LUND UNIVERSITY

SCHOOL OF ECONOMICS AND MANAGEMENT

Leveraging Large Language Models for Firm-Intelligence:
A RAG Framework Approach.

By
Niclas Wölner-Hanssen

January 2024

Master's Thesis in Statistics

Supervisor: Jonas Wallin

Acknowledgement

During my five years at Lund University, I had the pleasure of being a part of the Trading & Quantitative Research department (TQR) within the student organization LINC for three of those years. In addition to the insights gained from the research projects I conducted on behalf of TQR in collaboration with Lynx Asset Management and OQAM, I was fortunate to also partake in internships at these firms, further exploring my interest in systematic investing. I regard my time at TQR as equally important to the skillset I have acquired over these years as the academic part. I would therefore like to extend my thanks to all the previous members of TQR/LINC who made TQR possible. I am also grateful to all the amazing people I have met through the organization over the years. Special thanks go to Thorbjörn Wallentin at OQAM, and Ola Backman and Eric Lundgren at Lynx. Your mentorship, dedication, and inspiration have been invaluable, not only to me but to all students fortunate enough to work with you. You will always hold a special place in my story.

Abstract

In the wake of OpenAI's release of ChatGPT in November 2022, powered by the 175 billion parameter neural network GPT-3, the potential applications of Large Language Models (LLMs) in various sectors have become evident. One such application lies in hedge funds and trading desks where knowledge sharing is paramount. These entities often possess a wealth of firm-specific knowledge that spans different research areas and personnel expertise. Leveraging LLMs on this knowledge is challenging due to its proprietary nature, the immense data and computational demands of training LLMs, and the inherent limitations of LLMs, such as the tendency to fabricate facts. The Retrieval Augmented Generation (RAG) framework, which has recently gained traction (Shi et al., 2023), presents a solution. This thesis explores the potential of creating a firm intelligence unit using the RAG framework, leveraging research reports from Lund University Finance Society's Trading & Quantitative Research (TQR) department as a representative dataset. The envisioned AI Assistant aims to answer questions based on the TQR reports, admit ignorance when necessary, and provide detailed answer sources. This study provides insights into the theory behind LLMs and the implementation of the RAG framework and offers a comprehensive evaluation, discussing results, limitations, and future prospects for firm intelligence units.

Contents

1. Introduction

In November 2022, OpenAI released ChatGPT to the public. This chatbot demonstrated an impressive ability to generate human-like text that could solve a wide range of tasks simply by receiving instructions. The Large Language Model (LLM) behind ChatGPT, GPT-3, a 175 billion parameter neural network, had been trained on a vast amount of text from the internet, which allowed it to implicitly store significant world knowledge in its parameters. GPT-3 was then fine-tuned using reinforcement learning from human feedback, and the result was a chatbot that reached 100 million users just two months after its launch.

Knowledge sharing between researchers, departments, and desks is vital for hedge funds and trading desks. Researchers, Portfolio Managers, Traders, etc., exhibit different levels of domain knowledge and experience. Moreover, past research might have been forgotten. Much of this firm knowledge might have been documented, and the recent development of LLMs allows for the aggregation of this knowledge into a central firm intelligence unit. This firm-intelligence unit could then have various applications, for example:

- When a new project is initiated, the researcher is up to speed with previous research on the topic.
- The firm-intelligence unit could provide thorough research assistance based on firm knowledge.
- The firm-intelligence unit could potentially draw conclusions between projects that the human researchers have overlooked.

This firm-intelligence unit has potentially many more use cases. However, this firm knowledge is not available to the public. Thus, no LLM has been trained on this knowledge. Moreover, fine-tuning (updating the weights) these 100B parameter neural network models on new data requires significant data and computational resources. Furthermore, LLMs like GPT-3 have additional limitations, such as a propensity for "*hallucination*", i.e., to make up facts. Moreover, they have an inability to represent the full scope of the knowledge from their training data. To overcome these issues and to make it possible to leverage LLMs on custom data for individuals and businesses, Retrieval Augmented Generation (RAG) is a framework that has become popular in recent months (Shi et al., 2023).

1.1 LINC & TQR

Lund University Finance Society (LINC) is a student organization at Lund University founded in 1991 (LINC, 2023a). Within LINC, under the department Trading & Quantitative Research (TQR), students at Lund University have since 2018 conducted research projects in quantitative finance (LINC, 2023a). Since 2019, these projects have been conducted in collaboration with Swedish quantitative hedge funds such as Lynx Asset Management, OQAM, and ValidAlpha. These research projects, with topics ranging from machine learning applications to risk management etc., are similar in nature to the projects that are being conducted on-site of these hedge funds. For this reason, the TQR reports act as a good toy example of the actual research material on hedge funds and research departments.

1.2 Thesis Objective

The main objective of this thesis is to construct what could be considered the embryo of the firm intelligence unit discussed in the previous section using the TQR reports. Specifically, the objectives are as follows:

1. To construct a question-answering (QA) AI-Assistant based on the RAG framework using the TQR reports.

2. If the TQR reports do not have the information needed to answer the question asked by the user, the AI-Assistant clearly states that it does not know and does not make up facts.

1.3 Thesis Outline

Section 2 – Theory: In section 2, the theoretical concepts in which the AI-Assistant is built on will be presented. Starting with a brief discussion of the differences between decoder-only and encoder-only transformers to recognize the two main LLM components under the hood of the AI-Assistant, one text embedding model (encoder-only transformer) and one text generating LLM (decoder-only transformer). Then, a brief historical progression of the GPT-series and LLMs will be outlined, followed by a description of the RAG framework and the text embedding model GTE as well as the evaluation framework RAGAS.

Section 3 – Method & Implementation: This section details how the theoretical concepts outlined in section 2 are implemented to construct the AI-Assitant.

Section 4 – Results: In this section, the AI-Assistant is evaluated according to the RAGAS framework discussed in section 2 using different RAG-configurations. Specifically, two different prompt instructions are tested. Moreover, GPT-3.5 vs. GPT-4 is tested as the answer-generating. Finally, the embedding model GTE discussed in section 2 is compared to the black-box text-embedding-ada-002 model. Upon presenting the results, a discussion of the results will follow.

Section 5 – Conclusion and Future Work: In this section, conclusions are drawn, and future work for improving RAG systems is discussed, including the implementation of Agents.

2. Theory

In this section, the theoretical underpinnings of the AI-Assistant are explored, starting with a brief comparison of decoder-only and encoder-only transformers. The section then traces the development of the GPT series and LLMs**.** Furthermore, the RAG framework**,** the text embedding model GTE, and details **of** the RAGA**S** evaluation framework are presented**.**

2.1 Transformers

Since their debut in (Vaswani et al., 2017), transformer models have set new benchmarks in performance across a wide array of Natural Language Processing (NLP). As outlined in (Vaswani et al., 2017), the original architecture was conceived as a denoising autoencoder, and its design was articulated through the components of an encoder and a decoder. In the time since, the landscape has evolved significantly with transformer variants, such as those detailed in (Radford et al., 2018), who presented the first GPT model, and (Devlin et al., 2018), who presented BERT. Notably, these newer models diverged from the autoencoder framework that characterized the original transformer in (Vaswani et al., 2017). To facilitate clarity and communication within the research community, the authors of these variants have often framed their work using terminology that harks back to the original transformer's autoencoder roots as described by (Vaswani et al., 2017). For instance, in the context of GPT, the model is designated as a 'decoder-only' transformer. Conversely, the team behind BERT labels their innovation as an 'encoder-only' structure.

