Politecnico di Torino

Politecnico di Torino

Data Science and Engineering
A.a. 2023/2024
Sessione di Laurea Luglio 2024

Master's Degree Thesis

The Use of LLMs in the Legal Field: Optimizing Contract Management with Generative Artificial Intelligence

Relatori:
Maurizio Morisio
Corelatore:
Daniele Sabetta

Candidati:
Alessio Mongoli

ABSTRACT

In recent years, Artificial Intelligence (AI), including the emergence of ChatGPT, has attracted significant attention due to its increasing prevalence in several aspects of business processes. AI involves the development of automated systems capable of executing tasks traditionally performed by humans, with the aim of speeding up processes and reducing wasted time within organisations. This technology has also opened significant opportunities for application in the legal sector, traditionally engaged in analysing large amounts of documentation. This Master's thesis explores the use of Large Language Models (LLM) to support legal staff and reduce document management time. The aim of this research is to study, design, develop a POC (proof of concept) to address these challenges by implementing a web application where lawyers can analyse contracts e generate contract. The application is based on Retrieval-Augmented Generation (RAG) capable of providing fast, effective and high-quality responses. To achieve this goal, an in-depth analysis was conducted on large language models and the prompts used to guide them. To achieve this, the analyses focused on the effectiveness of LLMs in interpreting legal language and their ability to integrate information to produce relevant and coherent output. Particular attention was paid to the configuration of prompts and their optimisation to improve the accuracy of responses. In conclusion, this thesis highlights the considerable potential of generative AI in the legal field. By integrating the advantages of semantic embeddings for information retrieval with those of generative AI for producing answers, lawyers can significantly reduce the time spent in drafting new contracts, taking into account previous clauses, and analysing new contracts. This approach enables effective optimisation of legal processes, making contract management more efficient and accurate.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Over the past several years, Artificial Intelligence (AI), including the emergence of ChatGPT, has attracted significant attention due to its increasing prevalence in various aspects of enterprise operations. AI involves developing automated systems capable of performing tasks traditionally carried out by humans, aiming to streamline processes and reduce time wastage within enterprises. This integration of AI into systems is gaining momentum as companies recognize its potential to enhance efficiency and productivity. In the legal field, enterprises are often inundated with a vast number of contract documents containing various types of clauses. Many times, during contract drafting, lawyers spend considerable time analysing numerous contracts to choose the correct clause.

Centralizing and standardizing document management across various sources and formats would empower any lawyers to access necessary information by querying an intelligent system.

This master's thesis, conducted in partnership with Orbyta, explores the integration of generative AI in the legal field by proposing a pipeline for contract analysis and clause generation. The key component of this approach is an information retriever that employs semantic embedding to grasp the connection and implicit meaning among words. By adopting this method, the system can identify and retrieve a set of documents that are most pertinent to the user’s query efficiently. This functionality enhances the user experience by offering precise and direct solutions, reducing cognitive overload, and eliminating the need to access multiple documents for information.

This thesis aims to develop a POC (prof of concept) to address these

challenges by implementing a web application where lawyers can analyse contracts e generate contract. This web application is intended to offer a user-friendly experience, enabling users to make requests without the need for complex query languages.

2. ORBYTA

2.1 PROVIDER PROFILE

Orbyta Tech is the technology company of the Orbyta Group, which comprises seven companies. Thanks to the know-how and specialised skills of the individual companies, Orbyta offers its customers comprehensive support and consulting, covering all areas of business interest. The group's offering is structured in two macro-brands, each of which addresses specific customer targets and sectors:

FIGURE 2.1: THE ORBYTA GROUP LOGO

- **Orbyta Technologies**: A leading IT consultancy company, Orbyta Technologies specialises in both applications and systems. It develops highly complex projects using the most advanced technologies and state-of-the-art methodologies. Its expertise includes the design, implementation, integration and maintenance of software, hardware and IoT systems. Its divisions include:
    - **Orbyta Tech**: Focused on software development and systems support, it offers integrated solutions and designs IT

infrastructures, providing consultancy services and turnkey projects.

  - **Orbyta Infogest:** Focuses on designing, supplying and reselling hardware, as well as installing and servicing PCs, servers, storage and internetworking solutions for diversified operating environments.

- **Orbyta Business Partner:** This division provides essential business support services, assisting customers in the areas of compliance, engineering design, and accounting, administrative, tax and financial management. It also offers payroll processing and outsourced human resources management services, as well as out-of-court and in-court legal assistance.

