Mittuniversitetet
MID SWEDEN UNIVERSITY

Automating Data Extraction from Documents Using Large Language Models

A Study Exploring How AI Can Be Used to Transform Unstructured Data into Structured Formats

Lucas Persson

Document type – Bachelor Thesis
Main field of study: Computer engineering
Credits: 15
Semester/year: Spring/2024
Supervisor: Martin Kjellqvist
Examiner: Stefan Forsström
Course code: DT099G

At Mid Sweden University, it is possible to publish the thesis in full text in DiVA (see appendix for publishing conditions). The publication is open access, which means that the work will be freely available to read and download online. This increases the dissemination and visibility of the degree project.

Open access is becoming the norm for disseminating scientific information online. Mid Sweden University recommends both researchers and students to publish their work open access.

I/we allow publishing in full text (free available online, open access):

☒ Yes, I/we agree to the terms of publication.

☐ No, I/we do not accept that my independent work is published in the public interface in DiVA (only archiving in DiVA).

Location and date
Sundsvall 2024-06-16

Programme/Course
Bachelor of Science in Engineering, Computer Engineering

Name (all authors names)
Lucas Persson

Year of birth (all authors year of birth)
2001-07-27

Abstract

The objective of this thesis is to accurately extract at least 70% of external references from one of three test documents and to compare the performance of two Large Language Models (LLMs) using quantitative methods. This is measured by evaluating both the number of identified references and the amount of these references that are similar to the expected output. The process begins with extracting text from a PDF document, followed by dividing the text into sentences. Embeddings are then generated for each sentence. Cosine similarity is performed on these embeddings to filter out sentences that do not contain the requested data. The remaining sentences are processed using two OpenAI models, gpt-3.5-turbo-0125 and gpt-4-turbo-2024-04-09, accessed via their API. Each model is instructed to extract external references from the filtered sentences. The extracted references are then compared against the expected outputs in two ways: by the number of correctly identified references and by the level of detail in the extracted references. The gpt-4-turbo-2024-04-09 model successfully extracted 42 out of 43 references, with 41 being optimal and the remaining missing some information. The gpt-3.5-turbo-0125 model extracted 41 out of 43 references, with 31 matching the expected output perfectly. These results demonstrate the potential of Large Language Models in accurately extracting data from unstructured sources.

Keywords: Large Language Models, data extraction, unstructured sources, embeddings, cosine similarity

Sammanfattning

Målet med detta examensarbete är att extrahera minst 70 % av externa referenser från ett av tre testdokument och att jämföra resultaten mellan två stora språkmodeller (LLM) med kvantitativa metoder. Detta mättes genom att utvärdera både antalet korrekt identifierade referenser och likheten hos dessa referenser jämfört med det förväntade resultatet. Processen börjar med att extrahera text från ett PDF-dokument, följt av att dela upp texten i meningar. "Embeddings" genereras sedan för varje mening. "Cosine similarity" utförs på dessa "embeddings" för att filtrera bort meningar som inte innehåller den begärda informationen. De återstående meningarna bearbetas med två OpenAI-modeller, gpt-3.5-turbo-0125 och gpt-4-turbo-2024-04-09, som nås via deras API. Varje modell får sedan en detaljerad instruktion att extrahera externa referenser från de filtrerade meningarna. De extraherade referenserna jämförs sedan mot de förväntade ut datat på två sätt: med antalet korrekt identifierade referenser och med detaljnivån i de extraherade referenserna. Modellen gpt-4-turbo-2024-04-09 extraherade framgångsrikt 42 av 43 referenser, där 41 var optimala och den återstående saknade viss information. Modellen gpt-3.5-turbo-0125 extraherade 41 av 43 referenser, där 31 matchade det förväntade ut datat perfekt. Dessa resultat visar potentialen hos stora språkmodeller för att extrahera data från ostrukturerade källor med en högre träffsäkerhet.

Nyckelord: Stora språkmodeller, dataextrahering, ostrukturerad data, embeddings, cosine similarity

Acknowledgements

I would like to give a huge thank you to my supervisor Martin Kjellqvist at Mid Sweden University and to both my supervisors at Standard Solution Group (SSG) for the help and guidance through the whole project.

Table of Contents

Terminology / Notation

Acronyms/Abbreviations

AI	Artificial Intelligence
LLM	Large language model
t-SNE	t-distributed stochastic neighbor embedding

1 Introduction

This thesis investigates the use of Artificial Intelligence (AI) for extracting information from unstructured data, a significant challenge given the exponential growth of data across various formats. The aim is to evaluate and compare different Large Language Models (LLM) to identify efficient, accurate, and scalable solutions for data extraction. This work is proposed and supported by the company Standard Solution Group (SSG).

The aim of this chapter is to provide an overview of the background, problem motivation, and the overall purpose and context of this study.

1.1 Background and motivation

In the rapidly evolving digital landscape, the need for sophisticated artificial intelligence (AI) systems has never been greater. This big shift isn't just because AI can take over everyday tasks. It's more about how AI can greatly improve things and speed up processes. As companies and entire industries rush to make the most of AI, there's a growing need for AI-powered tools that can analyze and extract valuable information from large piles of unorganized data.

AI-powered tools are essential because of the challenges associated with handling unstructured data, which is characterized by its complexity and variability in structure. Since traditional data processing systems are typically designed to handle structured data that follow a specific format and are easy to categorize, it causes them to struggle with processing the complexity and diversity of unstructured data.

In this thesis will the AI-powered tool Large Language Models (LLM) be utilized. A Large Language Model (LLM) is an advanced AI system designed to understand and generate human-like text by learning from a large database of existing content. These models, which include well-known examples like OpenAIs GPT series and Googles Gemini, use complex algorithms to process and produce language in a way that mimics human conversations. Capable of tasks such as translation, summarization, and even creative writing, LLMs are increasingly integral to various applications, from virtual assistants to content generation. [1]

1.2 Overall aim and problem statement

The specific problem investigated in this thesis is: How can LLMs be utilized to accurately convert unstructured datasets into structured datasets? This involves detailed exploration into methods of text handling and embeddings. The solution to this problem will be evaluated by how accurate LLMs can transform unstructured data into structured datasets including both quantity and quality. The higher reason for this work is the following:

- Unstructured data poses challenges in data analysis, limiting organizations' ability to derive actionable insights.
- Current data extraction methods may not efficiently convert unstructured data into usable formats.
- There is a need to explore AI-driven solutions for transforming unstructured data into structured datasets effectively and accurately.

1.3 Research questions/Scientific goals/Verifiable goals

The objectives of this thesis are:

1. Successfully extract at least 70% correct external reference from one of the documents.
2. Compare the performance of two Large Language Models (LLMs).

The research questions to be answered are the following:

1. How does the accuracy differentiate between the models when comparing the amount of identified external references?
2. How does the quality differentiate between the models when comparing the identified external references?

1.4 Scope

This thesis has been delimited in some areas, the chosen document to test on are standards produced by Standard Solution Group (SSG), the task has been to extract external references, and the documents are only in swedish. This has been tested on two different LLMs with the same instruction to extract external references.

1.5 Outline

This report begins with Chapter 2, which sets the theoretical groundwork necessary for the thesis. Chapter 3 then details the scientific methodologies and evaluation techniques used. Chapter 4 which covers the actual implementation itself. Chapter 5 presents the results of the study, while Chapter 6 engages in a discussion about these findings. The report concludes with Chapter 7, which synthesizes the conclusions drawn from the research.

2 Theory

This chapter presents relevant background theory for a better understanding of this thesis.

2.1 t-SNE

t-SNE (t-distributed Stochastic Neighbor Embedding) is a technique designed to visualize high-dimensional data by mapping each data point to a two or three-dimensional space. This method is an enhanced version of the original Stochastic Neighbor Embedding, offering improved visualization capabilities. [2]

2.2 Text representation

In this thesis, text representation is addressed through two fundamental techniques: Tokenization and Word Embeddings. These methods are essential for processing and understanding text within Large Language Models (LLMs). Tokenization serves as the initial step in deconstructing text into manageable pieces, while word embeddings provide a way to translate these pieces into numerical forms.

2.2.1 Tokenization and tokens

Tokenization is the process of turning text into a list of "tokens," which can be characters, parts of words or whole words. The purpose of tokenization in natural language processing is to break down text into these manageable units, enabling machine learning models to process and analyze the text numerically. This crucial step transforms raw text into a structured form that algorithms can understand and manipulate, enabling effective data analysis and interpretation. [3]

2.2.2 Embeddings

Embeddings are a technique to convert text into numerical values to represent words, phrases, or entire documents as vectors in a vector space. This transformation captures not just the identity of textual elements but also their semantic and syntactic relationships, allowing similar words to have similar numerical representations. The purpose of embeddings is to reduce the high dimensionality of text data into a more manageable form, facilitating more effective computation and enabling machine learning models to perform complex tasks. [4]

Figure 2-1: Example showing how words relate to others in the numerical space [4].

As shown in figure 2-1, the embedding dimensions have been reduced to two dimensions using a t-SNE model for clearer visualization. In this visual representation, words are colored as blue, red, and green, with each color forming clusters. This clustering indicates that the embeddings capture and group words with similar meanings or sentiments closely together. For instance, positive words like 'good', 'amazing', and 'wonderful' are clustered in one area, while negative words such as 'bad', 'worse', and 'worst' are grouped in another area. This separation and clustering highlight how embeddings, through t-SNE, enable the visualization of complex semantic relationships within the data.

Two types of embedding methods are static embeddings and contextual embeddings. In static embedding, is each word assigned a fixed vector, meaning that no matter the context in which the word appears, it will always have the same embedding. This contrasts with contextual embeddings, where the embedding for a word can vary depending on its context within a sentence. Therefore, the same word may have different embeddings based on the surrounding words and the overall semantic meaning of the sentence. [4]

2.3 Large Language Models

Large Language Models (LLMs) such as Chat-GPT, PaLM and LLaMA are advanced transformer-based models characterized by their wide scale, typically containing hundreds of billions of parameters. These models are trained on a huge amount of text datasets, enabling them to develop a profound understanding of natural language and the ability to perform complex text generation tasks. [5]

2.3.1 Context size

Context size, also known as context window or length when referring to LLMs, is the maximum number of tokens the model can handle at one time. This limitation includes both the input tokens it receives and the output tokens it generates. The context size is crucial because it determines how much information the model can process when making predictions or generating text.

A larger context size can lead to a better understanding of the input and enable the generation of more descriptive and contextual outputs. This is particularly important for tasks that require a deep understanding of long texts, such as summarizing lengthy documents or answering complex questions. [6]

2.3.2 Prompting

Prompting is a crucial technique used for interacting with Large Language Models (LLMs), where users provide a "prompt", an input string that contains instructions or tasks for the LLM to process. Common strategies include:

Zero-Shot Prompting: LLMs utilize their training data to respond to new queries without needing specific examples within the prompt.

