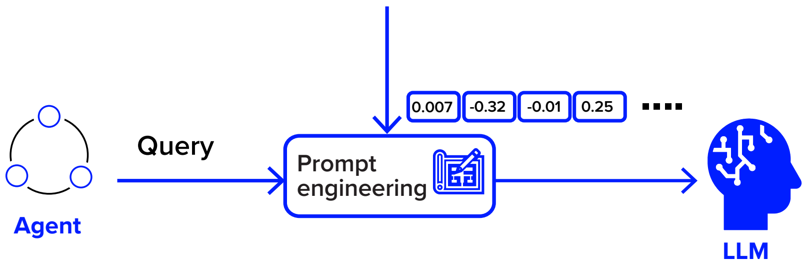
System performance and observations

To evaluate the performance of the retrieval-augmented generation system, we undertook a systematic examination of each of the key components and structured the analysis into three experiments.

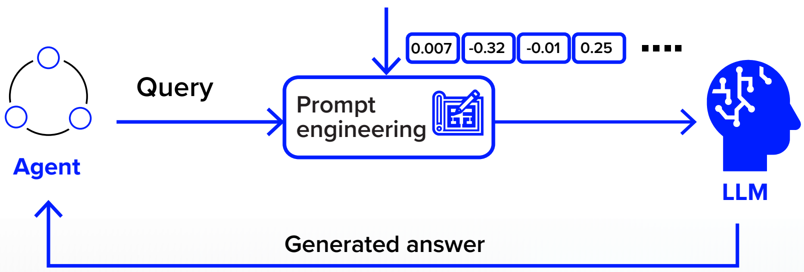
- **Experiment 1 – Similarity search:** this first step within the system has as substantial influence on the task's overall efficiency. In this part we explore the optimal number of document snippets to be returned as context for the LLM.



- **Experiment 2 – Context comprehension:** we determine how well the model understands the context within the identified snippets. We ask the model to answer with yes/no as to whether sufficient information is available in the context to answer a question and measure the success rate.



- **Experiment 3 – Answer quality:** finally, for tasks such as tabular data extraction, key values extraction, textual information, etc., we request the LLM to generate a response/final answer, which we assess both quantitatively and qualitatively.