The well-established synergy between the group's companies makes Orbyta a reliable and comprehensive partner, capable of offering a wide range of services and solutions. This integrated approach allows Orbyta to successfully meet the challenges of the market, guaranteeing efficient and tailor-made solutions for every customer need. The companies that are part of the business partner area are:

- **Orbyta Engineering**: Specialising in civil and industrial engineering, this company offers design and construction management services, identifying customised solutions that comply with legal requirements and aim to simplify corporate compliance.

- **Orbyta Tax&Finance**: This division focuses on tax and corporate consulting, providing support for management, accounting and all civil and tax obligations necessary for business activities.

- **Orbyta People**: Provides consultancy in the employment area, managing administration and human resources, payroll processing,

time management, benefits, welfare and labour relations.

- **Orbyta Legal**: This division provides legal assistance and advice, both judicial and extrajudicial, with a particular focus on business management and development, offering ongoing support.

- **Orbyta Strategy** This in-house company contributes to the continuous improvement of business processes. It provides integrated services for corporate organisation and growth, setting strategies and managing the Group's internal dynamics.

2.1.1 KEY PROVIDER INFORMATION

Orbyta is a constantly growing group, with a 2022 turnover of 15 million euros. The group has about 250 employees and is in various locations: Turin, Milan, Rome, Lecce. These offices and the presence of consultants located in other areas allow the company to cover geographically the whole Italian territory. The process of analyzing and evaluating investments and acquisitions in foreign offices, particularly in Germany, it’s currently happening.

2.2 PROVIDER ORGANIZATION OVERVIEW

At Orbyta Technologies, innovation is at the heart of every activity. The company guides its partners through the conception, design and development of technological processes that are not only interactive but also fully immersive. With highly specialised multidisciplinary expertise, Orbyta stands out in the field of design, development and implementation of complex information systems and state-of-the-art digital solutions. A dedicated team is always ready to provide support to partners and companies alike, turning accumulated experience into continuous innovation. Orbyta's offering is organised in the following areas:

- **Digital Transformation**: Orbyta Technologies guides and supports partners in their digital transformation journey, offering technological solutions and IT architectures in line with growth objectives. The cross-skilled consulting team manages every aspect of the process, coordinating all activities and monitoring performance. The ability to go beyond traditional schemes and a holistic view of business processes allow innovation and efficiency to be optimised.

- **Software Development**: develop tailor made technological solutions implementing a wide range of IT products and projects in multiple areas of intervention, with carefully composed teams with specialist skills ranging from project management to the most up-to-date ICT training. The aim is to become a reference point for the IT architecture of each partner thanks to the planning and management capacity of information systems and subsystems of

the team in our software house, Area 51.

- **Design & Strategy**: XLAB, Orbyta Tech's creative team, is dedicated to promoting digital growth and developing high impact omnichannel strategies. It focuses on creating connections using an effective mix of user experience, digital interface design, creative communication and digital marketing. Working at the intersection of business, technology and design in all its forms (Design Thinking, Human-Centred Design, System Design, Service Design, Futures Design, User Experience, User Interface) XLAB aims to transform innovative visions into tangible realities.

The team consists of pixel-perfect and enthusiastic futurists, who collaborate with the partners at all stages of the project, from analysis and design to prototyping and testing. The customer is a member of the team, an irreplaceable project partner in the co-creation of the best digital product. The company's competencies are User research, UI/UX Design, Brand Design, Brand Strategy, 3D Design, Creative & Integrated Communication, Web Experience Development and Metaverse Creation. The company's approach is:

- **Collaboration**: smart working, hybrid and presence modes to ensure efficiency in every situation. It uses the best collaboration tools, such as Trello, FigJam, InVision, Zeplin and many others, to keep teams synchronised and projects aligned with customer expectations.

- **Design**: employs the best design and development tools available on a daily basis, including Figma, Sketch, Adobe, Blender and Webflow. This allows it to remain at the forefront of design and to

respond effectively to its customers' needs.

- **Innovation**: constantly dedicated to innovation and experimentation, especially in the creation of 3D environments in the Metaverse. This approach is geared towards devising new business models that take advantage of the latest technologies and market trends

- **Infrastructure Networking**: offerinh advanced networking and security enhancement services for the corporate network. It identifies connectivity needs and makes projects operational, working both remotely and on-site. Its consolidated experience, combined with a constant search for the most innovative technologies, guarantees systemic support and management consultancy in large data processing centres, including the banking, insurance and industrial sectors.

- **Hardware Reselling:** specialising in the creation of customised hardware infrastructures, supported by a continuous, high added-value consultancy process that facilitates organisational and management change and development of business flows. The process begins with understanding the customer's needs, followed by building the necessary framework and guiding integration with day-to-day business operations. Each solution is built to be secure and reliable, with careful evaluation of the best available technologies. The Infogest team utilises established partnerships with leading market players such as HP, Microsoft, Fortinet, VMware, Veeam and Arcserve, thus ensuring access to state-of-the-art solutions in the technology sector.