In-Context Learning (Few-Shot Learning): This approach involves presenting the LLM with multiple input-output pairs, which helps guide the model to generate responses by mimicking these examples.

Chain-of-Thought (CoT): Prompts under this strategy contain detailed reasoning steps, assisting the LLM in articulating a step-by-step reasoning process for more complex problem-solving.

Multi-Turn Instructions: This technique engages LLMs in an ongoing dialogue, allowing each response to influence subsequent prompts. It is particularly effective in interactive applications such as chatbots or complex task navigation.

These prompting strategies enhance the LLMs ability to generate accurate and contextually relevant responses across a variety of tasks. [6]

2.3.3 Transformer

Transformers are typically the architecture referring to when discussing LLMs. Transformers use positional encodings to maintain sequence order and stacked self-attention layers to understand context and play a crucial role for LLMs. The following sections will describe the structure of the transformer shown in figure 2-2. [7]

First is the input query and input embedding. The input query is the instructions the user has written for the LLM but as tokens, and the input embedding is the embedded versions of these tokens, transforming them into numerical vectors that the model can process.

Next is the positional encoding. This step creates an understanding of the order in the input, which is crucial because the model processes tokens in parallel, meaning that the order of tokens is not naturally preserved in the process. These positional encodings are typically calculated using sine and cosine mathematical functions. Each position in the sentence receives unique sine and cosine values based on its sequence by the following method.

$$PE\,(pos,\ 2i) \ = \ \sin\left(pos \ \div \ 10\,000^{2i \div d}\right) \quad (2.1)$$

$$PE\,(pos,\ 2i \ + \ 1) \ = \ \cos\left(pos \ \div \ 10\,000^{2i \div d}\right) \quad (2.2)$$

Where *pos* is the position in the sentence, i is the dimension and d are the number of dimensions, which are the same amount as the input embeddings. For more information about this, see source. These values are then combined with the word embeddings, providing context about word positions that the transformer uses to understand the text better. This method ensures that despite processing words in parallel, the model retains information about the order of words. [7]

Next step is the encoder, following the input stage, consisting of N identical layers (number of iterations). Each layer is equipped with two sub-layers, a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. The multi-head self-attention mechanism's purpose is to receive information about every token in the given input, based on the positions and the relations to the other tokens in the input. Meanwhile, the feed-forward network enhances the token representations refined by the attention mechanism, applying

transformations that integrate learned contextual information into each token's embedding. [7]

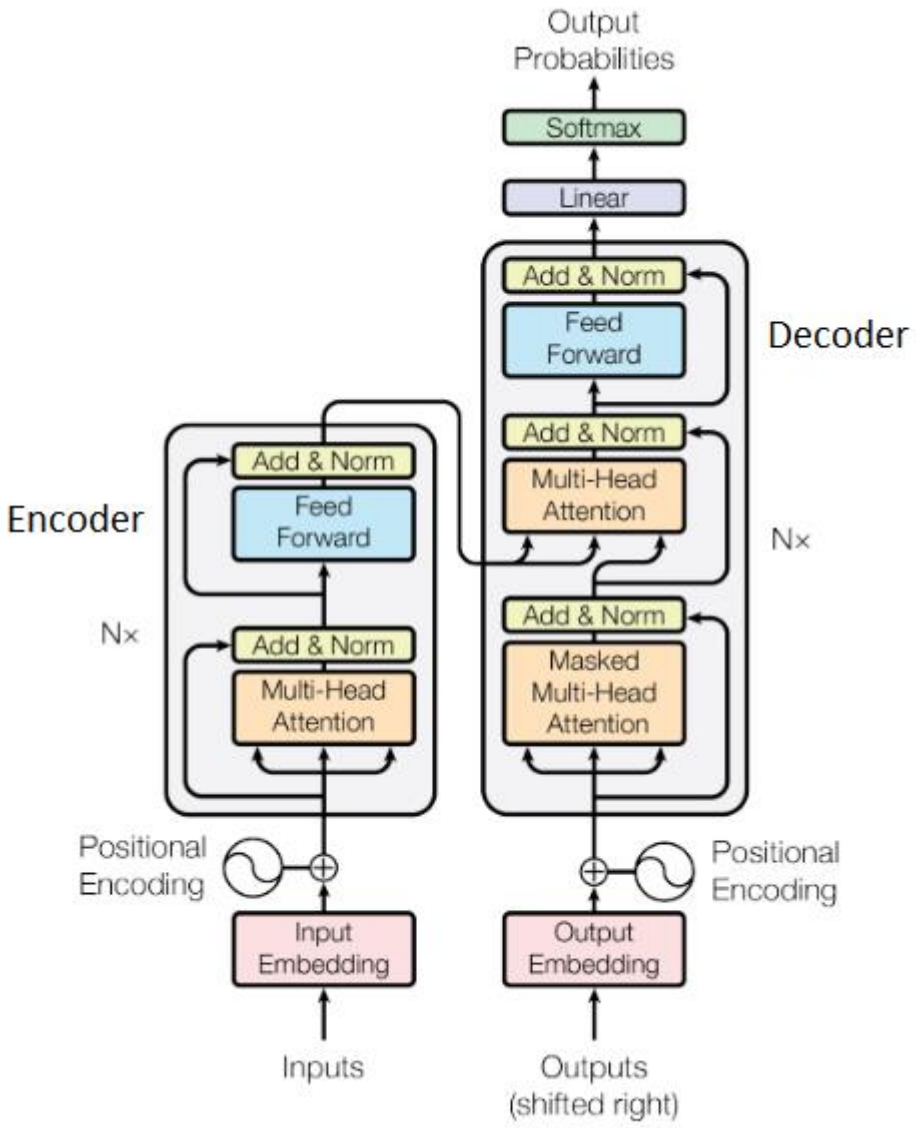

Figure 2-2: Image containing the structure of a transformer. This image has been slightly modified to clarify the components by adding "Encoder" and "Decoder". [7]

Similar to the encoder, the decoder contains N layers (number of iterations), each mirroring the encoder's structure with an additional sub-layer. The initial sub-layer in each decoder layer features a masked self-attention mechanism, which ensures that the generation of each

token is influenced only by preceding tokens, thereby preserving the autoregressive property of the decoder. Following this, the encoder-decoder attention layer integrates the context from the entire input sequence by focusing on the most relevant parts of the encoded embeddings. This layer crucially aligns the decoder's output with the encoder's input, facilitating accurate and context-aware token generation. Finally, the feed-forward network in the decoder operates similarly to that in the encoder, further processing each token's data to produce the final output sequence. [7]

The final step of the transformer model's process is the output generation, which begins with a linear layer. This layer takes the output embeddings from the decoder, which represent the current token's features informed by both the encoding and decoding phases. The linear layer transforms these embeddings into a vector whose dimension matches the size of the model's vocabulary, producing a set of scores for each possible token. These scores are then passed to a SoftMax layer, which converts them into probabilities, reflecting the likelihood of each token being the next in the sequence. Based on these probabilities, the model selects the most appropriate token using its selection strategy. This process repeats for each token until the model generates an end-of-sequence token, signaling the completion of the output. [7]

2.3.4 Scaling laws

Scaling laws are an important component of the optimization process for LLMs. They enable researchers and developers to predict the potential performance of these models based on various input parameters. By systematically analyzing how changes in model size, data volume, and computational resources affect model outcomes. Scaling laws provide a predictive framework that can guide strategic decision-making during the development and training phases. [5]

2.4 Cosine similarity

Cosine similarity is used to measure the similarity by calculating the cosine angle between two vectors, the vectors need to have the same number of dimensions. This is done by calculating the dot product of the vectors.

$$a \cdot b = |a|\,|b|\,cos\theta \quad (2.3)$$

$$cos\theta = \frac{a \cdot b}{||a||\,||b||} \quad (2.4)$$

The cosine of the angle θ between the vectors can then be calculated by dividing the dot product by the product of the magnitudes. This result, which ranges from -1 to 1, indicates how similar the two vectors are. A value of 1 means the vectors are perfectly aligned (the angle is 0 degrees), indicating maximum similarity. A value of 0 indicates that the angle is 90 degrees, and a value of -1 indicates that the vectors are diametrically opposed (the angle is 180 degrees). [8]

2.5 Accuracy and F1-Score

F1-Score is a calculation that utilizes precision and recall. Precision measures the accuracy for the positive predictions. This is done by counting how many times the positive prediction was correct and incorrect and then dividing the correct amount by the sum of the correct and incorrect. [9]

$$\textbf{Precision} = \frac{TP}{TP + FP} \quad (2.5)$$

And recall is used to measure how well the model can identify the positive cases. This is calculated by dividing the true positive (correct positive) by the sum of true positive and false negative (unidentified correct). [9]

$$\textbf{Recall} = \frac{TP}{TP + FN} \quad (2.6)$$

The F1 Score combines the precision and the recall to a single value by either dividing 2 by the sum of 1 divided by the precision and 1 divided by the recall.
Can also be calculated by dividing 2 multiplied by the precision and recall with the sum of precision and recall [10].

$$\textbf{F1 Score} = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} \quad (2.7)$$

$$\textbf{F1 Score} = \frac{2 \cdot precision \cdot recall}{precision + recall} \quad (2.8)$$

This will produce a 2x2 metric that represents the true positive (TP), false positive (FP), false negative (FN) and true negative (TN). Where true means that the model identified is correct and false that it is incorrect.

The first row in the metric will represent the predicted positive where the first column is the amount of correctly and the second column is the amount of incorrect. Same with the second row that includes the negative, the first column is the correct negative and the second column is the incorrect negative. [9]

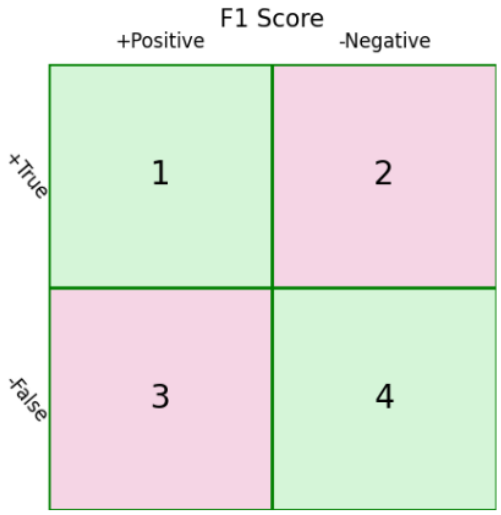

Figure 2-3: A 2x2 matrix showing how the F1 score will be presented.