2.2 Decoder-Only and Encoder-Only Transformers

Although quite similar architectures, there are some important differences between the Decoder-only and Encoder-only transformers. As the name suggests, the encoder-only transformers consist solely of an encoder part composed of a stack of multiple identical encoder-layers. Likewise, the decoder-only model consists of multiple stacked decoder-layers. Both encoder-only and decoder-only transformers use positional encodings to inject information about the position of tokens in the sequence, and the mechanism for this is generally the same in both types of models. Likewise, token-embedding techniques tend to be similar, GPT uses Byte-Pair Encoding (BPE) and BERT uses WordPiece. Each layer in both models has two main sub-layers: a multi-head self-attention layer and a fully connected feed-forward network. The main difference at this point lies in the masking of the self-attention mechanism. The objective for the decoder-only transformers, like GPT, is to predict the next word, (x_{t+1}) in a sequence of words, token-by-token, in an auto-regressive manner. For this reason, the attention scores for the words in the sequence that the model has not yet predicted must be masked. On the other hand, for encoder-only transformers like BERT the objective is to contextualize a text sequence into a *text embedding,* with the purpose to be used for various downstream tasks. Here, the bidirectional context of words in the sentences is important; therefore, no masking of the attention scores is required.

2.3 Decoder-Only Transformers - GPT-series

As will be discussed in further detail in section 2.4, under the RAG framework and central to the AI-Assistant's ability to produce human-like text is the GPT-3.5 model. The development that led up to GPT-3.5 and the progression from Language Models (LMs) to Large Language Models (LLMs) will be presented in subsequent sections.

2.3.1 GPT-1 Generic Language Models

A common practice in NLP tasks is to acquire some text data, label that data, and then train a model for a specific task, for example, in a spam-mail/no-spam classification problem. One would first acquire a bunch of emails, then label these as either spam or no spam, and then train a model based on this data. This same model & data would not be particularly useful for other NLP tasks, such as summarization. Since labeled data is scarce, but unlabeled text data is abundant, the authors in (Radford et al., 2018) proposed *generative pre-training* (GPT). The ambition was first to learn a high-capacity language model trained unsupervised on unlabeled text data, which could then be fine-tuned on downstream tasks. Given an unlabeled corpus of tokens:

$$x = \{x_1, \dots, x_n\},$$

the authors in (Radford et al., 2018) used a standard language modeling objective to maximize the following likelihood:

$$L(x;\theta) = \sum_i log\ P(x_i \mid x_{i-k}, \dots, x_{i-1};\ \theta)\ ,$$

where k is the size of the context window (preceding tokens), and θ are the parameters in the neural network. As such, the GPT series was born and is simply a language model (LM) with the objective of predicting the next word trained on unlabeled data with the ambition of being a task-agnostic language model.

2.3.2 GPT-2 - LM's are Multitask Learners

In (Radford et al., 2019) the language model objective stayed the same as for GPT-1. The unsupervised pre-training was increased both in terms of model size and data size. In (Radford et al., 2019), four model configurations were trained. Their parameters are shown in Figure 1. The largest model is what is referred to as GPT-2. The smallest model is equivalent to that of GPT-1.

Parameters	Layers	d_{model}
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600

Figure 1 – Hyperparamters GPT-2 (Radford et al., 2019). Parameters: refers to the number of weights in the neural network. Layers: refers to the number of decoder-blocks. d_{model} *size of the hidden layers.*

Besides using larger models, another significant change from GPT-1 to GPT-2 was that the supervised fine-tuning on downstream task idea was dropped. The model was not explicitly trained to solve any particular language task. Still, it performed decently on some tasks compared to other models that have been fine-tuned explicitly on such tasks. The evaluation for GPT-2 was conducted using *zero-shot* inference; that is the model received inputs, i.e *prompts* such as:

"*Translate this sentence to Swedish: <sentence>* "
or
"*Summarize the following document: <document>* "

without having seen examples of correct outputs while being trained. Given these input prompts, the most appropriate output generated from the model should solve the task (e.g., translating the sentence to Swedish or summarizing the document).

2.3.3 GPT-3 - Large Language Models (LLM)

In (Brown et al., 2020) the authors build upon the idea from (Radford et al., 2019) of a task-agnostic language model and what is referred to as *in-context learning*, where a pre-trained language model has developed a broad set of skills and pattern recognition on large amounts of text adapts to new tasks based on instructions or demonstrations given at the time of inference in the prompt.

Following the results in (Kaplan et al., 2020) who showed that performance of LM's scales with model size, data size and training compute, the authors in (Brown et al., 2020) present GPT-3, a 175 Billion parameter LM. The argument for increasing the number of parameters was also more nuanced. Since in-context learning involves absorbing many skills and tasks within the parameters in the model, it is reasonable to believe that the in-context abilities also increase with scale.

GPT-3 follows the same model structure as its predecessors, but the data and model size increased significantly along with the training time. The model size for the GPT-3 series can be seen in Figure 2, and the data specification in Figure 3.

Model Name	n_{params}	n_{layers}	d_{model}
GPT-3 Small	125M	12	768
GPT-3 Medium	350M	24	1024
GPT-3 Large	760M	24	1536
GPT-3 XL	1.3B	24	2048
GPT-3 2.7B	2.7B	32	2560
GPT-3 6.7B	6.7B	32	4096
GPT-3 13B	13.0B	40	5140
GPT-3 175B or "GPT-3"	175.0B	96	12288

Figure 2 – GPT-3 Hyperparameters (Brown et al., 2020). n_{params} *is the total number of trainable parameters in the model,* n_{layers} *is the total number of layers,* d_{model} *size of the hidden layers.*

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Figure 3 – Data specification for GPT-3 (Brown et al., 2020). "Weight in training mix" refers to the fraction of examples during training that are drawn from a given dataset. The model is trained on a total of 300 billion tokens, some datasets are seen up to 3.4 times during training while other datasets are seen less than once.

2.3.4 Evaluating GPT-3 - zero-shot, one-shot and few-shot learning.

As discussed in the previous section the approach of the GPT-3 model was that it is trained on a vast amount of text using an unsupervised auto-regressive next-word prediction approach. It is not trained for any specific NLP task such as sentiment prediction or translation between languages. Rather, the task one wants to conduct is specified at inference time, in the prompt. With this approach, the model is conditioned on a natural language instruction and/or a few examples (demonstrations) of the task, and is then expected to complete further instances of the task by predicting what word comes next based on this context.

In (Brown et al., 2020) the GPT-3 model was evaluated under three such context-learning conditions:

1. "few-shot learning", where multiple examples are provided in the prompt.
2. "one-shot learning", where only one demonstration is allowed.
3. "zero-shot" learning, where no demonstrations are allowed and only an instruction in natural language is given to the model in the prompt.

Examples of these are shown in Figure 4.

GPT-3 was evaluated on several benchmarks for different tasks and scored high or outperformed several SOTA models. However, these models had been extensively fine-tuned on that particular task, making the performance of GPT-3 most impressive. Across benchmarks, GPT-3 showed that LLMs become more effective at task-agnostic, in-context learning as they grow in size. We don't need task-specific data, we can just insert examples of the task we want to solve in the prompt and the LLM generates accurate responses on a variety of tasks, making GPT-3 the first practical example of using general purpose LLMs.