The current organization of Orbyta Tech in 4 Units and Dedicated Teams for customers and projects with similar technology stacks, makes possible

the parallel and coordinated development of initiatives:

- **Intelligent Platform**: Design of complex and resilient Cloud Native architectures, Data Analytics, ML and AI.
- **Process Automation**: Design and development of software modules on the Microsoft DotNet stack, Java, Node, Javascript and Python.
- **Digital & App Innovation**: Design and development of web client, desktop and mobile applications, with different targets and development stacks, such as Angular, React, Vue, Flutter, ReactNative, Swift, Kotlin.
- **Business Consulting**: Governance and management of complex projects, with the application of the best methodologies and development of automated test phases.

2.3 BUSINESS PROVIDER STRATEGY AND PROFILE

Orbyta Tech operates in the area as a System Integrator and offers consultancy to large corporate client companies from various fields, including:

- **Banking & Insurance**: Design products for every branch of business, from digital payment services to fraud control, web security and encryption services, from a template predictive decision-making on financing to an operations asset management software, up to the creation of an application for managing the migration of a complex set of data.
- **Automotive & Industrial**: Work in synergy with partners,

international companies of recognized fame, for the development of: high-speed data streaming and display mechanisms towards remote customers; a complete modeling of the life cycle of software with complex functions of predictive maintenance, intrusion detection, mitigation and firmware over the air update; platforms for the management of complete technical documentation of products with data profiling and automation capabilities for use by teams; application for the cross-management of stock availability and supplies purchase in relation to production times.

- **Transportation**: Carry out innovative technological projects that contribute to the relevant need of the transport and logistics sector to carry on a process of digitization of systems to promote increasingly integrated mobility; to return punctual and in real time information, to maintain the attractiveness for users of the services.

- **Manufacturing**: Structure solutions capable of integrating, harmonizing and aggregating data from multiple sources with the aim of extracting value and optimizing workflows. It's about projects of high strategic value that facilitate monitoring, verification and control and provide important forward-looking data.

- **Textile & Fashion**: Design digital solutions of great strategic impact that intervene in all phases of the production processes. Technology becomes an essential resource for being competitive in a sector strongly permeated by craftsmanship and element crucial to consolidate the presence on the market and satisfy, if not even anticipate customer needs.

- **Gaming**: Conceive and develop proposals that are characterized as

augmented and virtual experiences, totally immersive, also through the creation and use of avatars. Design solutions that through gaming elements are oriented to improve the company performance through user engagement strategies aimed at multiple goals.

Orbyta Tech mainly deals with: Technical consulting, Business analysis, Research and development, Software development and operations, Process management and support, Digital transformation, Data analysis, Cloud, lean processes & new digital core, IOT and connected services. Orbyta Tech is historically Gold Partner of Microsoft, cultivates further expertise in the public Cloud area also with Amazon Web Services (AWS) and Google cloud. As part of the management of multi-cloud native cloud platforms, the simplification of IT operations and the improving of software product efficiency, Orbyta Tech is a partner of the Mia Platform company.

3. BACKGROUND

Natural language is the primary medium through which humans communicate and express their thoughts. The term Natural Language Processing (NLP) describes the field that analyzes natural language.

This chapter provides a brief introduction to NLP in section 3.1, and then discusses the significant contributions of deep learning models in section 3.5, made possible by the exponential growth in computational power.

3.1 INTRODUCTION TO NATURAL LANGUAGE PROCESSING (NLP)

This chapter serves to introduce key concepts in Natural Language Processing (NLP) essential for comprehending the methodologies employed in this thesis.

The overarching objective of NLP has always been to enable machines to comprehend human-written text. Given the complexity of language, machine understanding is organized into distinct levels, resembling a pipeline:

- **Lexicon**: This refers to the dictionary of words utilized.
- **Morphology**: It involves analyzing how words are formed by combining morphemes, which are the smallest units of meaning (e.g., root and affixes). Morphological analysis aims to decompose words into their constituent morphemes, akin to stemming, which reduces a word to its base form or lemma.
- **Syntax**: This concerns the arrangement of words to form grammatically correct sentences, including understanding

grammatical relationships such as subject-object linkages and dependencies.

- **Semantics**: This involves interpreting the meaning of individual words and how they combine to convey the overall meaning of a sentence.

- **Pragmatics**: This examines how the meaning of a sentence is influenced by the context in which it is used, including social, cultural, and spatial-temporal factors. It considers the intended purpose or concept conveyed by the user within a given context.

In applications involving spoken language, an additional level of analysis—Phonetics and Phonology—is often required to understand the sounds that constitute a language.