2.6 Related work

The following subsections present related work and their similarities with this thesis.

2.6.1 Extracting Financial Data from Unstructured Sources: Leveraging Large Language Models

The study conducted by Huaxia Li, Haoyun Gao, Chengzhang Wu, and Miklos A. Vasarhelyi addresses the significant challenge of extracting financial data from unstructured sources. Utilizing LLMs, their research introduces a framework designed specifically for automated data extraction from PDF-formatted documents. This framework employs a combination of text mining and prompt engineering techniques.

The developed framework was tested on governmental annual reports and corporate ESG reports, showcasing its effectiveness in parsing complex financial documents. The framework demonstrated a high

accuracy rate, achieving an average of 99.5% in initial tests and maintaining an overall accuracy of around 96% in subsequent larger-scale tests. [11]

Both this study and the current thesis work with PDF files and aim to harness the capabilities of large language models to extract data from unstructured sources, highlighting a shared objective in advancing data extraction methodologies using AI technologies.

3 Methodology

This chapter presents the methodology used in this thesis including the scientific / project method descriptions and evaluation method.

3.1 Scientific method description

In this thesis, quantitative methods will be employed to systematically investigate and analyze the performance. Quantitative research is ideal for this study as it allows for precise measurement for the number of identified references and to compare the output to the expected output. To further address the research problem, a mix of comparative and experimental research designs will be utilized alongside quantitative methods.

This thesis builds upon the research "Efficient Automated Processing of the Unstructured Documents Using Artificial Intelligence: A Systematic Literature Review and Future Directions" (2021), which highlights the significant potential of AI-based approaches for extracting useful information from unstructured documents. Their systematic literature review identifies key challenges and gaps in the existing techniques, particularly in handling complex document layouts and the need for high-quality datasets. [12]

The first research question is how the accuracy differs between the models when comparing the number of identified references. To answer this needs all the implementation steps to have been implemented:

Step 1 Extracting and dividing text: This includes processes like extracting the text from the PDF and dividing the text into sections.

Step 2 Generate embeddings: This step generates embeddings for each section.

Step 3 Cosine similarity: This step performs cosine similarity on the embedded sections to identify and extract sections that could contain the external references.

Step 4 Large Language Model: The extracted sections from the cosine similarity are then sent into a LLM to extract the external references as a structured dataset.

When all the implementation steps are complete, they will be tested with three different documents. These extracted datasets are then compared between the models to evaluate the differences.

The second research question, similar to the first one, needs the implementation steps to be complete. This question compares the quality of the identified references between the models by setting either "partial match" or "exact match" on a correctly identified reference. This measures the number of identified references that contain the expected information.

3.2 Project method description

This thesis is divided into eight phases, the initial phase is a pre-study and consists of method discovery and planning. This is followed by phases 2 through 6 and are the implementation phases. And last phase 7 and 8 which gather the data for the results and discuss the result.

3.2.1 Phase 1: Pre-study

The initial phase of this thesis, the pre-study, includes method discovery and planning. This step is fundamental to the entire research project as it establishes the theoretical and practical work that will be applied. This was done by a comprehensive review of existing methods in the field of artificial intelligence, specifically focusing on Large Language Models (LLMs) and their applications in data extraction.

3.2.2 Phase 2: Extract text

The second phase signifies the onset of the practical implementation, focusing on utilizing libraries to extract text from PDF documents. This phase involves selecting the appropriate tools that reliably convert PDF content into manageable text while preserving the original formatting to the greatest extent possible.

3.2.3 Phase 3: Divide text

After the text is extracted, it must be divided into segments and every segment will later be embedded. The division strategy could be based on natural language cues such as chapters, headings, paragraphs, sentences or just a fixed size of characters. This segmentation is critical as it forms the basic units of text that will be compared in the similarity search.

3.2.4 Phase 4: Generate embeddings

This phase involves the embedding of text segments obtained from the previous step. During generation of embeddings, is each text segment transformed into embeddings, also known as vectors. These embeddings are essentially lists of numbers, ranging between -1 and 1, where each number represents a dimension of the text's features captured by the model.

The dimensionality of these embeddings, essentially the number of numbers in each embedding, is determined by the selected model. Higher-dimensional embeddings can capture more detailed information about the text but at the cost of increased computational complexity. Each embedding essentially serves as a numerical fingerprint of a text segment, taking semantics and possible context if the model is trained for it in account.

Properly executed embeddings is critical as it converts qualitative text data into a quantitative form that can be systematically compared and analyzed in the next phase of similarity search. This transformation is what allows the algorithms of the next steps to perform comparisons between different text segments.

3.2.5 Phase 5: Similarity search

Phase 5 focuses on the similarity search, a critical step in uncovering relationships and patterns in the dataset. In this phase, the embeddings generated are compared using similarity metrics to determine how closely the segments of text are related in terms of their semantic or contextual content.

This process involves algorithms that can efficiently handle and process large matrices of embeddings to compute the similarity scores.

The results from this search can be used to cluster segments that share similarities.

3.2.6 Phase 6: LLM

Phase 6 represents the end of the implementation process, where the segments identified as similar or relevant by the similarity search are further analyzed using an LLM.

The process begins by inputting the related text segments into the LLM. Depending on the requirements, the LLM can be utilized for various tasks such as extracting data and relationships, summarizing information, answering questions based on the text, or even generating new text based on the learned context. But as mentioned before, this thesis will focus on extracting data.

3.2.7 Phase 7: Gather results

Upon completion of the implementation steps, the results from the embeddings, similarity search and LLMs are collected. During the embedding and similarity search phases, the vectors are split into categories to visually be able to see how the vectors are placed. In the LLM phase, the outputs generated by the LLMs are listed. To visualize the data, all vectors are reduced to two dimensions using a t-SNE model and plotted within the context of their respective documents. The outputs from the LLMs are compared against the expected outputs, and the precision, recall, and F1 scores are calculated to measure the performance and accuracy of the LLMs in identifying references correctly.

3.2.8 Phase 8: Evaluation and discussions

After the results have been gathered from the documents and the LLMs, they are compared between the different documents and a discussion about the results and the whole process is made.

4 Implementation

Figure 4-1 provides an overview of the entire process, beginning with text extraction from a PDF. The extracted text is divided into sections, which are then converted into embeddings. These embeddings represent both the text itself and the relationships within the text as numerical values. A similarity search is then conducted on these embeddings, using a transformed input. Texts that align with the search input are extracted and fed into a LLM, which is tasked with specific functions such as extracting external references or charts. The final output is a list of the data identified by the LLM.

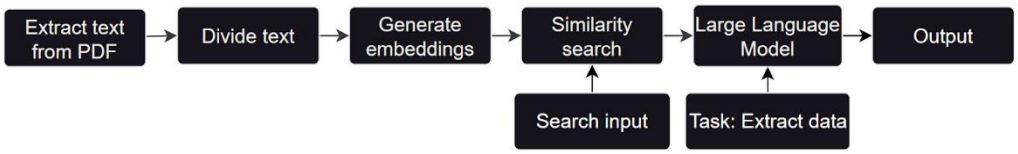

Figure 4-1: An overview of the entire process.

Figure 4-2 Also shows an overview of the entire process, but this figure shows how the data flows in the process. It starts with the entire text from the PDF, the text is then divided into sections. Embeddings are then generated for each section. A cosine similarity search is then performed on each section with a set input transformed into embeddings to identify which of the sections that could contain the requested information. The texts of the embeddings that matched in the cosine similarity search are then sent to a LLM which is instructed to extract the data. For the complete source code, see Appendix A.

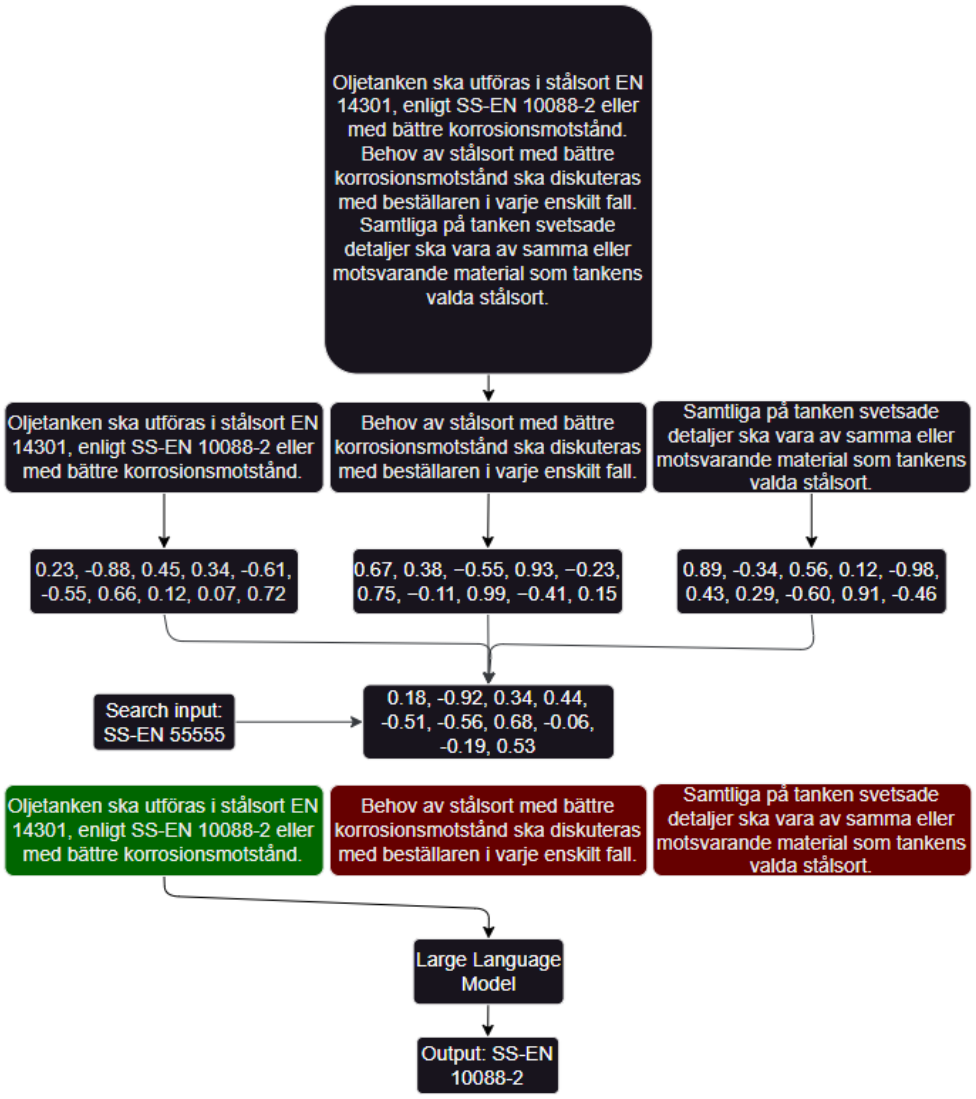

Figure 4-2: An overview of the entire process, showing how the data is processed in every step.