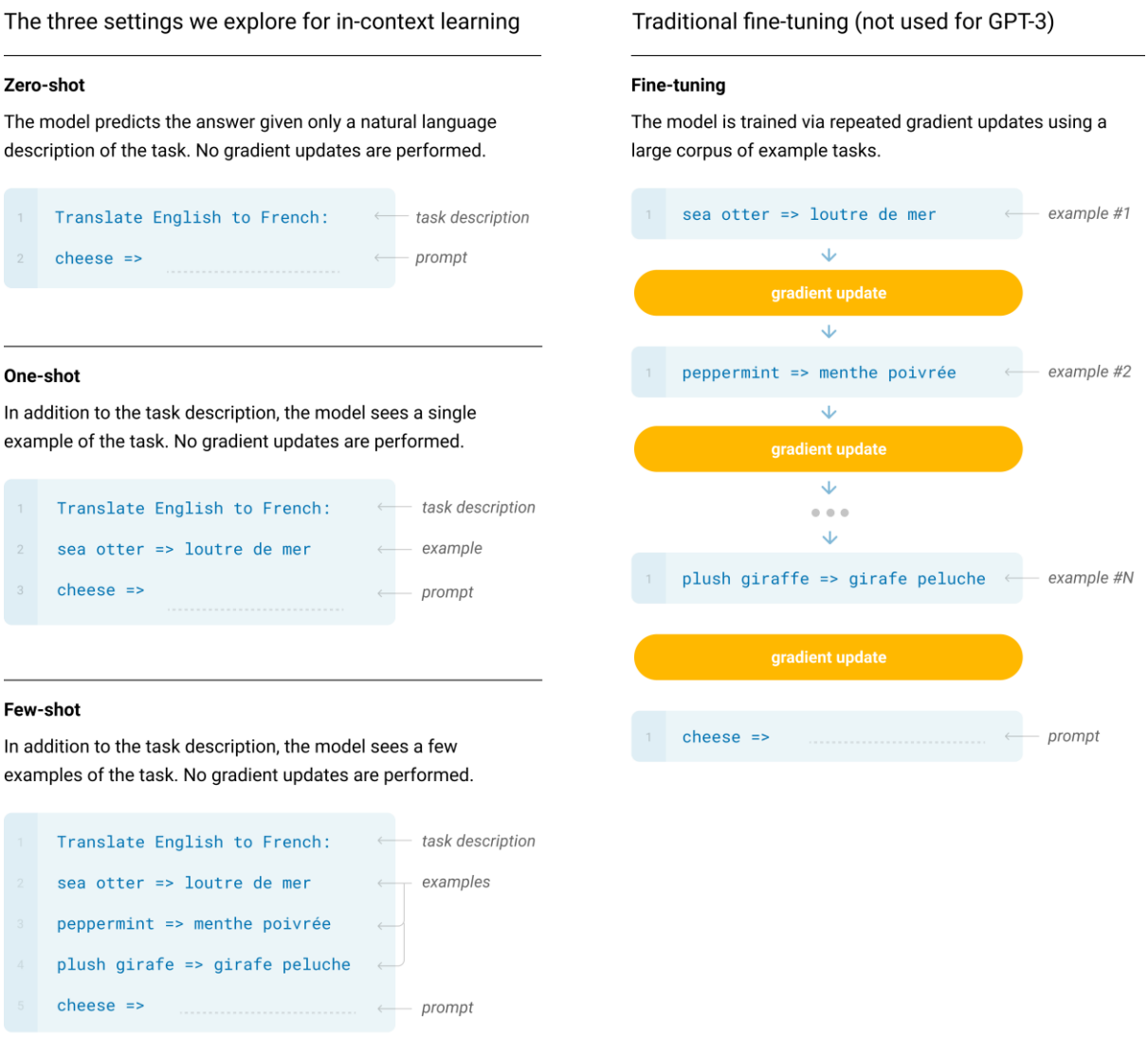

Figure 4 – Zero-shot, one-shot and few-shot learning, contrasted with traditional fine-tuning. (Brown et al., 2020)

2.3.5 InstructGPT and ChatGPT

As discussed in the previous section LLMs can be "prompted" to perform a range of NLP-tasks. However, these models often express unintended behaviors such as making up facts (i.e "hallucinations"), generating biased or toxic text, or simply not following the prompted instructions. The training objective for the LLMs is to predict the next word, which differs from the objective "follow the user's instructions helpfully and safely". The LLM objective is thus "misaligned". For this reason, the authors in (Ouyang et al., 2022) presents InstructGPT which fine-tunes GPT-3 using reinforcement learning from human feedback (RLHF) to align better with user objectives. ChatGPT is a sibling to InstructGPT which uses the same RLHF fine-tuning but with ChatGPT an additional dialog dataset from human AI trainers was used in the fine-tuning step before further fine-tuned with RLHF (OpenAI, 2022).

2.4 Retrieval Augmented Generation (RAG)

As discussed in previous sections, LLMs are task-agnostic models in which task instructions are specified in the prompt at the time of inference. In section 2.3.4 examples were provided showing how to instruct the LLM to perform task by giving examples of how to solve the task at hand in the prompt (few-shot learning). Similarly, one can assign contextualized information in the prompt and tell the LLM to only focus on that information when for example answering a question. For example, a prompt may look like the one in Figure 5:

```
You are a chat bot who loves to help people! Given the following context
sections, answer the question using only the given context. If you are
unsure and the answer is not explicitly in the documentation, say "Sorry,
I don't know how to help with that."

Question:
{user_question} -> input_text

Context sections:
{context} -> Retrieved text snippets that are relevant to Input_text

Answer:
```

Figure 5 – Prompt template

The fundamental idea behind Retrieval Augmented Generation (RAG) systems is to populate a prompt with instructions with external contextual information. RAG systems enable one to leverage the communicative ability of LLMs and restrict the knowledge it communicates to the information found in the documents of our choice. Moreover, as discussed in the previous sections of this thesis, fine-tuning a 100B parameter LLM on custom data is computationally expensive and requires a lot of data. RAG systems are a way to overcome these issues and enable leveraging LLMs on custom data for individuals and businesses.

In the RAG framework, there are two primary steps involved:

- Retrieval Step: When presented with an input like a user's question, the model engages with an external corpus of documents. Its goal is to retrieve document passages from the external corpus that are relevant to the given input question. Document retrieval is further discussed in section 2.4.2.

- Generation Step: Following the retrieval, these passages serve as supplementary context to the original question input. The original question together with the retrieved passage forms a new prompt. This new prompt is then processed by a Large Language Model (LLM), which generates an output, such as an answer to the user's question.
-

Originally outlined as REPLUG by Shi et al. (2023), this method has since become widely recognized as the RAG framework.

2.4.1 Text Embeddings

Unlike word embedding models such as Word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), which transform words to numerical vectors, sentence embedding or text embedding models transform sentences and phrases to fixed-size numerical vectors that captures the semantic meaning of the text. Semantic meaning on a sentence level refers to what the entire sentence conveys as a whole, based on the meanings of its individual words and the way they are combined. For example, in the sentence "The dog is barking," the semantic meaning is that there is a dog, and it is engaging in a specific behavior, which is barking.

2.4.2 Document Retrieval

Given an input text, x, (such as a question), the document retriever aims to retrieve relevant documents to x from an external corpus $D = \{d_1, \dots, d_m\}$. Using a text embedding model, x, and each document $d \in D$ are first encoded into numerical representations or *text embeddings*. Then, to retrieve semantic similar documents in D to x, a similarity search is conducted between the documents and input text. The similarity metric used in the RAG framework for the AI-Assistant is the *cosine similarity*:

$$s(d, x) = \cos(\boldsymbol{d}, \boldsymbol{x}) = \frac{\boldsymbol{d} \times \boldsymbol{x}}{\|\boldsymbol{d}\|_2 \times \|\boldsymbol{x}\|_2}$$

where $\boldsymbol{d}$ and $\boldsymbol{x}$ are the text embeddings generated from the text embedding model of d and x, respectively. $s(d, x)$ assigns a number between -1 and +1 based on the similarity between the text embeddings of the documents in D and the input text, x, where a +1 indicate highest semantic similarity and -1 low semantic similarity.