Due to the exponential growth of textual data, Natural Language Processing (NLP) has gained significant importance in recent years. Particularly with the growth of machine learning and deep learning techniques, a wide array of problems can now be effectively tackled, including sentiment analysis, machine translation, text summarization, and more.

Understanding the context in which language is used is crucial, but it presents several challenges. Firstly, many words in various languages can have multiple meanings, necessitating the elimination of ambiguity. Word sense disambiguation, an active area of research in NLP, aims to address this by identifying the correct sense of ambiguous words within a specific document.

Secondly, the understanding task involves documents from diverse domains, each with unique characteristics that NLP models need to grasp.

Despite these challenges, the success of an NLP algorithm is determined by its ability to accurately accomplish its task, regardless of whether it possesses explicit knowledge of the underlying linguistic structure or concepts involved.

3.1.1 HISTORY

During the 1960s, research was primarily focused on creating rules to manually model human language. Data-driven approaches were deemed impractical due to the large data sizes required, high processing overhead, and the need for efficient learning algorithms. Consequently, the most viable method for language models involved manually defined rules that incorporated local linguistic dependencies for specific NLP tasks. Despite their usefulness, these approaches had notable limitations: the effectiveness of the rules depended on the knowledge of their creators and updating and adapting them to new languages required substantial effort.

In the late 1980s, performance improvements were achieved through the adoption of statistical methods, facilitated by advancements in computational power and access to extensive text corpora. Statistical techniques addressed the limitations of their predecessors by enabling the long-term modelling of language dependencies, automating the training process, and reducing reliance on human intervention. This transition from traditional to statistical techniques paved the way for the development of new machine learning algorithms, such as decision trees, which have demonstrated efficacy in various natural language processing tasks, including part-of-speech tagging.

In applying statistical techniques, language processing and generation activities utilized the concept of n-grams. An n-gram represents a

sequence of n consecutive elements in a text sample, expressed in words or characters. For example, a bi-gram extracted from the phrase "machine learning and deep learning" would include pairs like "machine learning," "learning and," and so on. This approach, extended to character-based bi-grams, captures sequences such as "ma," "ac," "ch," "hi," "in," "ne," and so forth. The objective is to establish a series of elements and leverage statistical techniques to ascertain the likelihood of their co-occurrence.

This succinct yet dynamic definition of n-grams has spurred the development of innovative text generation systems, information retrieval techniques, text mining methods, and other applications. By applying n-gram definitions, a fresh approach to document representation has emerged, focusing on individual words.

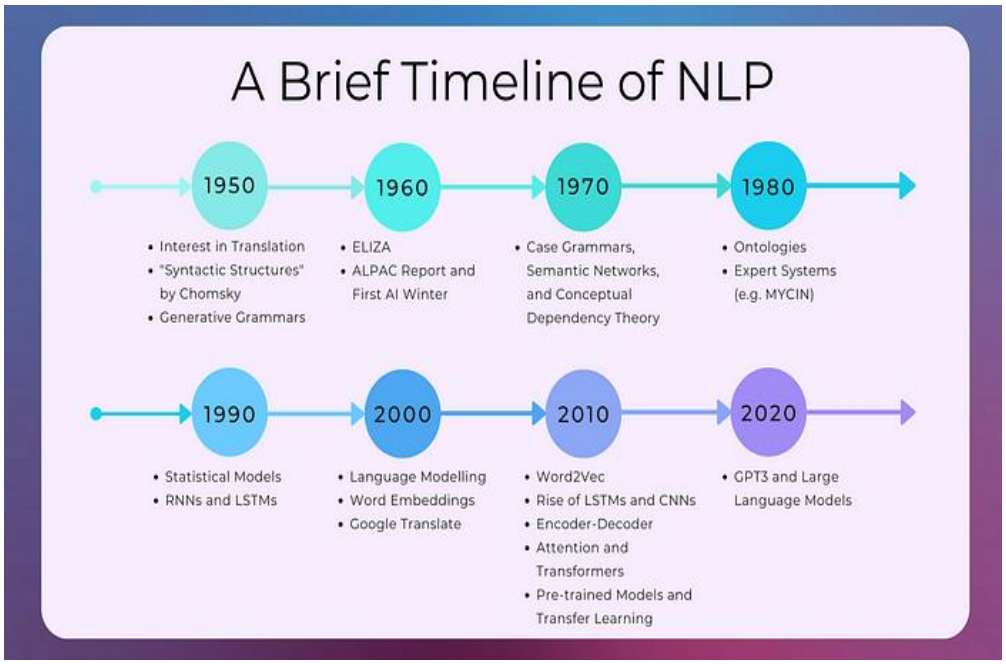


FIGURE 3.1: THE EVOLUTION OF NATURAL LANGUAGE PROCESSING

3.2 TEXT REPRESENTATION

In the field of NLP, how data is represented is critical. Text representation in NLP involves converting words, sentences, or documents into numerical or vectorized formats, enabling analysis and processing by Machine Learning (ML) algorithms.