4.1 Extract text

The initial step involves extracting text from a PDF file using the PyPDF2 library [13]. This will take all the text from the PDF into one string.

4.2 Divide text

Once the text is extracted, it is divided into sections based on sentences, utilizing the NLTK library [14].

4.3 Generate embeddings

The next step involves generating embeddings for each section using OpenAI's "text-embedding-3-small" model which produces embeddings with 1536 dimensions. To use OpenAI's models, users must have credits in their account and initialize an API key within the code. [15] [16]

4.4 Cosine similarity

This stage involves performing cosine similarity analysis between a search input (once transformed into embeddings) and the embeddings generated in the previous step. Embeddings that result in a similarity score above the chosen threshold (45%) are selected for further analysis. This was done with sklearn's cosine similarity function. [17]

4.5 Large Language Model

In the final step, the selected sentences are inputted into an LLM. The LLM is tasked with extracting specific data from these sentences based on given instructions. The prompt given to the LLMs to extract external references from standard was the following, it is in Swedish since the targeted documents are in Swedish:

“Du är en hjälpsam assistent som har i uppdrag att hitta externa referenser från meningar. Dessa externa referenser är formaterade som sifferkod eller standarder som vanligtvis erkänns i tekniska och vetenskapliga dokument, till exempel 'SS', 'EN' eller 'ISO' eller en kombination av dem, men det finns även fler. Så din uppgift är följande: hitta alla externa referenser. Det kan vara så att det också finns vilken del av standarden som den refererar till och kan göras på två sätt, ena är genom ett '-x' där 'x' är en siffra, och det andra är genom att skriva 'del x' där 'x' är en siffra. Om du hittar en referens som skrivs med 'del' sättet så omvandla det till '-x' och inkludera det i referensen. Det kan också vara så att referensen innehåller årtal på följande sätt ':xxxx' där 'xxxx' är ett årtal. Inkludera detta i referensen om det finns. Sen ta bort referenser till allmänna publikationer, företagsinterna dokument från 'SSG' eller några hyperlänkar. Nästa steg är att ta bort referenser som du har hittat flera gånger, detta inkluderar då hittade referenser som har samma kod men del nummer och årtal får vara olika, ta bort duplikat genom att exkludera de referenser som innehåller minst information. Svara enbart med en lista på de hittade externa referenserna, om det skulle vara så att du inte hittade någon svara med ‘ ’ ”

But here is the English version: “You are a helpful assistant tasked with finding external references from sentences. These external references are formatted as numeric codes or standards commonly recognized in technical and scientific documents, such as 'SS', 'EN' or 'ISO' or a combination of them, but there are more. So your task is the following: find all external references. It may also include what part of the standard it refers to and can be done in two ways, one is through a '-x' where 'x' is a number, and the other is by writing 'part x' where 'x' is a number. If you find a reference written in the 'part' way, convert it to '-x' and include it in the reference. It may also be that the reference contains a year in the following way ':xxxx' where 'xxxx' is a year. Include this in the reference if available. Then remove references to public publications, internal company documents from 'SSG' or any hyperlinks. The next step is to remove references that you have found multiple times, this includes references found that have the same code but the part number and year may be different, remove duplicates by excluding the references that contain the least information. Reply only with a list of the external references found, if you did not find any reply with ’ ’ ”.

The chosen LLMs were gpt-4-turbo-2024-04-09, this model has a context window of 128,000 tokens and is trained on data up to December 2023. And gpt-3.5-turbo-0125 which has a context size of 16,385 tokens. Both also used the configuration temperature 0.2 and top_p 0.1. A lower temperature configuration reduces the randomness when given an output and top_p reduces the number of tokens taken into consideration when producing the output [18]. That is important since the task is to extract references and only focus on that. [19]

4.6 Evaluation setup

To evaluate the generated embeddings a t-SNE model will be used to reduce the number of dimensions. This enables the embeddings to be viewed in a two-dimensional diagram to see how the sentences that contain the external references are placed compared to the sentences without references. The cosine similarity will also use this kind of visualization of the embeddings, but this diagram will also show what sentences that have been matched with the search input and where the input embeddings have been placed on the diagram. And the LLMs will then be evaluated by comparing the output to the correct answers, this is done by measuring the F1 score by counting the correct (TP), incorrect

(FP), missed (FN) and number of tokens (TN). And to evaluate the quality of the output, each identified reference will get a tag either "correct" or "partially correct". This will be used to measure what percentage of the identified references that is correct or partially correct.

5 Results

This chapter outlines the results obtained in this thesis, illustrated through various diagrams that represent the embeddings generated for each document, the results of the cosine similarity search, and the final outputs from the LLMs.

5.1 Generated embeddings

The following diagrams represent the embeddings generated on each sentence in its document, to reduce the dimensions a t-SNE model was used. The blue dots are sentences consisting of at least one external reference while the red text does not contain any external reference.

Figure 5-1 shows the generated embeddings for the first document. This diagram displays a total of 458 data points, of which 70 contain external references and 388 do not.

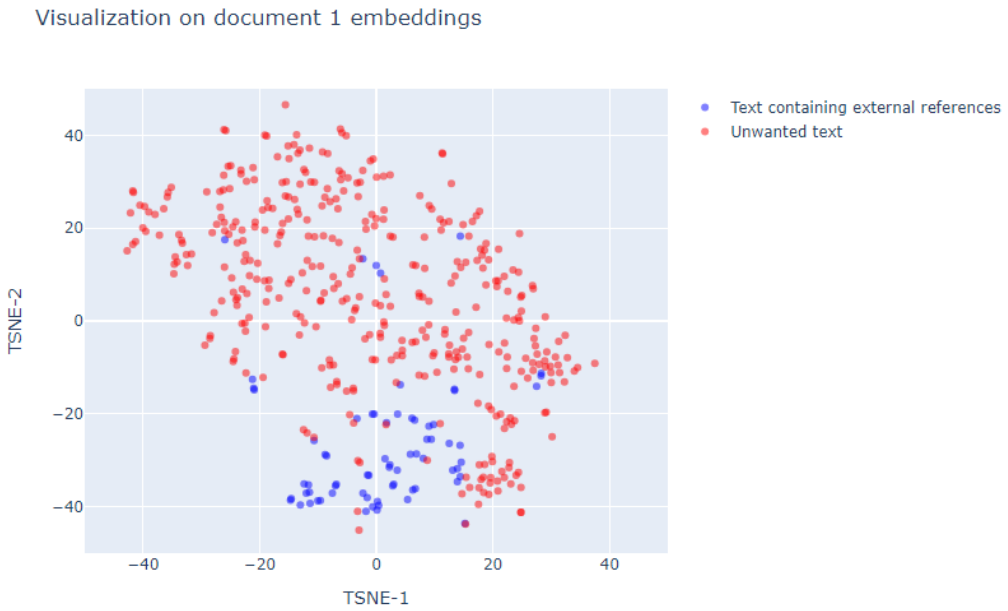

Figure 5-1: A visualization of the embeddings for the first document.

The next figure, figure 5-2 represents the generated embeddings on the second document. Presented in this figure are the embeddings for 313 sentences, with 29 featuring external references and 284 lacking them.

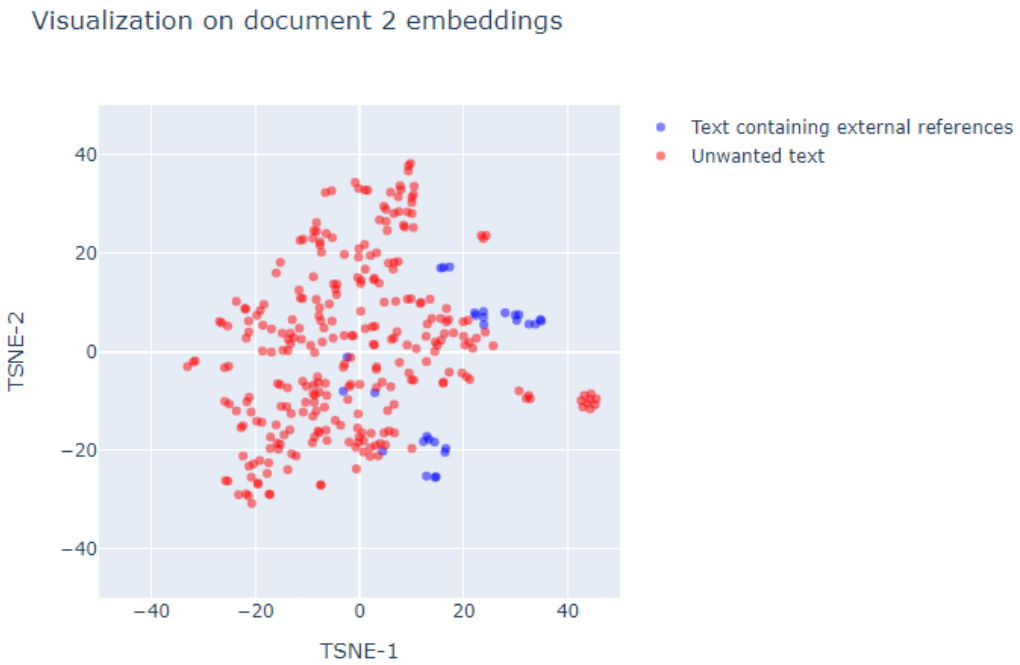

Figure 5-2: A visualization of the embeddings for the second document.

The last diagram of this subchapter, figure 5-3. This diagram contains 369 data points. Where, 30 sentences include external references and 339 sentences do not.