Source: LSEG, Nov 2023

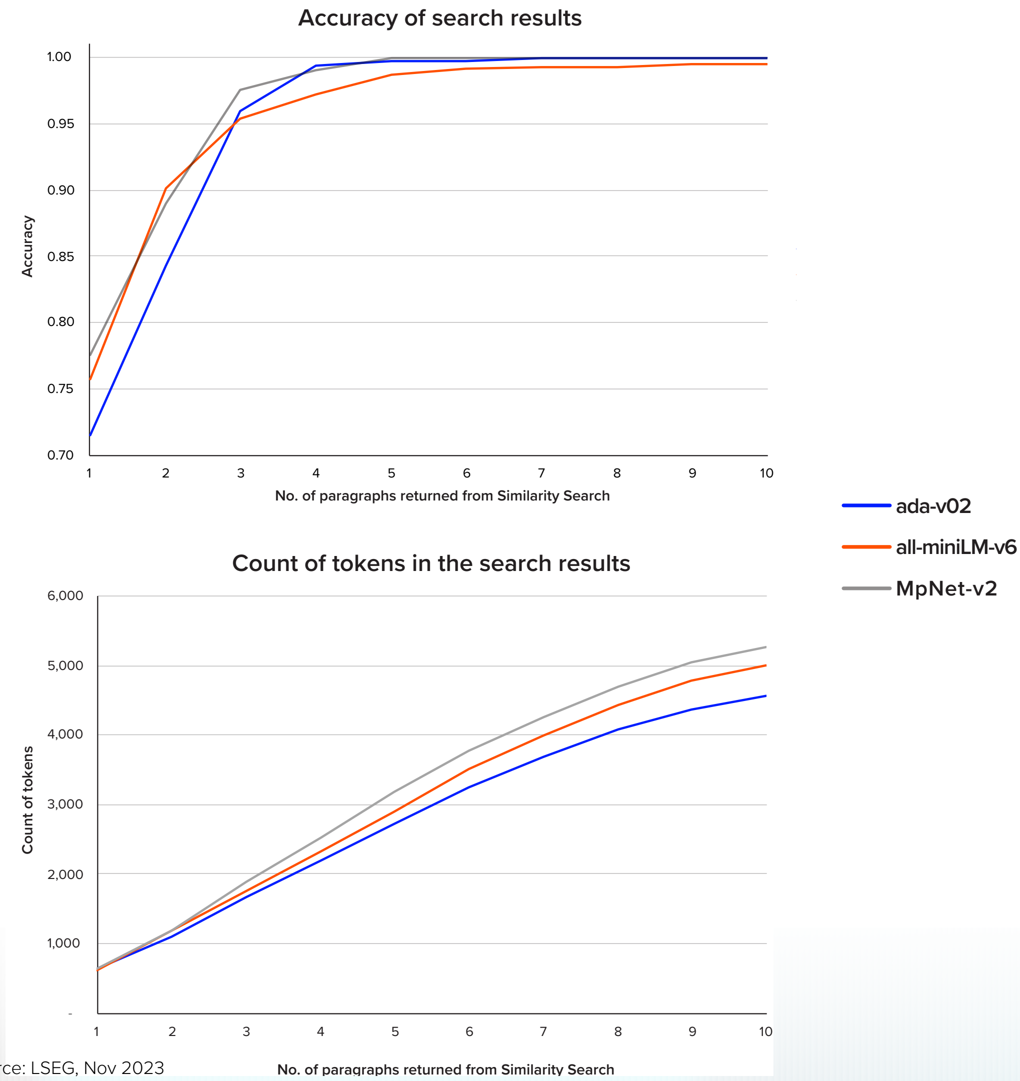
Experiment 1: Similarity search

We aim to identify the optimal quantity of retrieved responses (document snippets) to ensure with a high degree of confidence that the relevant information is located within the provided context. This is driven by both cost and latency considerations: if we can achieve the same results while having to retrieve fewer snippets, then the cost of running the system will be lower and the LLM can produce a response faster. We manually label the document snippets containing the relevant information for each question. These labelled snippets are then compared with the results derived from the similarity search, utilising cosine similarity as a vector distance metric on separate databases containing embeddings from three distinct models. The models we compare are the latest embeddings available from OpenAI (text-embedding-ada) and the two best performing SBERT [8] embedding models (MpNET, all-miniLM-v6).

We find that: text-embedding-ada and MpNET perform best for our use case – as shown in Figure 4. They confidently contain the correct information within the first three to five responses. In 94% and 96% of cases the correct answer is retrieved within three responses for text-embedding-ada and MpNET respectively. Within five responses this number rises to 99% for both models.

Therefore, if the required context is not found within the first five database searches, we find that rephrasing the query can assist in locating the correct context without needing to increase the number of returned searches.

Figure 4: Accuracy of similarity search (top) and the generated token counts (bottom) by the number of returned paragraphs



Source: LSEG, Nov 2023

Experiment 2: Context comprehension

In this experiment we aim to evaluate the LLM's ability to discern which retrieved document snippets, contain sufficient and adequate information to offer a reliable answer. A design option for the system can be to directly surface a quotation from the original document if, for certain question types, the generated text is not very reliable. In the next section we shall see that in some cases the response generated may not have a high degree of accuracy. In such situations, it maybe be useful to also provide the customer with a relevant snippet of evidence or even supply a generated answer altogether. By performing the granular analysis in Experiment 3 of matching types of queries with their response quality, we can control the flow to provide the customer with the mostly likely accurate response given the type of query.

To accomplish this, we design a prompt instructing the LLM to provide a straightforward **yes/no** response and asking it to indicate whether the query can be correctly answered based on the given context. If a paragraph supplies only partial information, it is deemed unsuitable. This analysis uses GPT3.5 exclusively. This analysis is less relevant for the long context model version as the answers will by definition always be in the provided text. By employing metrics such as **precision** and **recall** we can gain insights into the proposed method's performance via the following results:

- **Recall:** evaluates the LLM's ability to correctly detect and not omit the relevant context in the documents provided. The model's performance against this metric was **91%** accurate.
- **Precision:** examines the LLM's ability to correctly identify the documents containing relevant context to answer the given query, for which the results show a performance of **76%.** This in turn suggests the presence of noise further down the pipeline – i.e., the information present in an inaccurately labelled document is not sufficiently relevant to generate a final answer.

The model's overall performance demonstrates promising proficiency in recognising relevant information. Although this investigation reveals that context identification alone may introduce some degree of noise and error in subsequent stages of the retrieval augmented generation (RAG) system, we consider it to be within acceptable levels. Improved prompting techniques demonstrate the LLM's ability to disregard this information in the subsequent answer generation process.

We note that this step in the pipeline can be directly used to provide an answer to the customer, i.e. instead of asking the LLM to summarize a set of candidate answers, we can provide a short list of answers taken verbatim from the source. This way, we are essentially using the system as a semantic search instead of RAG based on optimal question-answer performance.

Experiment 3: Answer quality

Having gained insights into the retrieval and context comprehension elements, we seek to build on this knowledge and create a unified answer generation tool. This system is designed to cater to both internal and external use cases, which implies a wide range of interest areas and corresponding tasks.

We separate the investigation into two key tasks that can that can constitute the functionality of to a RAG system:

- **Value extraction:** involves queries that require data from tabular formats and textual values. These may be presented independently, within phrases, or as part of bullet point lists and diagrams.
- **Textual information extraction**: involves queries that necessitate drawing context from paragraphs. These can provide a broader textual explanation of specific elements such as asset details, legal aspects or involved parties.