The Bag-of-Words (BOW) approach is a widely employed method for text representation in NLP. Its concept is straightforward yet highly practical. BOW treats text as a "bag" of words, disregarding their order, and counts the occurrences of each word in the document or collection of documents. The process consists of the following steps:

1. **Tokenization**: The text is segmented into individual words or "tokens."

2. **Vocabulary creation**: A vocabulary is constructed, containing all unique words present in the documents.

3. **BOW vector**: For each document or sentence, a numeric vector is generated with a length equal to the vocabulary size. Each position in the vector corresponds to a word in the vocabulary, and the value at each position indicates the frequency of the word in the document.

The encoding used in the bag-of-words representation presents a fundamental issue, as it only indicates whether a word appears at least once in a document. While straightforward to use, this method fails to account for the frequency of a word within the document. The Vector Space Model (VSM) offers a solution to this problem. Originally designed for information retrieval, the VSM has been extensively applied in various NLP tasks. In the VSM, the frequency of each word is used to represent the document, rather than just the presence of words. This approach is further refined by incorporating TF-IDF (term frequency-inverse document frequency) [28], a commonly used statistic that highlights the significance

of a term in a document relative to a collection or corpus. The inverse document frequency (IDF) component of TF-IDF penalizes frequently occurring words, as these are less distinctive within the collection. Although more precise than the bag-of-words approach, TF-IDF still faces challenges related to data sparsity and the absence of contextual

3.3 EMBEDDING

3.3.1 WORD EMBEDDING

Unlike traditional text representation methods, word embeddings capture semantic relationships and contextual information. Essentially, word embeddings represent words as vectors in a continuous vector space, positioning words with similar meanings closer to each other.

Word2Vec is a pioneering approach proposed by Mikolov et al. in 2013 [1] to learn word embeddings. The core idea is to train a neural network where, given a target word in a dictionary, a sliding window moves over the text to collect training samples and make predictions. The objective is to predict the surrounding words of the target word, using the sliding window to define contextual positive examples (self-supervision). The size of this window determines how many words before and after the target word are considered as context words. Mikolov and his colleagues also addressed the complementary task of training a neural network to predict the target word given the context words. To achieve this, they proposed two architectures:

CBOW (Continuous Bag-of-Words) [1]: This model predicts the current word based on a window of surrounding context consisting of C words within a range of k words. The order of the context words does not influence the prediction (bag-of-words assumption), and any repetition of

words is ignored.

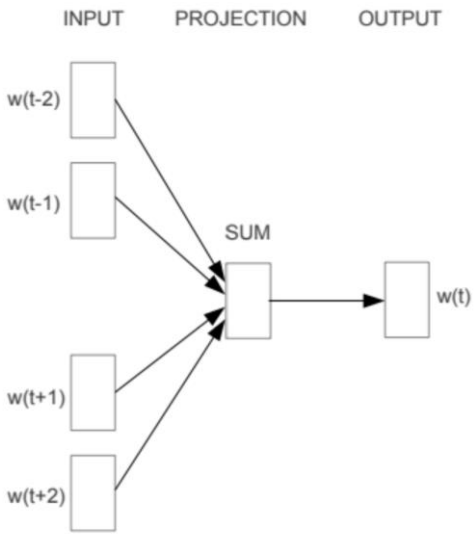


FIGURE 3.2: CONTINUOUS BAG-OF-WORDS(CBOW) ARCHITECTURE [1]

Skip-Gram [1]: This model uses the current word to predict the surrounding context within a window of C words. The skip-gram architecture assigns more weight to nearby context words than to those that are more distant.

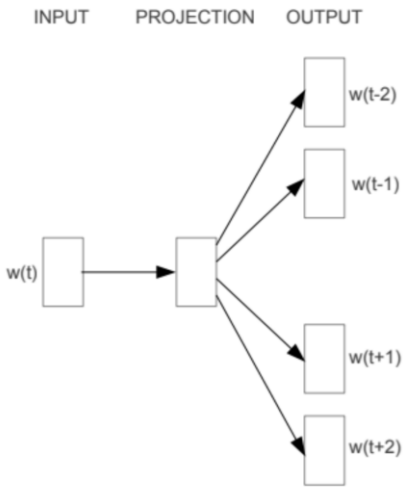


FIGURE 3.3:SKIP-GRAM ARCHITECTURE [1]

The model consists of a single hidden layer that produces the vector representation of words. These vectors are initialized with random values and gradually updated during training. It is important to note that the training procedure does not require annotations. Both training strategies are illustrated in Figure 3.1.