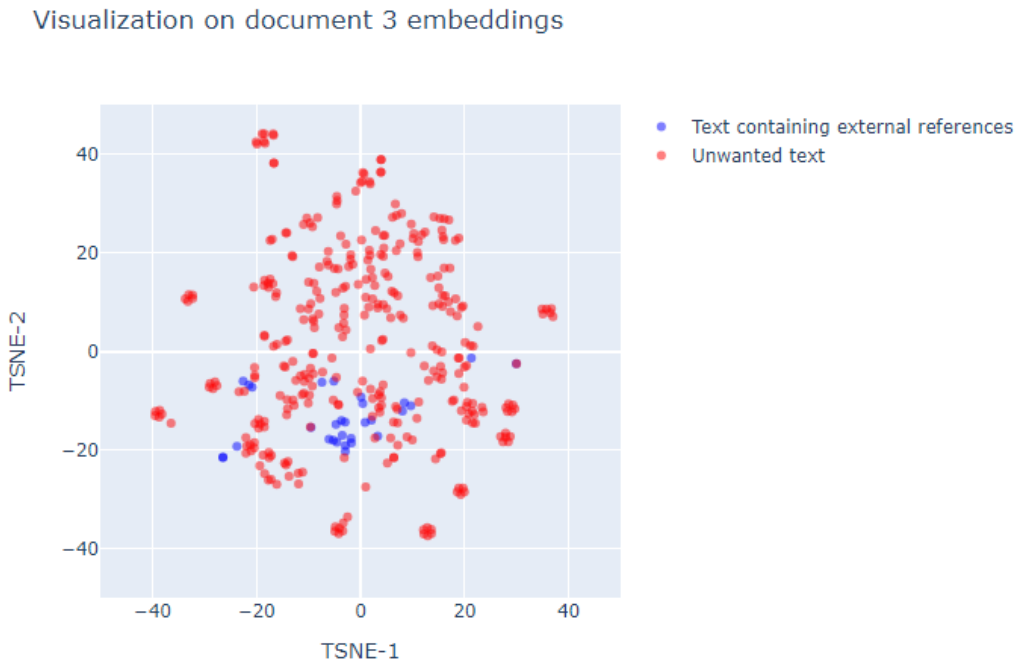

Figure 5-3: A visualization of the embeddings for the third document.

5.2 Cosine similarity

The diagrams shown in this chapter are based on the embeddings shown in the previous chapter but include one additional data point. These diagrams have also been made with the help of a t-SNE model. The similarity search consists of an input "Enligt standard SS EN ISO 5555" in English "according to standard SS EN ISO 5555" and a threshold of 45%. The blue dots are sentences that contain at least one external reference and matched with the input embeddings, the turquoise dots are also sentences that contain at least one external reference but were not matched with the input embedding. While the red dots are sentences that do not consist of any external reference but were matched with the input embedding, and the pink dots are sentences that do not contain any external references and did not match with the input. Meaning, it's the blue and red dots that will be further analyzed in the next step.

Figure 5-4 shows the embeddings for the first document after a cosine similarity has been performed. The search ended up removing 243 sentences that do not contain any external reference and 4 that do contain external reference. Which means 211 sentences were matched with the search, see table 1.

Table 1: Overview of the results for cosine similarity on the first document

	Initial amount of sentences	Amount of sentences after cosine similarity	Amount of removed sentences
Sentences containing external references	70	66	-4
Sentences not containing external references	388	145	-243
Total	458	211	-247

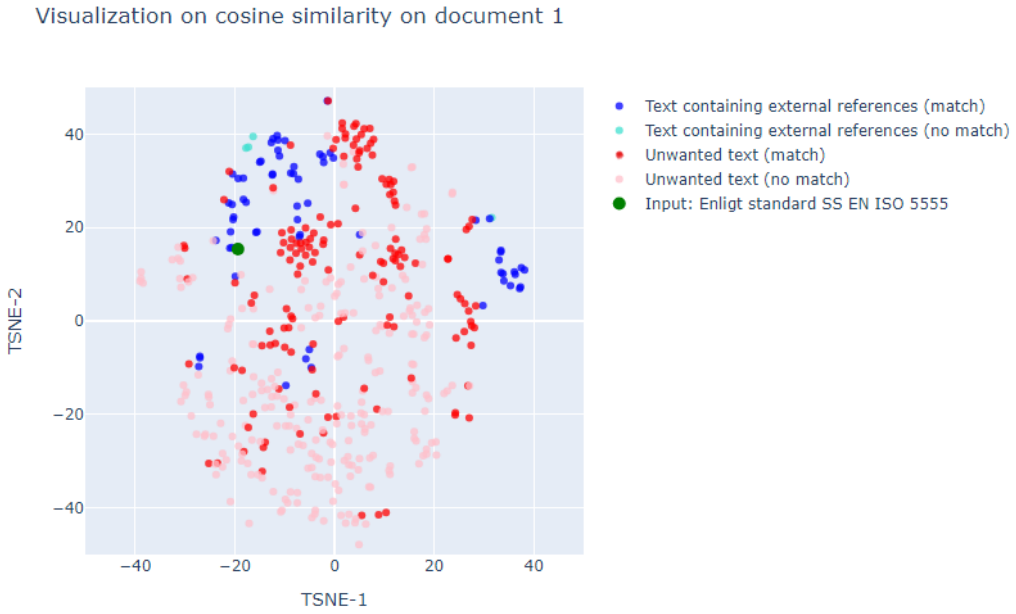

Figure 5-4: A visualization on the embeddings after cosine similarity has been performed on the first document.

Figure 5-5 presents the embeddings for the second document after the cosine similarity was performed. In this document did the search remove

a total of 202 sentences where 195 did not contain any external reference and 7 did, see table 2.

Tabel 2: Overview of the results for cosine similarity on the second document

	Initial amount of sentences	Amount of sentences after cosine similarity	Amount of removed sentences
Sentences containing external references	29	22	-7
Sentences not containing external references	284	89	-195
Total	313	111	-202

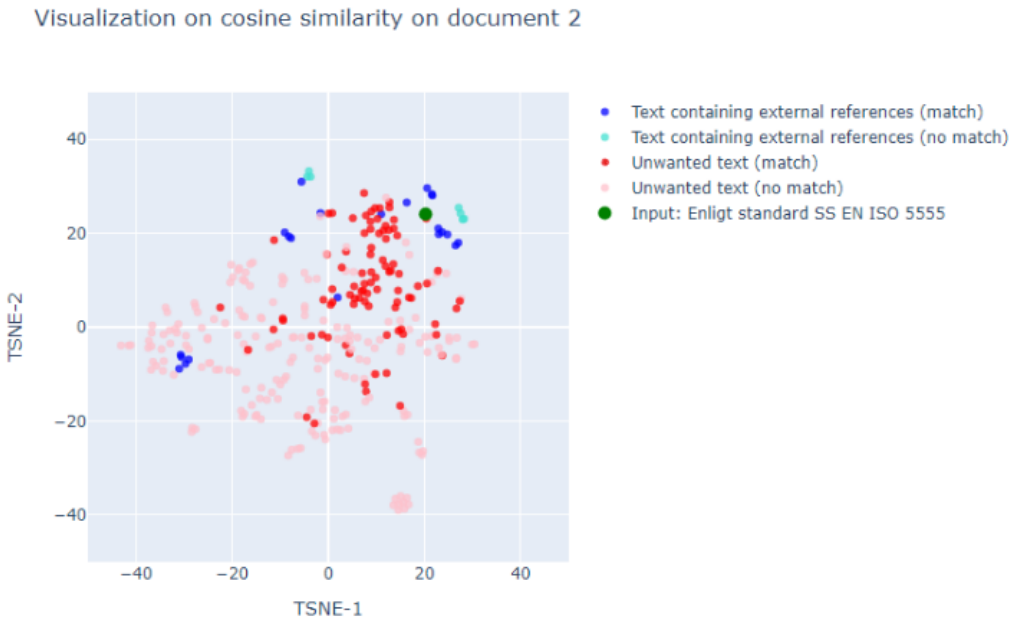

Figure 5-5: A visualization on the embeddings after cosine similarity has been performed on the second document.

Figure 5-6 show the results after the cosine similarity was performed on the third document. That ended up removing a total of 309 sentences where 302 do not contain external references and 7 did, see table 3.

Tabel 3: Overview of the results for cosine similarity on the third document

	Initial amount of sentences	Amount of sentences after cosine similarity	Amount of removed sentences
Sentences containing external references	30	23	-7
Sentences not containing	339	37	-302

external references			
Total	369	60	-309

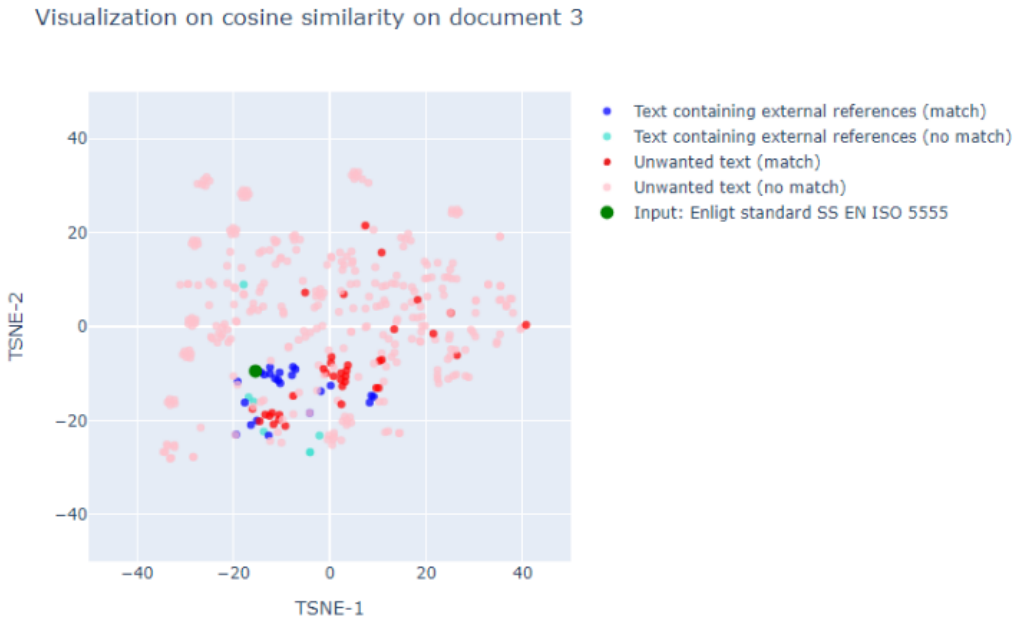

Figure 5-6: A visualization on the embeddings after cosine similarity has been performed on the third document.

5.3 Large Language Model

The following three sections present the results from the models on each document. Each section contains a table with five columns, the first one is the correct answer, and this is followed by two pairs where the first one in the pair is the model output and the second one is the quality of the predicted output. Meaning it will get tagged either, match, partial match or incorrect. These tags means that the predicted contains as much or more information as the expected output "exact match", the predicted output contains less information that the expected output "partial match", "incorrect" if the prediction was incorrect, "Not found" if the reference was not identified or "Removed by cosine similarity" if removed by the previous step. The tags that are associated with an

identified reference also have a color associated with it, “exact match” green, “partial match” orange and “incorrect” red.