2.4.3 Contrastive Learning

The main goal of contrastive learning is to train a model (like a neural network) so that it learns to generate text embeddings that are close together for positive pairs and far apart for negative pairs. A “positive pair” consists of two different representations of the same sentence or two different sentences that are semantically similar. For example, the sentences "The dog is sitting on the mat" and "A dog is on a mat" could form a positive pair because they have similar meanings. In addition to positive pairs, we also have "negative pairs." A negative pair consists of two sentences that are not similar in meaning. For example, "The dog is sitting on the mat" and "It is raining outside" could form a negative pair because they have different meanings. The loss function in contrastive learning is designed to penalize the model if the embeddings of a positive pair are far apart or if the embeddings of a negative pair are close together. By minimizing this loss function during training, the model learns to generate useful sentence embeddings reflecting semantic similarities and differences between sentences. After training with contrastive learning, the model should be able to take sentences, or text snippets and generate meaningful embeddings of them, and then, using similarity search, one could find text snippets that are semantically similar to each other (Xu et al., 2023).

2.4.4 GTE - Text Embedding Model

In sentence embedding modeling, fine-tuning the pre-trained BERT model using a contrastive learning approach has resulted in several SOTA (state-of-the-art) sentence embedding models (Zhang et al., 2023). In (Li et al., 2023) the authors present GTE, a general-purpose text embedding model trained with multi-stage contrastive learning that, at the time of writing (summer 2023), is number three on the MTEB leaderboard (Hugging Face, 2023a) which is a leaderboard over the best text embedding models and notably exceeds the performance of the Black-Box embedding model text-embedding-ada-002 from OpenAI.

GTE uses a dual-encoder model architecture. Specifically, it uses two BERT models that are fine-tuned in two stages using contrastive learning. Each BERT model receives an input (such as a text piece) and encodes these texts into embeddings, as discussed in section 2.2. Then, in a dual encoder architecture, the model is optimized based on similarity metrics between these embeddings. The GTE model is learned through the contrastive objective by distinguishing semantic relevant text pairs from irrelevant ones. The authors in (Li et al., 2023) use what they refer to as an "improved contrastive learning objective" and define the following loss function:

$$L = -\frac{1}{n}\sum_{i=1}^{n} log\frac{e^{s(q_i,d_i)/\tau}}{Z}$$

where n is the number of positive query-document pairs in the batch, $s(q_i, d_i)$ is the cosine similarity between q_i and d_i, and the partition function Z is defined as:

$$Z = \sum_{j} e^{s(q_i,d_j)/\tau} + \sum_{j\neq i} e^{s(q_i,q_j)/\tau} + \sum_{j} e^{s(q_j,d_i)/\tau} + \sum_{j\neq i} e^{s(d_j,q_i)/\tau}$$

where τ is the temperature and is fixed at 0.01. The loss function, L, incorporates a softmax operation. The numerator in L is the exponentiated similarity score for the positive pair $s(q_i, d_i)$. This is the desired pairing, and the model aims to update its weights to maximize this value. The denominator in L, Z, is the partition function, which is the sum of the exponentiated similarity scores for all pairs in the batch, both positive and negative ($s(q_i, d_j)$). By dividing the similarity score for the positive pair by the sum of the scores for all pairs, the model effectively compares the score for the correct pairing against the scores for all possible pairings. The goal during training is to adjust the model's weights such that the similarity score for the positive pair is much higher than for any of the negative pairs, resulting in a higher probability for the positive pair after applying the softmax function. The softmax operation converts the similarity scores to probabilities, and by minimizing the negative logarithm of these probabilities, averaged over the batch, the contrastive loss encourages the model to increase the distance (reduce the similarity score) between the negative pairs while decreasing the distance (increasing the similarity score) between the positive pairs.

The training of the GTE model is done in two stages. The first stage uses unsupervised training with only in-batch negatives on 800M text pairs from various open source datasets. This data is collected without any manual annotation. The second stage uses a supervised fine-tuning learning approach based on a smaller 3M text pair data set in which manual annotation is conducted on the relevance between two pieces of text.

2.6 RAG Evaluation - RAGAS

To evaluate QA RAG systems the authors in (Es et al., 2023) propose the RAGAS framework (Retrieval Augmented Generation Assessment). In a QA RAG system, two dimensions need to be evaluated:

- The retrieval component.
- The generation component.

The evaluation of the retrieval component focuses on the retrieval system and its ability to identify relevant contexts. On the other hand, the assessment of the generation component focuses on the LLM's ability to exploit the retrieved context in a faithful and meaningful way.

In (Es et al., 2023) three metrics are proposed for evaluating a QA RAG system:

- *Faithfulness* is a metric based on the idea that the generated answer should be grounded in the context provided. Faithfulness is, hence, a metric that evaluates the generation component and the LLM model's propensity to hallucinate. Further details of the Faithfulness metric will be described in section 2.6.1.

- *Answer Relevance* is a metric to evaluate whether the generated answer is useful and relevant to the question. Answer Relevance is also a measure regarding the generation component and complements Faithfulness; however, instead of measuring the propensity to hallucinate, it measures the quality of provided answers. The details of the Answer Relevance metric will be discussed in section 2.6.2.

- *Context Relevance* is a metric related to the retrieval component. Specifically, Context Relevance evaluates the focus of the retrieved context, which should be relevant to the question asked and not include abundant irrelevant information. Details of the Context Relevance metric will be discussed in section 2.6.3.

As will become clear in subsequent sections, each of the three evaluation metrics proposed in (Es et al., 2023) leverages additional LLM models to evaluate the RAG system. In (Es et al., 2023), the authors use the GPT-3.5-Turbo-16k model; however, in this thesis, the GPT-4 model will be the engine behind the evaluation metrics.

2.6.1 Faithfulness (FA)

Given a question, q, and a context $c(q)$, the RAG system generates an answer $a(q)$. $a(q)$ is regarded faithful to the context $c(q)$ if the claims outlined in $a(q)$ are coming from the context. The Faithfulness (FA) calculation has a two-step procedure:

- First, an LLM (GPT-4) is used to identify the statements, s_i, provided in $a(q)$ from the RAG system. The prompt for instructing the LLM to do so is shown in Figure 6.
- Second, another LLM is used to verify if the statements from step 1 can be inferred from the context $c(q)$. For each statement, s_i, a yes or no verdict is determined by the LLM. The prompt for this step is shown in Figure 7.

The FA metric is then calculated as:

$$FA = \frac{Number\ of\ statements\ in\ the\ answer\ inferred\ from\ context}{Total\ Number\ of\ statements\ in\ the\ answer}$$

FA is calculated as the proportion of statements that can be inferred from the context, $c(q)$, to all statements derived from the answer. Notice the prompt in step 1, that if the answer is "I don't know" then the statement is "I don't know" and in turn that the prompt for step two if the statement is “I don’t know” it is assigned a yes verdict. This is done so that the FA score is related to hallucination only, and an "I don't know" answer does not provide false facts. It provides no facts at all. By removing the “I don’t know” answers, the FA score will reflect the proportion of statements, s_i, that can be inferred from the context, $c(q)$, to the total number of statements in the answer $a(q)$.

```
Given a question and answer, create one or more statements from each
sentence in the given answer. Each statement must start with "Statement
nr. i:". Separate each statement by "XXX". If the answer is "I don't
know.", assign "I don't know." as statement.

question: {question}
answer: {answer}
```

Figure 6 – Faithfulness (FA) – Prompt 1.

```
Consider the given context and following statements, then determine
whether they are supported by the information present in the context.
Provide a final verdict either "yes" or "no" for each statement in the
given format. If the statement reads "I don't know" or similar, assign
a "yes" verdict.

context = {context}
statements = {statements}

format:
statement nr 1: yes
...
statement nr n: no

Do not deviate from the specified format.
```

Figure 7 – Faithfulness (FA) – Prompt 2.