Given a sequence of training words $w1,w2,\ldots,wTw1,w2,\ldots,wT$ with a total length T, the objective function of the Continuous Skip-Gram model is to maximize the average log probability:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log\, p\left(w_{t+j} \mid w_t\right)$$

where c is the size of the training context.

In contrast, the Continuous Bag-of-Words (CBOW) model aims to predict the target word given the surrounding context words. In this model, the input layer consists of NN words, which are encoded and passed to a projection layer applied to all words. A hidden vector is then created,

element-wise averaged, and passed to the output layer. The output layer is responsible for generating the probability distribution across the vocabulary.

Given a sequence of training words $w1,w2,\ldots,wTw1,w2,\ldots,wT$ with a total length T, the objective function of the CBOW model is to maximize the average log probability:

$$\log P(w_c \mid w_{c-m}, \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+m})$$

where m is the size of the training context.These architectures have some drawbacks. Firstly, during training, only the weights corresponding to the target word might receive significant updates, while the weights related to non-target words might experience only minor changes or no changes at all. Secondly, calculating the final probabilities using the softmax function is highly inefficient, as the computational cost is proportional to the size of the vocabulary.

3.4 RNN

Recurrent Neural Networks (RNNs) are particularly effective in processing sequences of data due to their ability to maintain a form of 'memory' of previous inputs. This distinguishes them from traditional neural networks that process each input independently without reference to previous ones. The sequential nature of RNNs allows them to take into account the context provided by the previous elements of a sequence, a fundamental feature for applications such as natural language processing. For example, in text generation, an RNN can predict the next word based not only on the

immediately preceding word, but on the entire sequence of words generated so far, thus enabling linguistic productions that respect a more natural syntactic and semantic coherence.

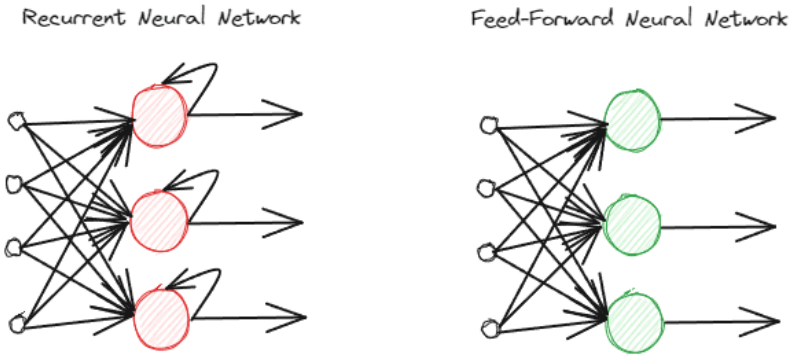


FIGURE 3.4:RECURRENT NEURAL NETWORK VS FEED-FORWARD NEURAL NETWORK [2]

As illustrated in Fig.3.4, Recurrent neural networks (RNNs) are distinct from feed-forward neural networks because, in RNNs, the output of one node can influence the next, forming a loop in the data flow. This feature allows RNNs to use previous information to influence future computations, making them suitable for processing data sequences such as text, where contextual understanding is essential.

Although effective in sequence processing, traditional RNNs find it difficult to maintain long-term dependencies due to the gradient vanishing problem, which makes it difficult for the network to learn long-term dependencies in sequences. In order to overcome these bottlenecks, variants of RNNs such as Long Short-Term Memory (LSTM) [3]and Gated Recurrent Units (GRU) [4] have been developed. These models introduce 'gate' structures that regulate the flow of information, allowing the network

to 'decide' which information to keep or discard. This mechanism helps to preserve gradients and improve the network's ability to learn dependencies between data that occur at long time intervals.

Although these networks can achieve excellent results in various domains, they require long training and are computationally expensive. Each iteration necessitates the computation of all previous steps, impeding the parallelisation of the training phase.

A significant challenge over the past decade has been to capture dependencies between words, trying to speed up or parallelise the training process.

3.5 TRANSFORMER

Transformers, introduced by Vaswani et al. in 2017 [5], have revolutionized the fields of NLP and Computer Vision. Prior to Transformers, state-of-the-art NLP solutions heavily relied on Recurrent Neural Networks (RNNs) such as LSTM and Gated Recurrent Units (GRUs). However, the sequential nature of RNNs made parallelization during training difficult. The Transformer architecture uses an encoder-decoder model based on self-attention. This allows for non-sequential processing and parallelization, significantly speeding up the training process. The input sequence is first transformed into three matrices representing keys (K), values (V), and queries (Q). To compute the output matrix, the authors proposed a modified Dot-Product Attention, called "Scaled Dot-Product Attention":

$$\text{attention}\ (Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where dk represents the dimension of the queries and keys.