Each row in the tested model's column represents an identified reference and the one in the first column is the expected answer. An empty row on any of the model’s columns but not on the first means that the reference was not identified. On the bottom of the table can there be empty rows on the first column but not on the tested model's columns. This means that the model has identified an incorrect reference.

Each of the following three sections will also have two 2x2 matrixes that were used to measure the F1 score for each model. The true negative (NT) value in these matrixes is set to the number of tokens in the text extracted with the cosine similarity measured by OpenAI's tokenizer [20].

5.3.1 First document

Table 4 presents the output from the LLMs for the first document. It shows that gpt-4-turbo managed to identify 30 out of 31 (≈97%) correctly but also 4 incorrect references. This gives the gpt-4-turbo a precision of ≈0.88, a recall of ≈0.97 and a F1 score of ≈0.92. And 29 out of the 30 (≈97%) identified references were an exact match with the expected output. While the gpt-3.5-turbo managed to identify 29 out of 31 (≈87%) correctly with 1 incorrect prediction. This gives the gpt-3.5-turbo a precision of ≈0.97, a recall of ≈0.92 and a F1 score of 0.95. And 21 out of the 29 (≈72%) identified were an exact match with the expected output.

Table 4: A comparison between the correct and model’s outputs on the first document.

Expected output	gpt-4-turbo-2024-04-09	gpt-4-turbo-2024-04-09 quality	gpt-3.5-turbo-0125	gpt-3.5-turbo-0125 quality
ISO 23309	ISO 23309	Exact match	ISO 23309	Exact match
SS-ISO 3320	SS-ISO 3320	Exact match	SS-ISO 3320	Exact match

SS-ISO 4393	SS-ISO 4393	Exact match	SS-ISO 4393	Exact match
SS-ISO 4406:2021	SS-ISO 4406:2021	Exact match	SS-ISO 4406:2021	Exact match
SS-ISO 4413: 2010	SS-EN ISO 4413:2010	Exact match	SS-EN ISO 4413:2010	Exact match
SS-ISO 6022: 2006	SS-ISO 6022:2006	Exact match	SS-ISO 6022	Partial match
SS-ISO 6162-1	SS-ISO 6162-1	Exact match	SS-ISO 6162-1	Exact match
SS-EN ISO 8434-1:2007	SS-EN ISO 8434-1:2007	Exact match	SS-EN ISO 8434-1	Partial match
SS-EN 837-1	SS-EN 837-1	Exact match	SS-EN 837-1	Exact match
SS-EN 837-3	SS-EN 837-3	Exact match	SS-EN 837-3	Exact match
SS-EN 853	SS-EN 853	Exact match	SS-EN 853	Exact match
SS-EN 856	SS-EN 856	Exact match	SS-EN 856	Exact match
SS-EN 857	SS-EN 857	Exact match	SS-EN 857	Exact match
SS-EN 10088-2:2014	SS-EN 10088-2:2014	Exact match	SS-EN 10088-2	Partial match

SS-EN 10088-3:2014	SS-EN 10088-3:2014	Exact match	SS-EN 10088-3	Partial match
SS-EN 10204: 2005	SS-EN 10204:2005	Exact match	SS-EN 10204	Partial match
SS-EN 10216-5: 2013	SS-EN 10216-5:2013	Exact match	SS-EN 10216-5	Partial match
SS-EN 60204-1	SS-EN 60204-1	Exact match	-	Not found
SS-EN ISO 12100:2010	SS-EN ISO 12100:2010	Exact match	SS-EN ISO 12100:2010	Exact match
SS-ISO 8132: 2016	SS-ISO 8132:2016	Exact match	SS-ISO 8132:2016	Exact match
SS-ISO 16889: 2011	SS-ISO 16889:2011	Exact match	SS-ISO 16889	Partial match
SS-EN 12760: 2016	SS-EN 12760	Partial match	SS-EN 12760	Partial match
SS-EN 12627	SS-EN 12627	Exact match	SS-EN 12627	Exact match
SS-EN 13480-5	SS-EN 13480-5	Exact match	SS-EN 13480-5	Exact match
SS-EN 10305-4	SS-EN 10305-4	Exact match	SS-EN 10305-4	Exact match
AFS 2005:16	AFS 2005:16	Exact match	AFS 2005:16	Exact match

AFS 2011:19	AFS 2011:19	Exact match	AFS 2011:19	Exact match
AFS 2017:3	AFS 2017:3	Exact match	AFS 2017:3	Exact match
DIN 2353	DIN 2353	Exact match	DIN 2353	Exact match
DIN 3015-2	-	Not found	-	Not found
SAE J518	SAE J518/1	Exact match	SAE J518/1	Exact match
-	SSG 2113	Incorrect	SSG 3800	Incorrect
-	SSG 4700	Incorrect	-	-
-	SSG 7571	Incorrect	-	-
-	EN 1.4401	Incorrect	-	-

Figure 5-7 and figure 5-8 presents the result from the models as a summarized version for the first test document and does not take detail into account. The true positive value represents the correctly identified references, the false positive is the incorrectly identified references and the false negative is the references that were not included in the output but should have been. The true negative value is set to the number of tokens in the text that was processed by the LLM. Figure 5-7 summarizes the result for gpt-4-turbo-2024-04-09 gaining 30 true positive, 4 false positive, 1 false negative and 14238 true negative.

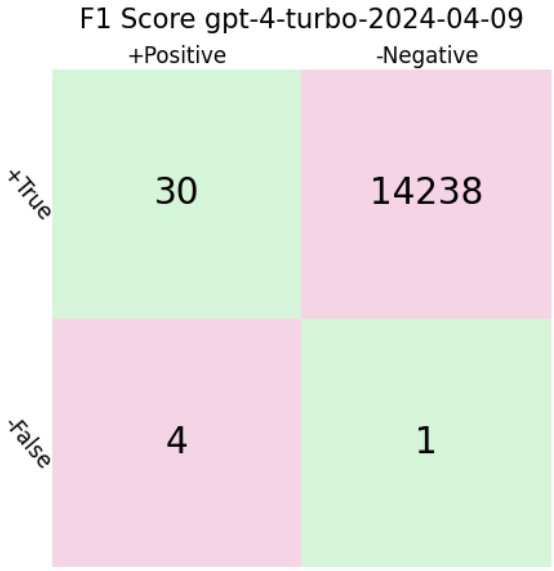

Figure 5-7: A 2x2 matrix showing how the model gpt-4-turbo predicted for the first document.

Figure 5-8 shows the result from the gpt-3.5-turbo-0125 model. Gaining 29 true positive, 1 false positive, 2 false negative and 14238 true negative since the same text has been processed.

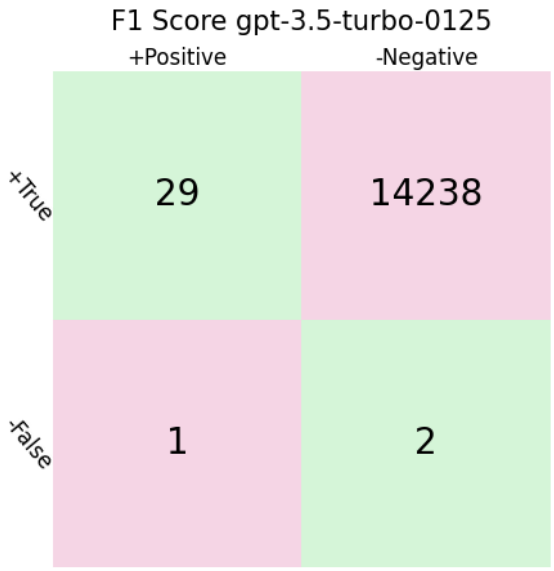

Figure 5-8: A 2x2 matrix showing how the model gpt-3.5-turbo predicted for the first document.

5.3.2 Second document

Table 5 presents the output from the LLMs for the second document. The three references that were not identified because they were removed by the cosine similarity which means that gpt-4-turbo managed to identify 10 out of 10 (100%) correctly with 0 incorrect references, see figure 5-9. This gives the gpt-4-turbo a precision, recall and F1 score of 1. And 10 out of the 10 (100%) identified references are an exact match with the expected output. While the gpt-3.5-turbo managed to identify 10 out of 10 (100%) correctly with 1 incorrect prediction, see figure 5-10. This gives the gpt-3.5-turbo a precision of ≈0.91, a recall of 1 and a F1 score of ≈0.95. And 10 out of the 10 (100%) identified were an exact match with the expected output.

Table 5: A comparison between the correct and model's outputs on the second document.

Expected output	gpt-4-turbo-2024-04-09	gpt-4-turbo-2024-04-09 quality	gpt-3.5-turbo-0125	gpt-3.5-turbo-0125 quality
SS-ISO 10816-3	SS-ISO 10816-3	Exact match	SS-ISO 10816-3	Exact match
SS-ISO 10816-6	SS-ISO 10816-6	Exact match	SS-ISO 10816-6	Exact match
SS-ISO 10816-7	SS-ISO 10816-7	Exact match	SS-ISO 10816-7	Exact match
SS-ISO 20816-1	SS-ISO 20816-1	Exact match	SS-ISO 20816-1	Exact match
SS-ISO 20816-2	-	Removed by cosine similarity	-	Removed by cosine similarity
SS-ISO 20816-4	-	Removed by cosine similarity	-	Removed by cosine similarity
SS-ISO 20816-5	-	Removed by cosine similarity	-	Removed by cosine similarity
ISO 21940-11	SS-ISO 21940-11	Exact match	ISO 21940-11	Exact match
ISO 21940-31	ISO 21940-31	Exact match	ISO 21940-31	Exact match
ISO 21940-32	ISO 21940-32	Exact match	ISO 21940-32	Exact match

SS-ISO 11342	SS-ISO 11342	Exact match	SS-ISO 11342	Exact match
ISO 14694	ISO 14694	Exact match	ISO 14694	Exact match
ISO 8528-9	ISO 8528-9	Exact match	ISO 8528-9	Exact match
-	**-**	**-**	SSG 7640	Incorrect

Figure 5-9 and figure 5-10 summarizes the outputs shown in table 5 but does not take detail into account. Figure 5-9 shows a summarized result for gpt-4-turbo-2024-04-09, gaining 10 true positive, 0 false positive, 0 false negative and 9238 true negative.