2.6.2 Answer Relevance (AR)

An answer $a(q)$ is relevant if it directly addresses the question in an appropriate way. For the given answer $a(q)$, an LLM is instructed to generate 5 questions, q_i, based on $a(q)$. The prompt to the LLM is shown in Figure 8. Each q_i is then inserted to the GTE embedding model to generate embeddings to these LLM-generated questions. Then, the average cosine similarity score between these 5 generated questions and the original question makes up the Answer Relevance (AR) score. That is,

$$AR = \frac{1}{5}\sum_{i=1}^{n} cos(q, q_i)$$

If there is a high semantic similarity between the questions generated from the original answer and the original question, then the original answer must be relevant to the original question.

```
Generate 5 questions based on the given answer.
Each question must start with "Question nr. i:".

answer: {answer}
```

Figure 8 – Answer Relevance (AR) Prompt.

3.6.3 Context Relevance (CR)

Preferably, we would want the retriever part of the AI-Assistant only to retrieve relevant information to the question. To quantify context relevance, given a question q and its context, $c(q)$, an LLM is instructed to extract sentences from the context, $c(q)$, that are important to answer the question q. The prompt for the LLM is shown in Figure 9. The Context Relevance (CR) score is then computed as:

$$CR = \frac{Number\ of\ relevant\ sentences}{Number\ of\ total\ sentences\ in\ the\ context}$$

```
Please extract relevant sentences from the provided context that can
potentially help answer the following question. If no relevant
sentences are found, or if you believe the question cannot be answered
from the given context, return the phrase "Insufficient Information".
While extracting candidate sentences you're not allowed to make any
changes to sentences from given context. Each extracted sentence must
start with "Sentence nr. i: ".

question: {question}

context: {context}
```

Figure 9 – Context Relevance Prompt.

3. Method & Implementation

In this section, the implementation steps taken for the AI-Assistant are presented. The model will be implemented using LangChain python package version 0.0.273 (LangChain, 2023a).

3.1 Model Overview

The framework for the AI-Assistant follows a similar approach to that of REPLUG in (Shi et al., 2023), discussed in section 2.4. Figure 10 outlines the model overview for the AI-Assistant.

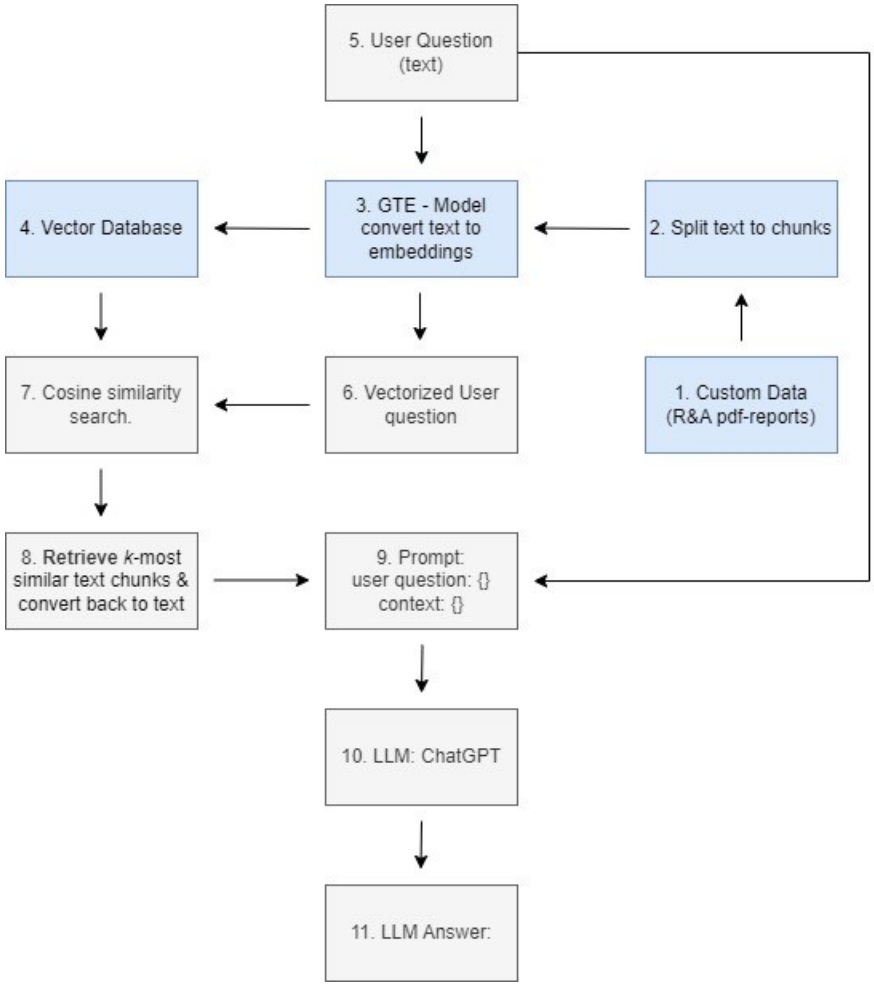

Figure 10 – Model Overview for the AI-Assistant.

The steps in Figure 10 are as follows:

1. The R&A reports, which come in pdf files, are converted into raw text.

2. The text is then split into smaller text chunks, which will be further discussed in section 3.2.
3.
4. These text chunks are converted into embeddings discussed in section 3.3.

5. The text embeddings are then stored in a vector database (VB) discussed in section 3.4.

Steps 1 to 4, highlighted in blue in Figure 10, are done once.

6. As user questions arrive, they come in the form of a text to the AI-Assistant.

7. A copy of the user text is then transformed into an embedding.

8. A similarity search is conducted between the user question embedding and the embeddings stored in the vector database.

9. The *k*-most similar text chunks are then retrieved in their text format.

10. The user question, together with the retrieved text chunks and an instruction, populates a new prompt.

11. The prompt is sent to ChatGPT; specifically, the gpt-3.5-turbo-0613 API (OpenAI, 2023a) will be used along with the gpt-4-0613 as reference.

12. ChatGPT generates an answer based on the question and the context.

3.2 Load and Chunk Text Data

Using LangChains PyMuPDFLoader(), the TQR pdf reports are downloaded from linclund.com. Then, the text within each page, for each pdf, is chunked into pieces of 450 tokens, with 62 “overlap” tokens. 512 tokens approximately translate to 384 words (OpenAI, 2023b). 512 tokens are chosen since this is the maximum size the GTE embedding model can take (Hugging Face, 2023b). The gpt-3.5-turbo-0613 model has a token limit of 4096 tokens that both the input prompt and generated answer must obey (OpenAI, 2023c). In step 8 in the previous section *k*-most similar text chunks to the question are going to be retrieved and populate the new prompt that is going to be sent to ChatGPT. Choosing $k = 5$, would mean a token size of approximately 1537 tokens, or approximately 1153 words for the question, instructions, and generated answers. For the gpt-4-0613 model, the token limit is 8192 tokens (OpenAI, 2023c).

3.3 Convert Text to Embeddings

The large version of the GTE model discussed in section 2.4.4 is used to convert the text chunks and questions into numerical vectors, i.e., text embeddings. The GTE model is downloaded from Hugging Face (Hugging Face, 2023b). As a reference embedding model, the black-box text-embedding-ada-002 model from OpenAI will be used. The ada model has a token limit of 8192 (OpenAI, 2023d).

3.4 Chroma – Vector Database

I've chosen an external vector database provider, Chroma, which is integrated into the LangChain landscape (LangChain, 2023b). One could easily set up an in-memory vector database using the Pandas package in Python, for example; however, from my experience, using a vector database that is integrated into the LangChain landscape seems more efficient for downstream LangChain methods. As step 4 in section 3.1 describes, the vector database stores the embeddings produced by the embedding model GTE. In addition, meta-data such as the source of the text chunk, pdf report, and page number is also stored. This enables the user to fact-check what the AI-Assistant is saying and further read into the question.