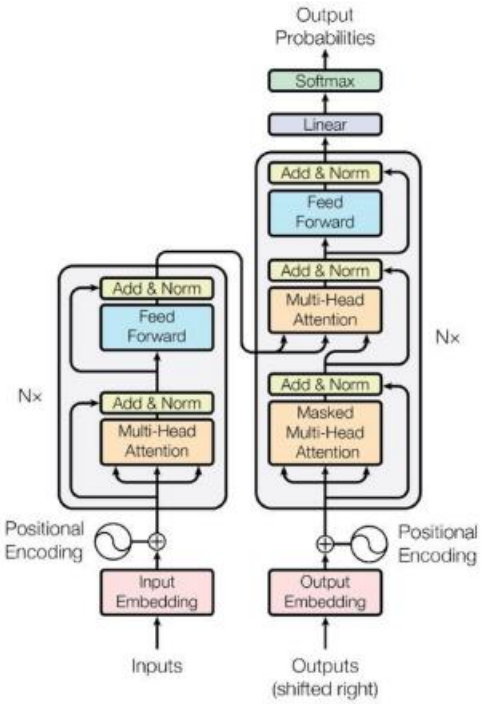


FIGURE 3.5:TRANSFORMER ARCHITECTURE [5]

The Transformer architecture consists of an encoder and a decoder, each containing multiple layers of self-attention mechanisms. A typical structure of the Transformer model is shown in Figure 3.5, the encoder block on the left and the decoder block on the right.

The encoder processes the input sequence iteratively, with each layer generating encodings that provide information about which parts of the inputs are relevant to each other. The encoder's output is then used as input to the decoder. The decoder's role is to use the context information received to generate an output sequence.

The **attention mechanism** enables the Transformer to capture dependencies between sequence elements by assigning different levels of importance to each element using learnable weights during training.

Consider a sentence with n words S:$w1$,$w2$,…,wn *S*:*w*1,*w*2,…,*wn*, where each word wi is represented by an initial vector xi and a triplet of vectors qi, ki, and viv (representing "query", "key", and "value" respectively). These vectors are obtained by multiplying the initial weight matrices Wq, Wk, and Wv by the vector xi. To calculate the attention of each word wi with respect to itself (hence the term self-attention), the dot product between qi and ki is computed, followed by a softmax operation. The resulting scores are then multiplied by the value vector vi.

To enhance the self-attention layer, the multi-headed attention mechanism is used. This involves using n different sets of weight matrices Wq, Wk, and Wv, all initialized randomly. Each set captures distinct relationships, resulting in varied representations that are combined through another layer of trainable weights $W0$.

This architecture is highly parallelizable, making training more efficient. However, the computational complexity is O (n^2) for a sequence of length n, which generally limits the length of input sentences to 512 or 1024 tokens. Both the encoder and decoder layers include a feed-forward neural network for additional processing of the outputs and contain residual connections and layer normalization steps. The residual connections help bypass layers that do not provide significant information, while layer normalization speeds up the training process and reduces the risk of overfitting.

Depending on the task, it is possible to use only one part of the encoder-decoder architecture. For instance, GPT-3, a state-of-the-art language model for human-like text generation, uses only the decoder side. In contrast, BERT [6], a state-of-the-art model for sentence encoding, and its variants use only the encoder side.

3.5.1 GENERATIVE PRE-TRAINED TRASFORMER

The Generative Pre-Trained Transformer (GPT) is a language model based on the decoder block of the Transformer architecture. The architecture is modified by removing the encoder-decoder attention layer. The model inherits the autoregressive property from the decoder, meaning it processes the sequence from left to right. The pre-training task focuses on next-word prediction, enabling the model to understand the sequential structure of text. It is classified as an autoregressive model, where each generated word is fed back into the input to continue the sequence generation.

As the name suggests, GPT aims to generate text that is both coherent and contextually relevant. It is a useful tool for various tasks, including text generation and translation.

The first version, GPT-1 [7], was constructed with a stack of 12 Transformer decoder blocks, resulting in 117 million parameters. Pre-training was conducted on a dataset called BookCorpus, which comprises roughly 4.5GB of text from 7000 books. These features provide the model with a comprehensive and rich linguistic foundation capable of understanding intricate language structures. The latest version, GPT-3 [8], has 175 billion parameters (requiring 800 GB of storage), allowing the model to produce sophisticated text almost indistinguishable from human-generated text.