For these experiments, we have refined the prompt from Experiment 2 to guide the LLM on the scopes of the newly required tasks, provide general information about the context provided and offer instructions to minimise "hallucinations", generate concise answers and ensure correct output formatting. We will present our findings and the key aspects we have observed during testing.

Result quality of value extraction

Our initial focus is on the LLM's proficiency in providing precise responses to queries necessitating one of multiple values. We concentrate on several areas of interest where data is available in both tabular and textual formats. To utilise this data, we formulate a series of queries including tasks such as identifying the specified certificates/assets along with their corresponding data from tabular formats (e.g., name, principal allocation (US$), value (US$), weighted average life (years)) or extracting singular values from lists or other textual formats (e.g., dates of interest (dd/mm/yyyy), coupon (%), issue size (US$)). For performance evaluation we investigate two metrics: **quantity captured** and **accuracy**. The **quantity** captured metric pertains to the number of extracted values while accuracy represents the **accuracy** of extracted responses as illustrated in Table 1.

For assessing the **quantity captured**, our dataset comprises a total of 755 certificates/assets for tabular extraction tasks and 70 singular values for textual extraction tasks. Generally, GPT4 exhibits near-perfect performance in less complex value identification and extraction tasks, while GPT3.5's performance is up to 10% worse. However, when examining more complex queries, such as those requiring financial details based on specific vocabulary or multiple values rather than just asset names from tabular data, both versions of GPT4 show a more evident proficiency: their performance is up to 32% better than that of GPT3.5.

Regarding **accuracy**, GPT4 outperforms GPT3.5 across less complex tasks by 10%. For more intricate queries, however, even GPT4's performance declines. Issues observed include incorrect pairings of names and values, instances of data hallucination and over-extraction of information. Despite these challenges, there is an almost 33% increase in accuracy between the models, indicating that GPT4 is significantly more precise in complex data extraction tasks from tabular structures. We also notice that GPT4-128k's ability to correctly extract information from within complex tabular structures is lower by 4% compared to its counterpart. This could suggest that the turbo version of the GPT may exert similar abilities for simple tasks, yet for more complex scenarios its performance drops.

Table 1: Quantity of extracted values and the accuracy of the extraction process for tabular value extraction tasks

		Quantity			Accuracy		
Format	Tasks	GPT3.5	GPT4	GPT4-128k	GPT3.5	GPT4	GPT4-128k
Tabular	Names	92%	100%	100%	100%	100%	100%
Tabular	Names + Datapoints	67%	99%	99%	42%	75%	71%
Tabular + Text	Indicatives	97%	99%	100%	85%	94%	97%
Text	Relevant Dates	98%	99%	100%	98%	100%	100%
Text	Financial Details	87%	93%	95%	95%	97%	99%

Source: LSEG, Nov 2023

Result quality of textual answers

We perform a qualitative analysis on the answers derived from textual parts of the document, where information can be included in paragraphs, lists or diagrams. To assess the quality of the answers we use **ROUGE** (Recall Oriented Understudy for Gisting Evaluation) metrics **[14]**, a widely-adopted evaluation metric in natural language processing experiments for evaluating machine-generated summaries. Specifically, we employ the **ROUGE-1** metric which measures the overlap of unigrams between the LLM-generated summary and a human-generated reference summary. Further details on the process of comparing answers, along with relevant examples, can be found in Appendix 2. The results presented below in Table 2 showcase the performance of the system by investigating the **answer quality** and the **context inclusion** of the LLM-generated answers, two metrics derived from the ROUGE-1 results to best describe the general performance of generated answers.

Table 2: **Answer quality** and **context inclusion** results for all investigated areas of interest

		Answer Quality			Context Inclusion		
Source	Tasks	GPT3.5	GPT4	GPT4-128k	GPT3.5	GPT4	GPT4-128k
SPC	General Details	82%	89%	82%	89%	94%	91%
SPC	Mortgage Loan Details	97%	97%	89%	94%	93%	85%
SPC	Relevant Parties	93%	96%	96%	95%	95%	90%

Source: LSEG, Nov 2023

The formulated set of queries is designed to encompass a wide range of possible answers, ranging from brief and precise to more detailed and comprehensive, to evaluate the LLM's proficiency in providing accurate information and helpful summaries. We compute the **answer quality** metric which represents the **recall** of the ROUGE analysis combined with our evaluation of our experts against brief and succinct reference summaries. Conversely, the final answer was generated from a much larger piece of context, thus the **context inclusion** metric represents the **precision** of the ROUGE analysis where the reference summary comprises all context information. This metric indicates the percentage of the generated answer that incorporates raw context information relative to LLM-generated wording, which shows how verbose the LLM's answer is. Generally, we observe superior performance across both metrics when using GPT4, which offers more robust and predictable answers, in accordance with the prompt specifications.