3.5 Prompt-templates

Similar to Figure 5 in section 2.4, two prompt templates will be tested. The two different templates are shown in Figure 11 & 12. In Figure 11, Prompt Template 1, the LLM is given clear instructions to use the context when answering questions. Furthermore, it is instructed not to make up answers and convey an "*I don't know*"-answer when it does not know the answer. This will hopefully mitigate the LLM tendency to hallucinate. However, "*Use the following pieces of context to answer the question*" does not explicitly direct the LLM to use the context. In Figure 12, Prompt Template 2, a more restrictive instruction is given to the LLM. It instructs the LLM *only* to use the information provided in the context sections. This is a designer choice. On one hand, using Prompt Template 1 would let the LLM use its parameter knowledge it has acquired during training. As long as it is not making up facts, this might be desirable. But does the LLM know when it makes up facts? One might want to restrict the answers from the LLM only to be based on what is stored in the vector database and use the LLM only as a medium for communication. However, although Prompt Template 2 instructs the LLM only to use the context provided, there is no guarantee that it will obey the instructions 100%. In Figure 11 & 12, {context} is a variable that will be filled with $k = 5$ text chunks as context to the user question. As discussed, these 5 text chunks are selected based on their semantic similarity to the user question.

```
Use the following pieces of context to answer the user's question. If you
don't know the answer, answer with "I don't know", don't try to make up
an answer. Make sure each answer doesn't use reference terms like "in the
context", "in the project", or anything that's not available to the
reader.
----------------
Context: {context}
----------------
Question: {question}
----------------
Answer:
```

Figure 11 - Prompt Template 1.

```
Use only the following pieces of context when answering the user's
question. If you can't derive an answer to the question from the context,
answer with "I don't know", don't try to make up an answer. Make sure
each answer doesn't use reference terms like "in the context", "in the
project", or anything that's not available to the reader.
----------------
Context: {context}
----------------
Question: {question}
----------------
Answer:
```

Figure 12 – Prompt Template 2.

3.5 LLM - Parameters

The ChatGPT models typically generate text by sampling from a probability distribution over all possible tokens. This means that running the same prompt input can lead to different outputs. The temperature parameter influences the level of randomness in the text generation process, with values ranging between 0.0 and 2.0. For the models in the AI-Assistant the temperature parameter is set to 0 (default is 1) to make the model's responses as deterministic as possible, as it will then always choose the token with the highest probability. This means the model follows the patterns it learned during training as closely as possible, adhering to the most common associations and sequences found in the data it was trained on. Since these associations are based on a large corpus of text that presumably contains coherent and sensible information, sticking to the most probable tokens should, in theory, reduce the model's tendency to generate text that deviates from these patterns, thereby reducing the risk of making up information or "hallucinating".

4. Results

In this section the results of the AI-Assistant using different RAG-configurations will be presented and discussed along with the questions used to evaluation the configurations.

4.1 Samples

To evaluate the AI-Assistant, 25 questions were manually constructed. These questions were formulated such that the information needed to answer them has a high likelihood of being in the documents. The questions used are shown in Table 1.

	Questions
1	who are OQAM?
2	what is 'TQR' referred to?
3	what is 'TQR' referred to in the context of trading and investing?
4	who are lynx asset management?
5	explain what a price driver is in the context of trading/investing
6	explain what is refered to as a signal?
7	explain what risk management is in the context of trading
8	explain what technical analysis is and give some examples
9	explain what Breakout is
10	explain what ADX is
11	what is Double tops?
12	how can machine learning be used in trading? List some examples
13	what is a backtest?
14	what is backtest overfitting?
15	what is credit default swap?
16	give examples on how AI can be used in trading/investing?
17	how can computer vision be used in trading/investing?
18	how can neural networks be used in trading/investing?
19	give examples on some machine learning models that can be used in trading/investing?
20	explain the random walk theory
21	explain the efficient market hypothesis
22	what is calmar ratio?
23	explain sharpe ratio
24	what is a market neutral strategy?
25	what is asset allocation?

Table 1 – Sample Questions

4.2 Results

The two prompts discussed in section 3.5 are compared using different RAG configurations:

1. Comparing the usage of the GTE embedding model vs. the black-box text-embedding-ada-002 from OpenAI.

2. Comparing the usage of GPT-3.5 and black-box GPT-4 as the answer-generating model.

Table 2 shows the results using GPT-3.5 as the answer-generating model, and Table 3 shows the results using GPT-4 as the answer-generating model. These results are the average of the FA, AR, and CR scores over the 25 questions for each RAG configuration after having removed the “I don’t know” responses. Tables 4 & 5 list the questions in which one or more model RAG configurations provided an "I don't know" answer.

LLM: GPT 3.5	Prompt 1, ADA	Prompt 2, ADA	Prompt 1, GTE	Prompt 2, GTE
CR	29%	**33%**	25%	24%
AR	91%	91%	**92%**	90%
FA	83%	90%	76%	**91%**
Average	68%	**71%**	64%	68%

Table 2 – Results GPT 3.5 – Average scores over the 25 sample questions. Bold indicates the highest score for each score. The bottom row shows the average of the average CR, AR and FA score.

LLM: GPT 4.0	Prompt 1, ADA	Prompt 2, ADA	Prompt 1, GTE	Prompt 2, GTE
CR	**33%**	32%	21%	24%
AR	91%	90%	**91%**	91%
FA	96%	**98%**	95%	92%
Average	73%	**73%**	69%	69%

Table 3 – Results GPT 4.0 – Average scores over the 25 sample questions. Bold indicates the highest score for each score. The bottom row shows the average of the average CR, AR and FA score.

LLM: GPT 3.5	Prompt 1, ADA	Prompt 2, ADA	Prompt 1, GTE	Prompt 2, GTE
who are lynx asset management?	1	1	1	1
who are OQAM?	0	1	0	0
what is 'TQR' refered to?	0	1	1	1
what is 'TQR' refered to in the context of trading and investing?	0	1	0	1
give examples on some machine learning models that can be used in trading/investing?	0	1	0	0
explain what risk management is in the context of trading	0	0	0	0
explain what a price driver is in the context of trading/investing	0	0	0	1
explain the random walk theory	0	1	1	1
Total Sum:	*1*	***6***	*3*	*5*

Table 4 – "I don't know"s GPT 3.5 – a 1 indicate that the RAG configuration answered "I don't know" on the question and a 0 that the RAG configuration provided an answer other than "I don't know".

LLM: GPT 4	Prompt 1, ADA	Prompt 2, ADA	Prompt 1, GTE	Prompt 2, GTE
who are lynx asset management?	1	1	1	1
who are OQAM?	0	0	0	0
what is 'TQR' referred to?	0	1	1	1
what is 'TQR' referred to in the context of trading and investing?	1	1	0	0
give examples on some machine learning models that can be used in trading/investing?	0	0	0	0
explain what risk management is in the context of trading	0	0	0	1
explain what a price driver is in the context of trading/investing	0	0	0	1
explain the random walk theory	1	1	1	1
Total Sum:	*3*	*4*	*3*	***5***

Table 5 – "I don't know"s GPT 4 – a 1 indicate that the RAG configuration answered "I don't know" on the question and a 0 that the RAG configuration provided an answer other than "I don't know".