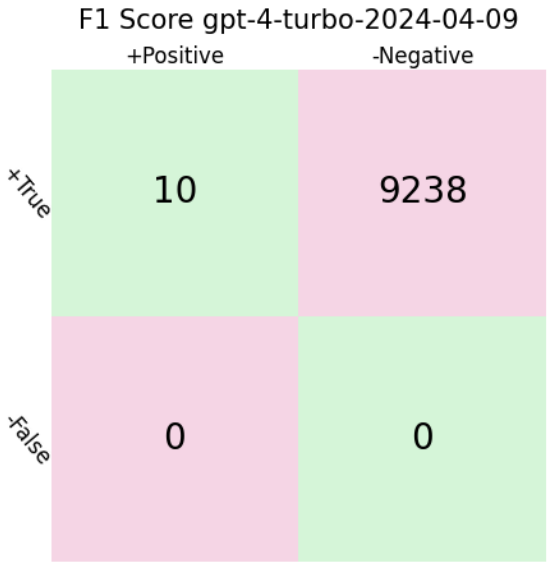

Figure 5-9: A 2x2 matrix showing how the model gpt-4-turbo predicted for the second document.

And figure 5-10 present a summarized result for the gpt-3.5-turbo-0125 model, achieving 10 true positive, 1 false positive, 0 false negative and 9238 true negative.

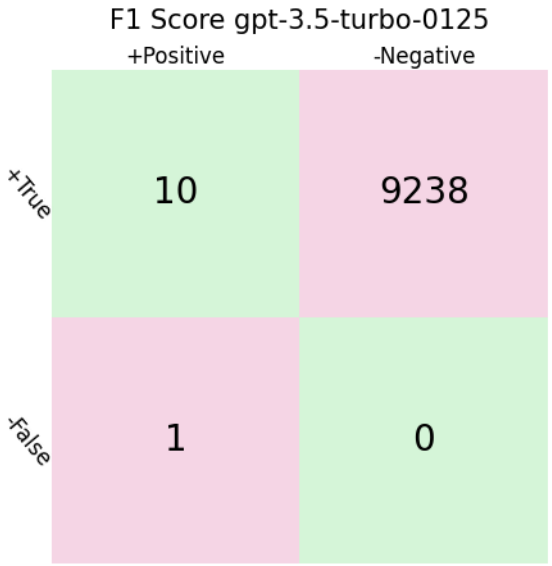

Figure 5-10: A 2x2 matrix showing how the model gpt-3.5-turbo predicted for the second document.

5.3.3 Third document

Table 6 presents the output from the LLMs for the third document. It shows that gpt-4-turbo managed to identify 2 out of 2 (100%) correctly with 0 incorrect references, see figure 5-11. This gives the gpt-4-turbo a precision, recall and F1 score of 1. And 2 out of the 2 (100%) identified references are an exact match with the expected output. While the gpt-3.5-turbo managed to identify 2 out of 2 correctly with 1 incorrect prediction, see figure 5-12. This gives the gpt-3.5-turbo a precision of ≈0.67, a recall of 1 and a F1 score of 0.80. And 0 out of the 2 (0%) identified were an exact match with the expected output.

Table 6: A comparison between the correct and model's outputs on the third document.

Expected output	gpt-4-turbo-2024-04-09	gpt-4-turbo-2024-04-09 quality	gpt-3.5-turbo-0125	gpt-3.5-turbo-0125 quality
SS-EN 13306:2017	SS-EN 13306:2017	Exact match	SS-EN 13306	Partial match
SS-EN 15341:2019	SS-EN 15341:2019	Exact match	SS-EN 15341	Partial match
-	-	-	SSG 2001	Incorrect

Figure 5-11 and figure 5-12 serve as summarized versions of table 6. Figure 5-11 showing the result for the gpt-4-turbo-2024-04-09 model, gaining 2 true positive, 0 false positive, 0 false negative and 6752 true negative.

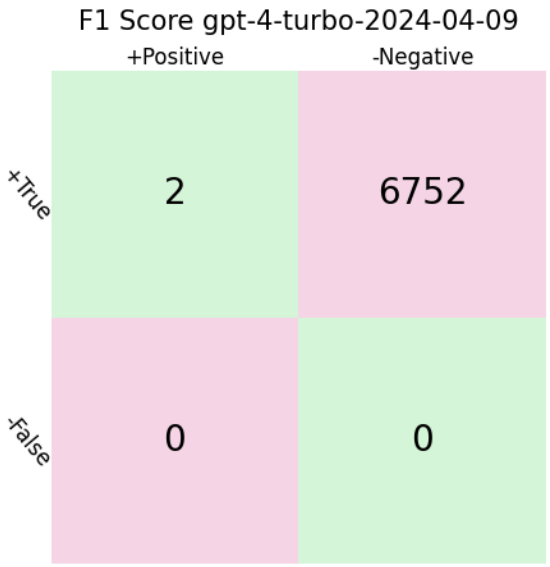

Figure 5-11: A 2x2 matrix showing how the model gpt-4-turbo predicted for the third document.

And figure 5-12 summarize the result for the gpt-3.5-turbo-0125 model, achieving 2 true positive, 1 false positive, 0 false negative and 6752 true negative.

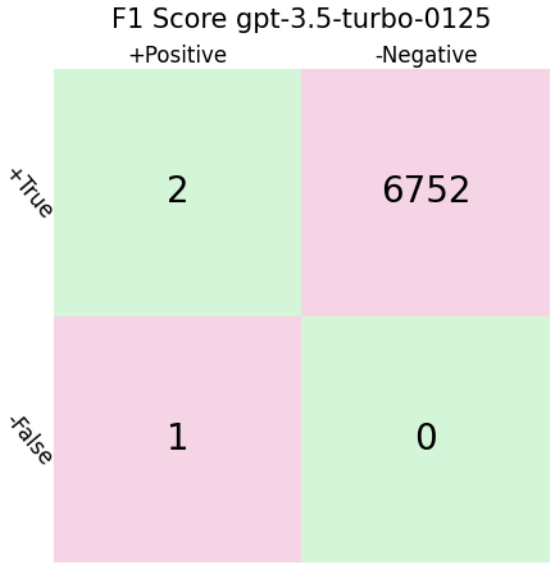

Figure 5-12: A 2x2 matrix showing how the model gpt-3.5-turbo predicted for the third document.

6 Discussion

This chapter includes discussions and analyses starting with an analysis and discussion of the result, then a discussion about the work methods, after that a scientific discussion, then a consequence analysis and last an ethical and societal discussion.

6.1 Analysis and discussion of results

The result of the embeddings for the first document shows that most of the vectors have been placed in one area while there are a few that stand out compared to the others. It also shows that the references for the most part are not mixed with the unwanted text. In the second document the result was a little different. It shows that the references have been placed in two different areas, one upper area where most of the references are clustered together and one area further down where some are clustered together and some mixed with the unwanted text. And the embeddings on the third document are all mixed with the unwanted text, even though most of the references are close to each other.

This is a consequence caused by the chosen method to divide the text by sentences. The chosen embeddings model uses context when transforming the text into embeddings. This allows sentences where the external reference has been used in the same way to match better with sentences containing big differences in phrasing. But on the other hand, can the contextual essence also make the embeddings focus on the wrong thing in the sentence, and working with larger sentences just amplifies this behavior. Which is the case in the third document, that document contains a lot of tables meaning that much unwanted text gets mixed with the text containing references. This explains why the sentences containing references are more mixed with the unwanted text.

In the analysis of the first document using cosine similarity, ≈6% of the sentences containing references (4 out of 70) and about ≈63% of the sentences without references (243 out of 388) were filtered out. Initially, the document contained 458 sentences, but post-filtering, only 211 sentences remained, 66 containing references and 145 without.

For the second document, the cosine similarity removed about 23% of the sentences with references (7 out of 29) and 69% without any references (195 out of 284), and unfortunately 3 references were removed from the

text in this process. Starting with 313 sentences, the process resulted in a final count of 111 sentences, with 22 containing references and 89 not containing.

Regarding the third document, the cosine similarity process eliminated roughly 23% of the sentences with references (7 out of 30) and 89% of those without references (302 out of 339). After the similarity search, only 23 sentences were left, of which 23 contained references and 37 did not.

And regarding the input and threshold for the cosine similarity, the chosen input "Enligt standard SS EN ISO 555" and threshold 45%. Worked well to remove unwanted sentences, having high percentages in removal for the unwanted texts. But also having a bit too high percentage when removing sentences containing references in the second and third document which can end up losing data. This could also be a consequence of the sentences being too large, since there are shorter and longer sentences is it hard to find a general input to include both the short and long ones. And also, that the long sentences can vary a lot, and the shorter ones are more limited.

When analyzing the removed sentences for the second document, it is shown that the reason behind three references not getting identified are because their sentences were not matched with the cosine similarity, and are only presented in one sentence each.

And lastly the results from the LLMs. Starting with the model gpt-4-turbo-2024-04-09, this model extracted a total of 42 out of 43 out of all the documents where 41 of the 42 were good quality meaning one missed some information. While the gpt-3.5-turbo-0125 model managed to extract 41 out of the 43 where 31 was good quality. These results show that gpt-4-turbo performed best, both at identifying external references and at including more details and also gaining higher F1 scores on each document.

6.2 Work method discussion

The initial phase, the pre-study, was crucial in laying the groundwork for the entire project. During this phase, I conducted extensive literature reviews and explored various methods for text extraction, embedding generation, and similarity searches. This helped me identify the most suitable tools and techniques for implementing the project.

The second phase, extracting text was done and implemented successfully. The third phase, used the extracted text to be divided into sections based on sentences. This was also implemented successfully although the sizes of the sentences can vary a lot. For instance, if the sentences instead would have been divided as a fixed size, meaning that every section contains the same number of characters. This method would have allowed each section to be more focused on the external reference but could remove some contextual capabilities since important words around the reference could get cut from that section. But using this method does also mean that the number of sections could become a lot so preparing structured tests would require more time.

The next phase, generating embeddings was also implemented successfully. The chosen model "text-embedding-3-small" by OpenAI showed good performance by for the most part separating the sentences containing references from the sentences without. The performance on this phase is also dependent on what method that was chosen in the previous phase. Since the chosen embeddings model takes context into account is it more reasonable to use sentences as each section to include the relations between the words. If the text instead was divided by a fixed smaller size a context-applying model could generate more inaccurate embeddings since important words in sentences could have been removed from that section.

Next phase, the cosine similarity, was also implemented and successful. By removing at least 85% unwanted sentences on each document with the input "Enligt standard SS EN ISO 5555" and a 45% threshold. Better performance was of course possible, but since the length and content of sentences can vary does it make it difficult to find a general input that can match with all possible sentences.

An alternative to this phase could be a machine learning model for text classification. But this approach would require more time since more data to train the model would be needed.

The last phase was also successful. Both models showed great performance while gpt-4-turbo performed better with more detailed output. It successfully extracted 42 out of 43 references since 3 got lost in the previous phase, with 41 being optimal and the remaining missing

some information. While the gpt-3.5-turbo-0125 model extracted 41 out of 43 references, with 31 matching the expected output perfectly.