The first notable feature of the results is the strong performance of GPT3.5 across the areas of interest examined, which raises the question of the optimal balance between using the two models to maximise performance while minimising costs. When comparing the performance of the models, it is immediately apparent that GPT4 exhibits superior performance across the examined tasks, suggesting it as the preferred model, providing up to 7% improvement on answer quality. On the other hand, both models demonstrate comparable results in comprehending the information in the context and incorporating it appropriately in the answer. This suggests that an efficient system could leverage both models, with GPT3.5 performing the primary extraction which requires significant API usage while GPT4 could formulate the final answer.

Secondly, we notice a slight decrease in the overall performance of GPT4-128k compared to its counterpart, with the difference for some investigated areas being as high as 8%. These differences may arise due to poorer understanding of the prompt instructions, introduction of unnecessary wordings or context misunderstanding given the whole document is used as a context with this model.

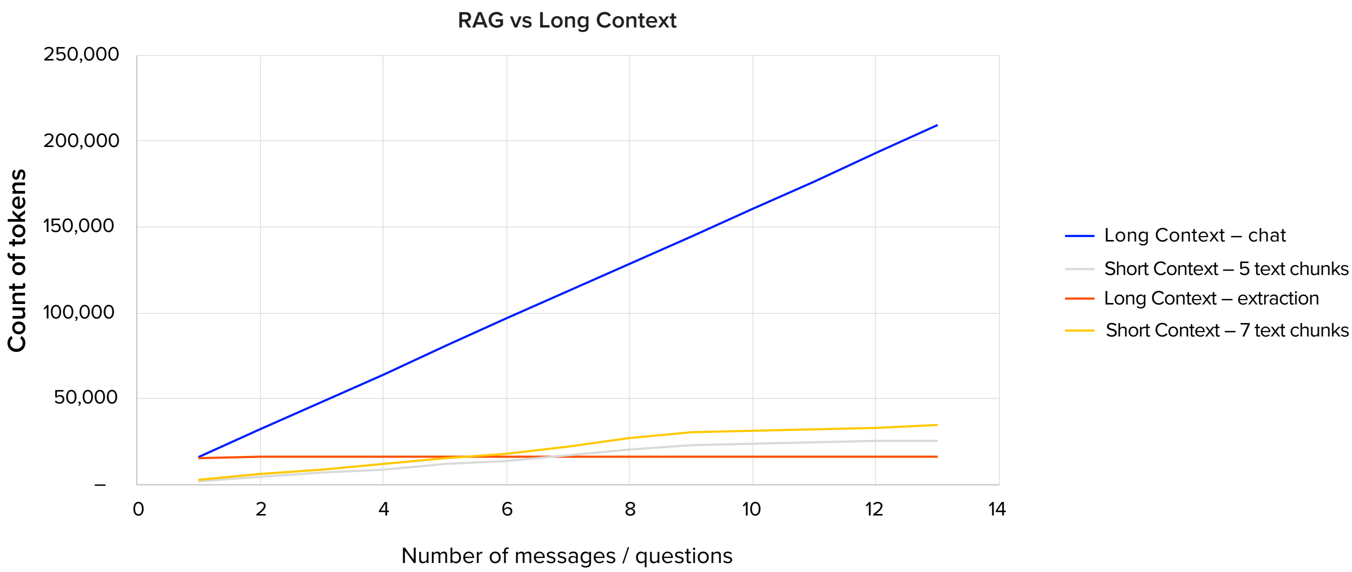
In summary, both GPT3.5 and GPT4 demonstrate proficiency in extracting the necessary information from the context provided, with GPT4's performance being notably more consistent and reliable. While the exclusive use of GPT4 for a system may currently seem inefficient in the long term, the existing LLM landscape is rapidly evolving and with new alternatives continually emerging **[15]** we can anticipate changes in the overall costs and capabilities of the models we tested.

Scalability and cost considerations - long vs short context

The ability to use a longer context creates a number of trade-off considerations. Let us consider two major scenarios for the use of the RAG tool we are building: **information extraction** and **chat**.

For **information extraction**, we know a set of questions we want to ask in advance for a large number of documents and we just want to do this at scale. In the long context case, given long enough context, we can actually ask all questions at one go and save ourselves iteratively sending context to GPT. We will simply send the full context and the full list of questions to the LLM and expect all answers back. In the case of a short context window all answers will simply not be included in a small number of text chunks, so we need to iteratively send the context snippets associated to each question for the answer generation process. However, the individual token length for each exchange is much shorter. In our previous experiments we experimented with using between 5 to 7 context snippets for each question. You can observe in Figure 5 below that given the requirements of the extraction task at hand (i.e., quantity of documents, amount of values to be extracted) there exists a trade off point at which the long context extraction method (orange line) becomes more economical and scalable than the short context/RAG methodology (grey and yellow lines).