4.3 Discussion

CR: The results in Tables 2 & 3 in section 4.2 indicate that the retrieval component has room for improvement. The black-box text-embedding-ada-002 model tends to outperform the GTE embedding model, but the CR scores are still rather low. Recall from section 3.6.3 that the CR score is the proportion of sentences in the context helpful for answering the question to the total number of sentences in the context. The parameters regulating this are the *k*- most similar text chunks and the text-chunk size. *k* was somewhat arbitrarily chosen to *k*= 5; the low CR scores indicate this could be an even lower number. As discussed in section 3.2, the chunk size was chosen to 512 tokens to maximize the token limit of GTE. However, to motivate a shrinkage of this parameter is not that clear; on one hand, it makes sense to cover as much semantic meaning as possible in the embeddings. On the other hand, too much information may be covered in the text chunk such that information with strong semantic similarity to the question gets blurred by the additional information in the text chunk when transforming the text into a vector representation. Since the context size consumes a lot of tokens, one would prefer to have the context as small as possible with only relevant text, i.e., a high CR score. On the other hand, having a "shot-gun" approach, with more information in the context, increases the chance that the information required to answer the user question will be covered in the context if it exists. With the "shot-gun" approach, it will then be up to the cleverness and token size limit of the LLM to sort out the context and provide that information to the user.

AR: Recall the AR score, which was the average cosine similarity between the originally asked question and five generated questions based on the RAG-generated answer. Tables 2 & 3 in section 4.2 show that the AR scores seem similar over all RAG configurations, indicating that the answers generated are highly relevant to the questions asked.

FA: As discussed in section 2.6.1, the FA metric measures the tendency of hallucination and was calculated as the proportion of the number of statements in the RAG-generated answer that could be backed by the information in the context to the total number of statements in the RAG-generated answer. In Table 3, using the GPT-4 model the scores are high for all RAG-configurations. However, in Table 2, using GPT-3.5, the prompts make a difference, where using prompt 1 makes the model deviate from the context information to a greater extent than prompt 2. These results indicate that GPT-3.5 is, to a greater extent than GPT-4, sensitive to how the prompt is formulated. This is somewhat backed by the results in Tables 4 & 5. Note that in Tables 4 & 5, all RAG configurations with the text-embedding-ada-002 model have the same context, likewise for the model configurations with the GTE embedding model. However, in Table 4, when using GPT-3.5, the number of "I don't know" responses tends to be higher using prompt 2 than prompt 1. The difference is not that large in Table 5, when using GPT-4.

Overall: Taking the average of the three scores in Tables 2 & 3, the black-box models, GPT-4 and text-embedding-ada-002 model, seem to be the better choices.

5. Conclusion and Future Work

This section will discuss the conclusions from the results provided in section 4. In addition, this section will also offer future work proposals, both in terms of consideration of improving RAG systems but also in terms of further development of firm intelligence.

5.1 Conclusion

The objectives stated in section 1.3, are repeated here once more:

1. *To construct a question-answering (QA) AI-Assistant based on the RAG framework using the TQR reports.*

2. *If the TQR reports does not have the information needed to answer the question asked by the user the AI-Assistant clearly states that it does not know and does not make up facts,*

could be considered to have been fulfilled. The application of the Retrieval-Augmented Generation framework allowed the leveraging of LLMs like GPT-3.5 and GPT-4, on the custom data set consisting of TQR reports. As for objective nr 2, the results in Tables 2-5 in section 4.2, showed that the integration of black-box models, particularly GPT-4 in conjunction with text-embedding-ada-002, exhibited superior performance across multiple metrics, suggesting their suitability for complex question-answering tasks.

Despite these promising outcomes, the study also revealed areas for improvement. The relatively low CR scores suggest a refinement of the retrieval component is necessary to enhance the precision of context selection. This finding points to a potential trade-off between the breadth of information included in the context and the relevance of that information to the question at hand. The AI-Assistant's proficiency in acknowledging its limitations, specifically by admitting when it does not know an answer, aligns with the ethical imperative to avoid misleading users. This transparent approach reinforces trust in AI systems and highlights the importance of clear communication in AI-user interactions.

5.2 Future work

This thesis was primarily written during the summer of 2023. Since the release of ChatGPT in 2022, the number of research papers on the topic to improve the usage of LLM’s has been exponential. As such, many paths have also been taken toward improving RAG systems. Here I will propose a couple of techniques to consider for future work within RAG-system application that might already have been proposed elsewhere but not to my knowledge.

5.2.1 Future Work - Improving RAG

As discussed in section 4.3, the GTE and text-embedding-ada-002 model transforms the text chunks into embeddings. If those chunks include different kinds of information, then it could easily be the case that the vector representation is too non-specific. For instance, if the chunk includes “The dog was barking. The house was burning”, then there are two things happening in the text chunk that will be reflected in the vector representation. If the question was "What is burning?" then the vector representation of the sentence "The house was burning" would be more similar than the vector representation of the whole chunk "The dog was barking. The house was burning". In that sense, the vector representation of the chunk is noisier than the vector representation of the sentence. The embeddings are thus worse than they need to be. Moreover, the retrieval step is also related to the Faithfulness score in that, when the AI-Assistant answers "I don't know", is this because the information needed to answer the question is not in the documents, or was it due to noisy embedding? Thus, the need for better retrieval is necessary. Below are some points one might want to consider for future work:

1. Retrieval: In the RAG system presented, text chunk size was set to 512 tokens and then these text chunks were converted to embeddings and stored in the vector base for later retrieval. In addition to the chunks of size 512, one could also store chunks with size of 256, 128, 64, etc. The retrieving step could then either be conducted using top-k chunks as before or maybe use the same approach as when calculating the CR metric in RAGAS, that is to instruct an intermediate LLM to sort out the relevant sentences and then send those to the answering LLM. Alternatively, one could use Parent Document Retrieval (LangChain, 2023c), in which the larger parent document that the embedding belongs to is retrieved. Another point related to improving retrieval is by implementing *Agents*, which will be discussed in section 5.2.2.

2. Question generation: For better formulated questions, one could have an intermediate LLM rephrasing the question. Then, the RAG system does a similarity search between this generated question or questions and the documents instead of the original user question and the documents.

3. Evaluation: In the evaluation step of the RAG systems, the 25 sample questions were manually formulated. Instead, One could instruct an LLM to generate a sample set consisting of synthetic questions based on the context and some questions not in the context. This would be beneficial regarding time and more robust metrics because one can generate a larger sample.

5.2.2 Future work - Agents

As discussed in the introduction section, the AI-Assistant presented in this thesis could be considered the embryo of a firm-intelligence unit. The next step towards such a unit could be incorporating what is called *Agents* (LangChain, 2023d). One of the main sought-out properties of a firm intelligence unit is that it can make conclusions that human researchers have overlooked. For example, finding new research ideas that human researchers have overlooked by putting together results from multiple research reports, here Agents seems promising. In (Yao et al., 2023) the authors presented the ReAct framework. On a high-level, the ReAct framework involves an LLM-network in which one orchestrator LLM applies reasoning about what action to take by utilizing *tools* that in turn have different properties. These tools can be calculators, API's, code interpreters, data storages, search and lookup utilities, or other LLMs etc. Conditionally on the question and based on the descriptions of the tools, the orchestrator LLM splits up the question and applies reasoning on how to distribute the question pieces to the tools it has at hand. An example of the thought process of such Agent is shown in Figure 12.