A significant reason for the results is the detailed prompt provided to the LLMs. When constructing a prompt, it is crucial to make it as clear and simple as possible while also providing all the necessary information for what needs to be extracted. Typically, a prompt starts small to see the basic information that can be extracted. And then gradually add more details. However, it is important not to add too many details at once, as the model can be sensitive to overly complex prompts or just incorrectly phrased sentences. Which is the case in my prompt, even though mentioning that no 'SSG' documents should be included. Were they included a total of 6 times. Meaning, that sentence should have been reformatted to make it more clear for the LLMs. Therefore, continuous testing is essential to ensure optimal performance.

6.3 Scientific discussion

Answering and Discussing Research Questions:

1. How does the accuracy differentiate between the models when comparing the amount of identified external references?

The first model, gpt-4-turbo-2024-04-09 managed to extract a total of 42 out of 43 getting an accuracy of roughly 97.7%. While the gpt-3.5-turbo-0125 model successfully extracted 41 out of 43 references giving it an accuracy of about 95.3%. Meaning the gpt-4-turbo-2024-04-09 model gained a slightly higher accuracy with about 2 percentage points.

2. How does the quality differentiate between the models when comparing the identified external references?

From the gpt-4-turbo-2024-04-09 model 41 out of the extracted 42 was optimal outputs resulting in 97.6% containing the expected information. While 31 out of 41 were optimal outputs from the gpt-3.5-turbo-0125 model resulting in roughly 75.6%. This indicates that gpt-4-turbo-2024-04-09 provided a more detailed output with an improvement of about 22 percentage points over gpt-3.5-turbo-0125.

The scientific knowledge gained from this thesis are the following.

1. The preprocess needs to be implemented properly to not lose any data.

2. GPT 4 turbo showed the best performance, both in quantity and quality when given the instruction to extract data.

The first point is more general, since the preprocess phase is crucial for how accurate the text gets filtered before sent to a LLM, in my case this was the divide text, generate embeddings and cosine similarity phases. While the second point is more project specific, for it to be more general more and diverse tests would have to be made.

6.4 Consequence analysis

The consequence of this work is that, although not optimal configurations, show that it is possible to extract detailed data from unstructured sources. Since this work was done in cooperation with the company Standard Solution Group (SSG) I recommend them to continue pursuing research on LLMs and how it could be used in other cases.

6.5 Ethical and societal discussion

A significant ethical issue in the development of LLMs is the way training data is collected. There have been instances where large corporations have been accused of data theft during this process. For example, the news organization New York Times has sued both OpenAI and Microsoft over the unauthorized use of their content for training LLMs, and there are several more.

Another critical area is the possibility of harmful content generation. Since LLMs are powerful and capable of generating coherent and contextually appropriate text, they can also produce biased, offensive, or harmful content. This raises questions about the responsibility of developers to implement robust filtering and monitoring mechanisms to prevent these behaviors. Additionally, the training data itself may contain inherent biases, which can be reflected in the model's outputs.

The integration of LLMs into various sectors have the potential to automate complex tasks, improving efficiency, and providing advanced analytical capabilities. For example, in document processing, LLMs can

significantly speed up the extraction and analysis of information, leading to faster decision-making and productivity gains.

7 Conclusions

This chapter presents an answer to the problem statement, objective of the thesis and research questions to conclude this whole work.

So how can LLMs be utilized to accurately convert unstructured datasets into structured datasets? To handle larger documents is preprocessing necessary due to the context size limitation associated with Large Language Models (LLMs), meaning they can only process a fixed number of tokens at a time. The process begins with extracting text from the document, which is then divided into sections, with sentences being the chosen method for division. Embeddings are generated for each sentence, converting them into lists of numerical values between -1 and 1 that capture words and context. Next, cosine similarity is performed with an input text and a threshold to filter out sentences that do not match the input. The matched text is then sent to a Large Language Model (LLM), which has been given detailed instructions to extract external references.

The objective of this thesis was to successfully identify 70% of the references from one of the three test documents and compare the performance of two Large Language Models (LLMs). The results for the three test documents demonstrate that the gpt-4-turbo-2024-04-09 model generally outperformed the gpt-3.5-turbo-0125 model in terms of accuracy and detail. For the first document, gpt-4-turbo identified 30 out of 31 references correctly (≈97% accuracy) with an F1 score of ≈0.92. In comparison, gpt-3.5-turbo identified 29 out of 31 references correctly (≈87% accuracy) with an F1 score of 0.95. For the second document, gpt-4-turbo identified 10 out of 10 possible references correctly (100% accuracy) with an F1 score of 1 because the cosine similarity removed three references from the text. gpt-3.5-turbo also identified 10 out of 10 references correctly (100% accuracy) with an F1 score of ≈0.95. For the third document, gpt-4-turbo identified all 2 references correctly (100% accuracy) with an F1 score of 1. gpt-3.5-turbo identified 2 out of 2 references correctly but with 1 incorrect prediction, resulting in an F1 score of 0.80. Meaning that the objective of this thesis was achieved.

The first research question was about “How do the accuracy differentiate between the models when comparing the amount of identified external references”, this study shows the following. The first model, gpt-4-turbo-

2024-04-09, managed to extract a total of 42 out of 43 references, achieving an accuracy of roughly 97.7%. In comparison, the gpt-3.5-turbo-0125 model successfully extracted 41 out of 43 references, resulting in an accuracy of approximately 95.3%. This indicates that the gpt-4-turbo-2024-04-09 model achieved a slightly higher accuracy, with an improvement of about 2 percentage points.

And regarding the second research question "How do the quality differentiate between the models when comparing the identified external references". Was 41 out of the extracted 42 outputs were optimal from the gpt-4-turbo-2024-04-09 model, resulting in 97.6% containing the expected information. In comparison, 31 out of 41 outputs from the gpt-3.5-turbo-0125 model were optimal, resulting in approximately 75.6%. This indicates that gpt-4-turbo-2024-04-09 provided a more detailed output, with an improvement of about 20 percentage points over gpt-3.5-turbo-0125.

7.1 Optimize this project

I believe a better result for sure is possible by optimizing the method for dividing the text to something fixed and relatively small, the input text at the cosine similarity step and the instruction given to the LLMs. And the result should give a more accurate view of the maximum possible performance of LLMs.

7.2 Explore open-source models

Explore possible open-source embeddings models and LLMs that can replace OpenAI's models and be run locally to be more in control of how data is being handled and stored.

7.3 Natural language preprocess

Another possible method instead of LLMs could be natural language preprocessing (NLP) with Named Entity Recognition (NER). This is an alternative for extracting external references but would require a own trained model since this is a specific problem. NER makes it easy to extract data by just giving the model an input text and the output will be a list text with its associated tag for example external reference. This is not as complex but also not as flexible when compared to the possibilities of an LLM.

References

[1] IBM, “What are large language models (LLMs)?” [Online], Available: https://www.ibm.com/topics/large-language-models [Accessed May 2, 2024].

[2] Journal of Machine Learning Research, “Visualizing Data using t-SNE,” [Online], Available: https://jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf [Accessed May 2, 2024].

[3] L. Tunstall, L. von Werra, and T. Wolf, "From Text to Tokens" in Natural Language Processing with Transformers. Sebastopol, CA, USA: O'Reilly Media, Inc., 2022, pp. 30-34.

[4] Stanford University, “Speech and Language Processing (3rd ed. draft)”, [Online], Available: https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf [Accessed May 3, 2024].

[5] Wayne Xin Zhao, Kun Zhou*, Junyi Li*, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie and Ji-Rong Wen, “A Survey of Large Language Models”, [Online], Available: https://arxiv.org/pdf/2303.18223 [Accessed May 3, 2024].

[6] Humza Naveeda, Asad Ullah Khana, Shi Qiub, Muhammad Saqibc, Saeed Anware, Muhammad Usmane, Naveed Akhtarg, Nick Barnesh, Ajmal Miani, “A Comprehensive Overview of Large Language Models”, [Online], Available: https://arxiv.org/pdf/2307.06435 [Accessed May 4, 2024].

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser and Illia Polosukhin,

"Attention Is All You Need", [Online], Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
[Accessed May 4, 2024].

[8] Daniel Jurafsky and James H. Martin, "Speech and Language Processing (3rd ed. draft)", [Online], Available: https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf
[Accessed May 4, 2024].

[9] David M.W. Powers, "What the F-measure doesn't measure…", [Online], Available: https://arxiv.org/pdf/1503.06410
[Accessed May 7, 2024].

[10] GeeksforGeeks, "F1 Score in Machine Learning", [Online], Available: https://www.geeksforgeeks.org/f1-score-in-machine-learning/
[Accessed May 7, 2024].

[11] Huaxia Li, Haoyun Gao, Chengzhang Wu and Miklos A. Vasarheli, "Extracting Financial Data from Unstructured Sources: Leveraging Large Language Models", [Online], Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4567607
[Accessed May 5, 2024].

[12] Dipali Baviskar, Swati Ahirrao, Vidyasagar Potdar and Ketan Kotecha, "Efficient Automated Processing of the Unstructured Documents Using Artificial Intelligence: A Systematic Literature Review and Future Directions", [Online], Available: https://ieeexplore.ieee.org/abstract/document/9402739
[Accessed May 6, 2024].

[13] PyPI, "PyPDF2", [Online], Available: https://pypi.org/project/PyPDF2/
[Accessed May 6, 2024].

[14] NLTK, "nltk.tokenize", [Online], Available: https://www.nltk.org/api/nltk.tokenize.html

[Accessed May 6, 2024].

[15] OpenAI, "Embeddings", [Online], Available: https://platform.openai.com/docs/guides/embeddings
[Accessed May 5, 2024].

[16] OpenAI, "Python Library", [Online], Available: https://platform.openai.com/docs/libraries/python-library
[Accessed May 6, 2024].

[17] scikit-learn, "sklearn.metrics.pairwise.cosine_similarity", [Online], Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html
[Accessed May 6, 2024].

[18] OpenAI, "API Reference - Chat Completion", [Online], Available: https://platform.openai.com/docs/api-reference/chat/create#chat-create-temperature
[Accessed May 21, 2024].

[19] OpenAI, "GPT-4 Turbo and GPT-4", [Online], Available: https://platform.openai.com/docs/models/gpt-4-turbo-and-gpt-4
[Accessed May 16, 2024].

[20] OpenAI, "Tokenizer", [Online], Available: https://platform.openai.com/tokenizer
[Accessed May 13, 2024].

Appendix A: Source Code

https://github.com/Lurre13/Examensarbete-SSG