Figure 5: Costs of RAG and Long Context methods for different task scenarios (chat and information extraction)



Source: LSEG, Nov 2023

For a **chat** use case, questions are not clear in advance and will have a unique chat history each time. This aspect makes the long context scenario much less appealing as we need to send all the context back to the LLM with each question. The set up for short context is similar to the information extraction use case, thus making the method a more appealing solution. In both cases the token length grows, yet at considerable different rates, with the short context performance clearly demonstrating its usefulness in a chat-like system. In Figure 5 we can observe the explosion of cost demonstrated by the long context (blue line) on the chart vs the short context ones.

Both methods are equally valid and applicable for dealing with large documents, yet the short context approach methodology still proves to be the better of the two worlds.

Further work towards an LLM-powered product

We have demonstrated the system's strong performance in responding to targeted questions in a specialised domain. With more customer and SME feedback we will continue to use this framework to evaluate and maintain performance. Further exploration can involve alternative data processing techniques (e.g., document splitting, embedding or vector storage) and the use of better similarity search techniques. Other experiments can include supplying pertinent documentation terminology along with corresponding definitions. One crucial enhancement moving forward is to expand our search ability to accommodate the necessity of generating comparative results between multiple documents within the same asset class.

To provide a more integrated workflow experience, in future work we will investigate the use of **conversational** agents. This approach would upgrade the existing question-answering product to a chat-like system capable of incorporating previous conversational context into current actions and collaborating with the user to achieve optimal results. Agents are in essence, systems combining multiple data processing tools, chains (i.e., retrieval-generation), LLMs and prompting techniques (e.g., chain-of-thought, ReAct, etc.) [16]. [17]

Ideally within this context an agent can perform a chain of actions, identifying specific tools necessary to answer user queries (like a similarity search) while preserving the conversation topic within the established guidelines, thereby maintaining a natural and engaging conversation flow.

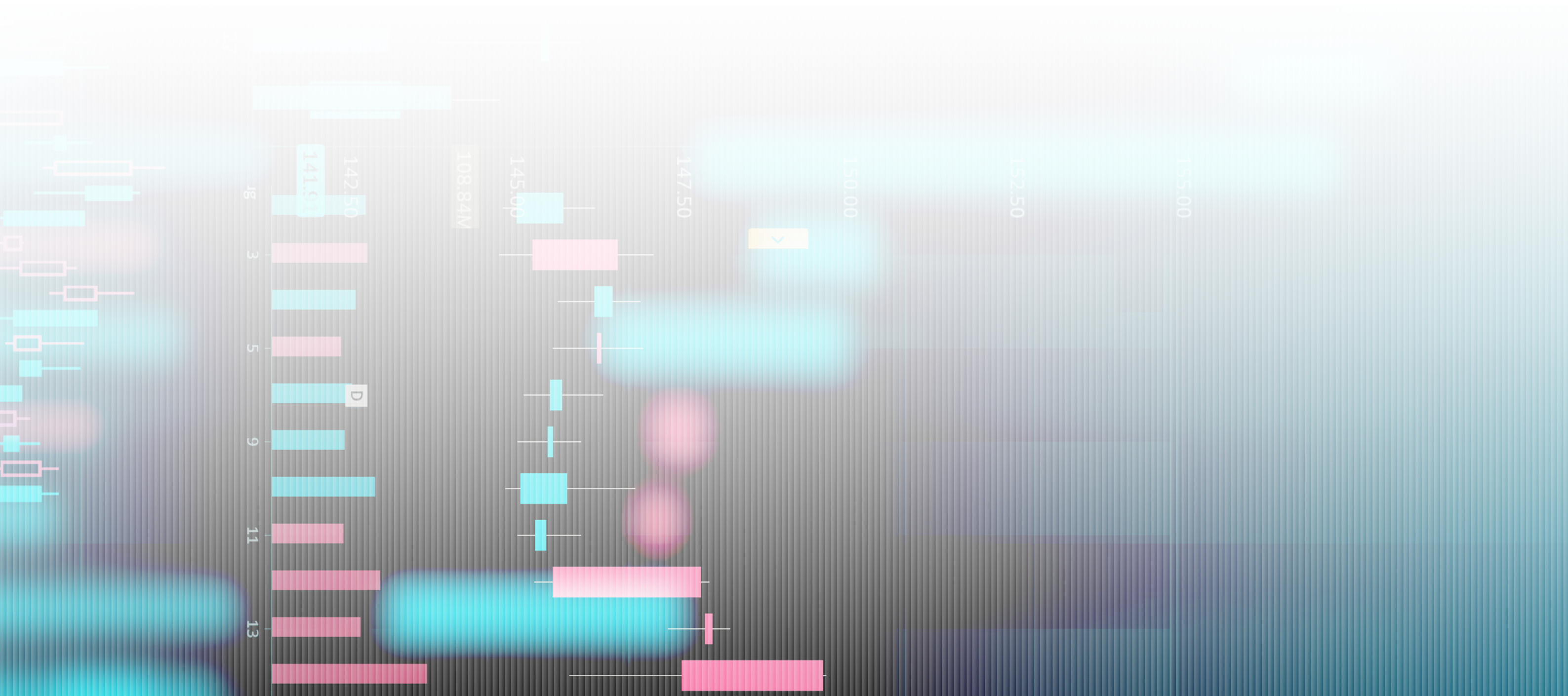
To close the gap between a question-answering system and a final product designed to meet financial services-related needs, additional elements must be considered:

- **Further choices of LLMs**: with new LLMs appearing constantly, we can keep adding more choices to the generational component of the system such as Llama2, which has been found to match GPT3.5's performance across a wide range of tasks [18], but for now, GPT4 remains state-of-the-art in most cases despite having some marked shortcomings.
  - **Fine-tuning LLMs:** the fine-tuning of an already existing LLM GPT3.5, Llama2 etc. will likely enhance the performance of the RAG-based system. In such a case a model will be fine-tuned to answer financial domain questions, equipping it with plenty of contextual key finance-specific terminology and ultimately promoting a more streamlined interaction with the end user.
  - **Pre-training LLMs:** meaning the development from scratch of an LLM focused on finance specific terminology, knowledge bases and tailored use cases. While this will be much more resource-intensive, the resulting model will support a wide range of LLM use cases across asset classes and generative AI applications.
- **Productisation efficiency:** the current experiments focus on accuracy, not latency or scale. In a production environment there are numerous optimisation choices to be made, such as the most reliable/scalable vector database, caching of questions and answers, etc. For instance, it is conceivable that many of the questions customers ask about a deal will be highly repeatable but with slight variations in language. In such scenarios, a lot of the components of the proposed system will be able to cache and reuse results to reduce both latency and cost of the overall system.
- **UX choices**: in the realm of user experience (UX) design, several criticisms have emerged concerning the chat interface of LLM-based products [19]. The conventional chatbot interaction, which often involves full-length questions, may not always be the most effective approach, especially in scenarios where the subsequent action can be accurately anticipated. Experimenting with appropriate suggestions, predefined structured extractions or dynamically prompting the user with buttons for anticipated actions all can provide a more streamlined interface. This area certainly warrants further exploration and experimentation.

Conclusion

We have analysed the impact of different designs and setups on a retrieval augmented system (RAG) for performing investment due diligence on documentation relating to different financial instruments. Such a system will likely be an integral reasoning component of LLM agents' design and the overall AI-powered experience for our customers. Current experimentation shows promising results both in terms of identifying the right context and extracting the relevant information. This in turn suggests that the RAG system is a viable tool for an LLM conversational agent to access if the user needs to extract specific deal definitions from extensive financial documentation.

In conclusion, the results of these investigations give us a solid foundation to inform the future design of LLM question-answering tools. However, we recognise that effective retrieval and generation is just one part of the design of a fully integrated conversational flow. LLM agents will likely use a set of such tools to understand and contextualise a range of customer needs and the right UX methods will play a crucial part in producing a timely and informative financial due diligence experience for our customers.



References

1. Liu, N.F. et al. (2023) Lost in the middle: How language models use long contexts, arXiv.org. Available at: https://arxiv.org/abs/2307.03172

2. K-deals® (2023) Multifamily – Freddie Mac. Available at: https://mf.freddiemac.com/investors/k-deals

3. OpenAI platform – Models. Available at: https://platform.openai.com/docs/models/gpt-4

4. Abacusai. GitHub. Available at: https://github.com/abacusai/Long-Context

5. OpenAI platform – Embeddings. Available at: https://platform.openai.com/docs/guides/embeddings

6. all-mpnet-base-v2. Sentence Transformers. Available at: https://huggingface.co/sentence-transformers/all-mpnet-base-v2

7. all-MiniLM-L6-v2. Sentence Transformers. Available at: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

8. sBERT. Sentence Transformers. Available at: https://www.sbert.net/

9. Ofori, D. (2023) GPT-3 vs other text embeddings techniques for text classification: A performance evaluation, Medium. Available at: https://derrickofori015.medium.com/gpt-3-vs-other-text-embeddings-techniques-for-text-classification-a-performance-evaluation-b3a3e6e84cb7

10. Introduction to Facebook AI Similarity Search (FAISS). Pinecone. Available at: https://www.pinecone.io/learn/series/faiss/faiss-tutorial/