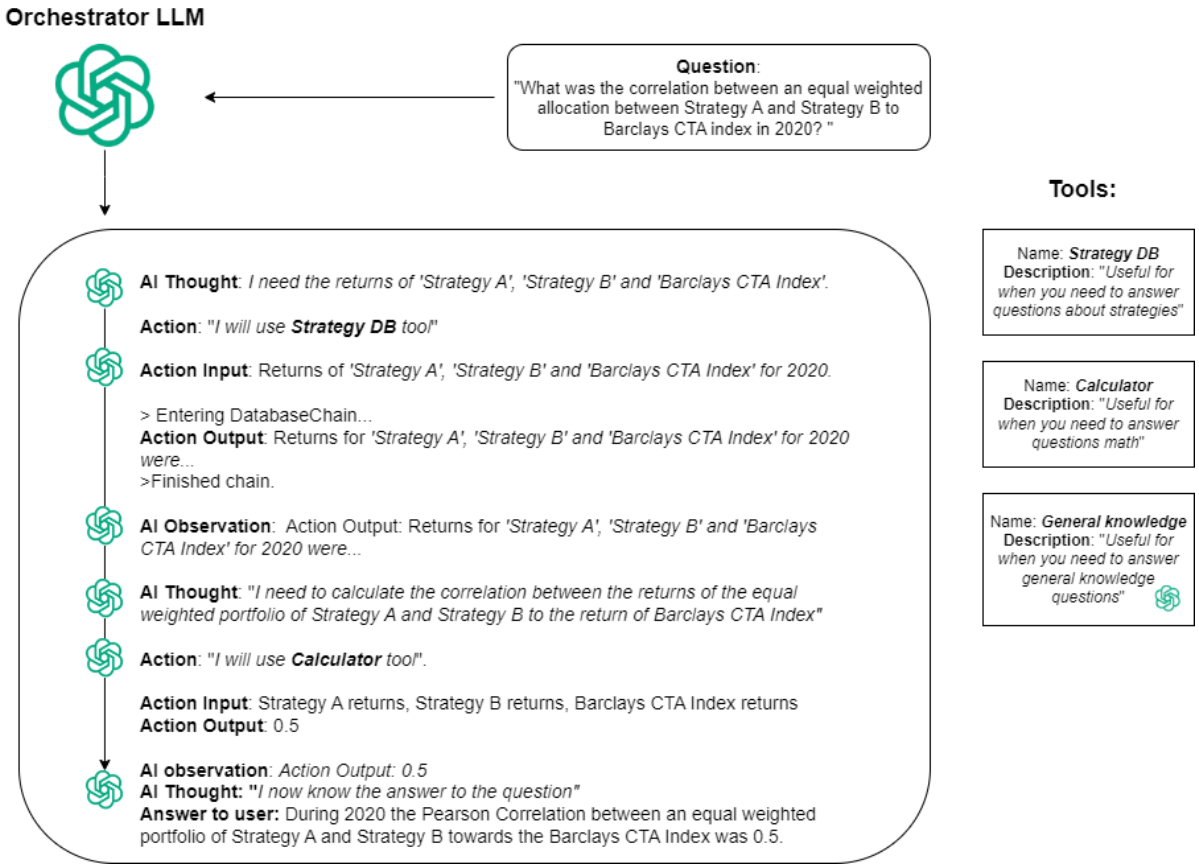

Figure 12 – Agent example - The orchestrator LLM have three tools at its disposal: 1 - A database that includes returns for the firm's strategies and in this example also the returns of the Barclays CTA Index. 2 - A calculator which conducts simple math operations. 3- A second LLM which is utilized for general knowledge questions. Each of these tool have descriptions and names. When building up an answer to the user, the orchestrator LLM makes decisions based on these descriptions when and if these tools should be used.

References

Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I. and Amodei, D., (2020). *Language Models are Few-Shot Learners*. OpenAI. Available online: https://arxiv.org/pdf/2005.14165.pdf [Accessed: 22 August 2023].

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K., (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Available online: https://arxiv.org/abs/1810.04805 [Accessed: 22 August 2023].

Hugging Face, (2023a). *MTEB leaderboard*. Available online: https://huggingface.co/spaces/mteb/leaderboard [Accessed: 22 August 2023].

Hugging Face, (2023b). *GTE-LARGE: General Text Embeddings (GTE) model*. Hugging Face. Available online: https://huggingface.co/thenlper/gte-large [Accessed: 22 August 2023].

Kaplan, J., McCandlish, S., Henighan, T., Brown, T.B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., Amodei, D. and others, (2020). *Scaling Laws for Neural Language Models*. Available online: https://arxiv.org/pdf/2001.08361.pdf [Accessed: 22 August 2023].

LangChain, (2023a). *Documentation*. Available online: https://docs.langchain.com/docs/ [Accessed: 22 August 2023].

LangChain, (2023b). *Chroma*. Available online: https://python.langchain.com/docs/integrations/vectorstores/chroma [Accessed: 22 August 2023].

LangChain, (2023c). *Agents*. Available online: https://python.langchain.com/docs/modules/agents/ [Accessed: 22 August 2023].

LangChain, (2023d). *Parent Document Retriever*. Available online: https://python.langchain.com/docs/modules/data_connection/retrievers/parent_document_retriever [Accessed: 26 December 2023].

Li, Z., Zhang, X., Zhang, Y., Long, D., Xie, P. and Zhang, M., (2023). *Towards General Text Embeddings with Multi-stage Contrastive Learning*. Available online: https://arxiv.org/pdf/2308.03281.pdf [Accessed: 22 August 2023].

Lund University Finance Society (LINC) (2023a). *About*. Available online: http://linclund.com/about/ [Accessed: 26 August 2023]

Lund University Finance Society (LINC) (2023b). *Research & Analysis Archive*. Available online: https://linclund.com/archive-test-content-views/ [Accessed: 26 August 2023]

Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013. *Efficient Estimation of Word Representations in Vector Space*. Available online: https://arxiv.org/abs/1301.3781 [Accessed: 22 August 2023].

OpenAI, (2022). ChatGPT. OpenAI Blog. Available online: https://openai.com/blog/chatgpt [Accessed: 22 August 2023].

OpenAI, (2023a). Model index for researchers. OpenAI Platform. Available online: https://platform.openai.com/docs/model-index-for-researchers [Accessed: 26 August 2023].

OpenAI, (2023b). What are tokens and how to count them?. OpenAI Help Center. Available online: https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them [Accessed: 26 August 2023].

OpenAI. (2023c). Models. OpenAI. Available online: https://platform.openai.com/docs/models [Accessed: 22 August 2023].

OpenAI. (2023d). New and Improved Embedding Model. OpenAI. Available online: https://openai.com/blog/new-and-improved-embedding-model [Accessed: 22 August 2023].

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C.L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J. and Lowe, R., (2022). *Training language models to follow instructions with human feedback*. Available online: https://arxiv.org/pdf/2203.02155.pdf [Accessed: 22 August 2023].

Pennington, J., Socher, R. and Manning, C.D., (2014). *GloVe: Global Vectors for Word Representation*. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, pp. 1532-1543. Available online: https://aclanthology.org/D14-1162/ [Accessed: 22 August 2023].

Radford, A., Narasimhan, K., Salimans, T. and Sutskever, I., (2018). *Improving Language Understanding by Generative Pre-Training*. OpenAI. Available online: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf [Accessed: 22 August 2023].

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I., (2019). *Language Models are Unsupervised Multitask Learners*. OpenAI. Available online: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf [Accessed: 22 August 2023].

Shi, W., Min, S., Yasunaga, M., Seo, M., James, R., Lewis, M., Zettlemoyer, L. and Yih, W., (2023). *REPLUG: Retrieval-augmented Black-box Language Models*. Available online: https://arxiv.org/abs/2301.12652 [Accessed: 22 August 2023].

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need*. Available online: https://arxiv.org/abs/1706.03762 [Accessed: 22 August 2023].

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y, (2022). *ReAct: Synergizing Reasoning and Acting in Language Models.* Available online: https://arxiv.org/abs/2210.03629 [Accessed: 22 August 2023].

Xu, L., Xie, H., Li, Z., Wang, F.L., Wang, W. and Li, Q., (2023). *Contrastive Learning Models for Sentence Representations*. ACM Transactions on Intelligent Systems and Technology, 14(4), Article 67. Available online: https://dl.acm.org/doi/pdf/10.1145/3593590 [Accessed: 22 August 2023].

Zhang, J., Lan, Z. and He, J., 2023. *Contrastive Learning of Sentence Embeddings from Scratch*. Available online: https://arxiv.org/abs/2305.15077 [Accessed: 22 August 2023].