11. Oliver, D. et al. (2023) Using GPT-4 with prompt engineering for financial industry tasks, LSEG Analytics. Available at: https://solutions.yieldbook.com/content/dam/yieldbook/en_us/documents/publications/using-chatgpt-with-prompt-engineering.pdf

12. Tomaz B. (2023) Knowledge graphs & LLMS: Fine-tuning vs. retrieval-augmented generation, Graph Database & Analytics. Available at: https://neo4j.com/developer-blog/fine-tuning-retrieval-augmented-generation/

13. Martineau, K. (2023) What is retrieval-augmented generation?, IBM Research Blog. Available at: https://research.ibm.com/blog/retrieval-augmented-generation-RAG

14. Lin, C.-Y. (2004) Rouge: A package for automatic evaluation of summaries, ACL Anthology. Available at: https://aclanthology.org/W04-1013/

15. Nvidia and hugging face to connect millions of developers to Generative AI supercomputing, NVIDIA Newsroom. Available at: https://nvidianews.nvidia.com/news/nvidia-and-hugging-face-to-connect-millions-of-developers-to-generative-ai-supercomputing

16. Wei, J. et al. (2023) Chain-of-thought prompting elicits reasoning in large language models, arXiv.org. Available at: https://arxiv.org/abs/2201.11903

17. Yao, S. et al. (2023) React: Synergizing reasoning and acting in language models, arXiv.org. Available at: https://arxiv.org/abs/2210.03629

18. Llama 2. Meta AI. Available at: https://ai.meta.com/llama/

19. The user experience of Chatbots, Nielsen Norman Group. Available at: https://www.nngroup.com/articles/chatbots/

Appendix 1 – List of queries used in investigations

Source	Query	Area of Interest	Type of data structure
Structured Pass-Through Certificates Structured Notes	Which are the (non) offered certificates (classes)?; What are the values of principal allocation and weighted average life of the (non) offered certificates (classes)? What are the assets and their associated initial price/ reference level?	Asset Names and Data	Tabular
Structured Pass-Through Certificates Structured Notes	What is the determination date?; What is the distribution of principal/interest payments date? When are the initial fixing and maturity dates (provide the answer in the format DD/MM/YYYY);	Relevant Dates	Textual (one-off values; paragraphs; lists)
Structured Notes	What is the ISIN code?; What are the Bloomberg tickers of the underlying securities?; What is the governing law / jurisdiction?	Indicatives	Textual (one-off values; lists)
Structured Notes	What is the issue size (capitalisation)?; What is the issue currency?; What is the nominal amount (denomination)?; What is the coupon value?	Financial Details	Textual (paragraphs; lists)
Structured Pass-Through Certificates	How are the losses on the underlying mortgage loans allocated to the certificates? What is the “Freddie Guarantee”, how does it work and what are the “Guaranteed” certificates?	Offered Certificates	Textual (paragraphs; lists)
Structured Pass-Through Certificates	Who is the Master Servicer?; Who is the Special Servicer?; Who are the Trustee and Custodian?	Relevant Parties	Textual (paragraphs; lists)
Structured Pass-Through Certificates	What are the main geographical regions that contain mortgages?; What sort of property types do the mortgages cover?	Underlying Mortgage Loans	Textual (lists; diagrams; paragraphs)

Appendix 2 – Candidate and reference summaries examples

Assessment methodology

The terms "candidate summary (C)" and "reference summary (R)" are used. Within the scope of this experiment, we can consider the output generated by the LLM to be a summary as the question-answering system analyses the entire content of a page/paragraph/section and will summarise it according to what the user has requested.
Candidate summary (C) is the answer that the LLM system has generated.

Reference summary (R) is an acceptable answer to the question (answer quality) or the whole used context (context inclusion).

Example 1: **Who is the master servicer?**

Reference summary (whole context):

Master Servicer and Special Servicer .... KeyBank National Association, a national banking association organized under the laws of the United States of America ("KeyBank"), will act as the master servicer and the special servicer with respect to the underlying mortgage loans. KeyBank will also act as the Affiliated Borrower Loan Directing Certificateholder with respect to Affiliated Borrower Loans that are not Affiliated Borrower Special Servicer Loans and may, if requested, act as the Directing Certificateholder Servicing Consultant. KeyBank also originated 4 of the underlying mortgage loans, collectively representing 7.0% of the initial mortgage pool balance, and will service the underlying mortgage loans it originated. KeyBank is not

Source: Freddie Mac Structured Pass-Through Certificates (SPCs) Series K-100

Candidate summary::

GPT3.5:

The Master Servicer is KeyBank National Association. It is responsible for collecting all payments due under the terms of the underlying mortgage loans and depositing them in the collection account daily. It also acts as the Affiliated Borrower Loan Directing Certificate holder and may act as the Directing Certificate holder Servicing Consultant if requested. Additionally, KeyBank originated 4 of the underlying mortgage loans and will service the loans it originated.

GPT4:

The Master Servicer is KeyBank National Association. It is responsible for making reasonable efforts to collect all payments due under the terms and provisions of the underlying mortgage loans. These payments are deposited in the collection account daily. KeyBank also acts as the Affiliated Borrower Loan Directing Certificate holder with respect to Affiliated Borrower Loans that are not Affiliated Borrower Special Servicer Loans and may, if requested, act as the Directing Certificate holder Servicing Consultant. KeyBank also originated some of the underlying mortgage loans and will service those it originated. As compensation for servicing the underlying mortgage loans, the master servicer receives a master servicing fee, a sub-servicing fee, and a master servicer surveillance fee. The master servicer is also entitled to additional servicing compensation in the form of borrower-paid fees. If KeyBank resigns or is terminated as master servicer, it will be entitled to retain a portion of interest on each underlying mortgage loan referred to as the "Excess Servicing Strip."

Reference summary (acceptable answer): KeyBank National Association

Example 2: What is the geographical concentration of the mortgages included in this deal?

Reference summary (whole context):

Geographic Concentration......................

Mortgaged real properties that secure underlying mortgage loans collectively representing 5.0% or more of the initial mortgage pool balance are located in each of the states listed in the table below. The table below shows the number of, and percentage of the initial mortgage pool balance secured by, mortgaged real properties located in these states:

State	Number of Mortgaged Real Properties	% of Initial Mortgage Pool Balance
California	7	23.3%
Arizona.........................	6	15.3%
Texas	8	13.1%
Maryland......................	1	11.2%
New Hampshire...........	1	7.1%
Washington	2	6.4%
Minnesota.....................	1	5.5%

The remaining mortgaged real properties are located throughout 10 other states. No more than 4.5% of the initial mortgage pool balance is secured by mortgaged real properties located in any of these other states.

3 of the California mortgaged real properties, securing underlying mortgage loans collectively representing 11.7% of the initial mortgage pool balance, are located in southern California (*i.e.*, addresses with zip codes of 93600 or below). 4 of the California mortgaged real properties, securing underlying mortgage loans collectively representing 11.6% of

Source: Freddie Mac Structured Pass-Through Certificates (SPCs) Series K-062

Candidate summary:

GPT3.5:	**GPT4 answer:**
California: 7 properties (23.3%)	California: 23.3%
Arizona: 6 properties (15.3%)	Arizona: 15.3%
Texas: 8 properties (13.1%)	Texas: 13.1%
Maryland: 1 property (11.2%)	Maryland: 11.2%
New Hampshire: 1 property (7.1%)	New Hampshire: 7.1%
Washington: 2 properties (6.4%)	Washington: 6.4%
Minnesota: 1 property (5.5%)	Minnesota: 5.5%

Reference summary (acceptable answer):

California, Arizona, Texas, Maryland, New Hampshire, Washington, Minnesota

Example 3: **How are the losses on the underlying mortgage loans allocated to the certificates?**

Reference summary (whole context):

Reductions of Certificate Principal Balances in Connection with Losses and Expenses .... As and to the extent described under "Description of the Certificates—Reductions of Certificate Principal Balances in Connection with Realized Losses and Additional Issuing Entity Expenses" in this information circular, losses on, and default-related or other unanticipated issuing entity expenses attributable to, the underlying mortgage loans will, in general, be allocated on each distribution date, after making distributions on such distribution date, to reduce the outstanding principal balances of the Principal Balance Certificates, sequentially, in the following order:

Reduction Order	Class
1st	Class C certificates
2nd	Class B certificates
3rd	Class A certificates

Any reduction of the outstanding principal balances of the Principal Balance Certificates will also result in a corresponding reduction in the notional amount of the corresponding component of the class XI certificates.

Source: Freddie Mac Structured Pass-Through Certificates (SPCs) Series K-062

Candidate summary:

GPT3.5:

Losses on the underlying mortgage loans are allocated to the certificates in the following order:

1. Class C certificates
2. Class B certificates
3. Class A certificates

GPT4:

Losses on the underlying mortgage loans are allocated to the certificates on each distribution date, after making distributions on such distribution date, to reduce the outstanding principal balances of the Principal Balance Certificates, sequentially, in the following order:

1. Class C certificates
2. Class B certificates
3. Class A certificates

Reference summary (acceptable answer):
1) Class C; 2) Class B; 3) Class A

Acknowledgements

Authors:

Mihail Dungarov, CFA

Product Management Lead, Text Analytics

Giusca, Iulian
Senior Analyst Product Development

Elefteriu, Bogdan-Mihai
Data Scientist, Product

Thanks to Tadvana Narayanan, Sebastian Genet, Aran Batth, Rani Shlivinski, Aysegul Erdem, David Oliver, Luca-Alexandru Popescu, Paul Jackson and Connor Weimer.

LSG3286353/12-23