3.6 LLM

A large language model is a type of artificial intelligence (AI) model specifically designed to understand and generate human language. It is a particular kind of transformer that has been trained on vast amounts of text data.

A language model takes a sequence of words as input and predicts the probability distribution of the next word or sequence of words. It learns from large datasets, including books, articles, websites, and other sources, to capture the statistical patterns and relationships between words. By analyzing these patterns, a language model can generate human-like text based on the context and input it receives.

A large language model, such as GPT-4 [9](one of the largest language models to date), refers to a model with an extensive number of parameters. Parameters are internal variables the model uses to make predictions and store information. The more parameters a model has, the more complex and nuanced its understanding of language can be.

Large language models like GPT-4 are trained on massive datasets containing billions of sentences to develop a deep understanding of grammar, syntax, and semantics. This training allows them to generate text that is remarkably coherent and contextually relevant. These models can be fine-tuned for specific tasks, such as translation, summarization, question answering, and more.

3.6.1 LLM – Common Use cases

Large Language Models (LLMs) have a wide range of use cases due to their ability to understand and generate human-like text based on the

extensive training data they've been exposed to [10]. Here's a detailed presentation of some key LLM use cases:

1. **Text Generation** (Generative Use Cases)

- **Creative Writing**: LLMs can generate creative pieces of writing, including poetry, short stories, and even novels. They can assist writers by providing inspiration and generating content.
- **Content Creation**: LLMs can automatically generate articles, blog posts, and other forms of written content, which is valuable for content marketers and publishers.
- **Code Generation**: LLMs can produce code snippets in various programming languages based on high-level descriptions or requirements, aiding software developers.
- **Data Augmentation**: LLMs can generate additional training data for machine learning models, helping improve the performance of various AI applications.

2. Natural Language Understanding (NLU) Use Cases

- **Chatbots**: LLMs power conversational AI by understanding user queries and generating human-like responses. They're used in customer support, virtual assistants, and more.
- **Sentiment Analysis**: LLMs can determine the sentiment (positive, negative, neutral) of text, which is valuable for understanding customer opinions and market trends.
- **Named Entity Recognition**: LLMs can extract specific information, such as names, locations, dates, and organizations, from text, which is useful in various data analysis tasks.

- **Language Translation**: LLMs excel at translating text from one language to another, enabling real-time language translation in various applications.

3. Text Summarization and Information Retrieval

- **Text Summarization**: LLMs can generate concise summaries of long articles or documents, which is valuable for quick information retrieval and content curation.
- **Search Engines**: LLMs can improve search engine results by better understanding user queries and retrieving more relevant documents or web pages.

4. Text Classification and Sentiment Analysis

- **Topic Classification**: LLMs can classify documents or text snippets into predefined categories, aiding in content organization and information retrieval.
- **Spam Detection**: LLMs can identify spam emails, comments, or other types of unwanted content, enhancing cybersecurity.

5. Personalization and Recommendations

- **Personalized Recommendations**: LLMs can analyze user preferences and behaviors to make personalized product, content, or service recommendations.
- **Content Tagging**: LLMs can automatically tag and categorize content, making it easier to organize and recommend relevant items to users.

6. Academic and Scientific Research

- **Research Assistance**: LLMs can assist researchers in finding relevant papers, summarizing research findings, and generating hypotheses.

7. Accessibility and Inclusivity

- **Text-to-Speech**: LLMs can convert written text into spoken words, making content accessible to visually impaired individuals.
- **Language Generation for Non-Native Speakers**: LLMs can help non-native speakers generate more fluent and accurate text in a given language.

These are just a few examples of the many use cases for Large Language Models. The versatility and capabilities of LLMs continue to expand as research and development in this field progress.

3.6.2 How LLM Can be used in legal sector

Large Language Models (LLMs) in the legal sector assist with various tasks, enhancing efficiency and accuracy:

1. Document Review

- **LLMs** can quickly review and analyze large volumes of legal documents, making them particularly useful for tasks such as due diligence, contract review, and e-discovery during litigation.

2. Legal Research

- **LLM tools** can help lawyers find relevant case law, statutes, and legal precedents, saving time and improving the quality of their legal research.

3. Predictive Analytics

- **LLMs** can predict case outcomes based on historical data, aiding lawyers in assessing the viability of a case and potentially facilitating dispute settlements outside of court.

4. Contract Analysis

- **LLMs** can extract and analyze key information from contracts, ensuring compliance and identifying potential risks.

5. Natural Language Processing (NLP)

- **NLP-based LLM tools** can automate the drafting of legal documents, generate client communications, and assist with regulatory compliance.

6. Administrative Tasks

- **LLMs** can automate administrative tasks in law firms, such as appointment scheduling, document organization, and client communication, allowing lawyers to focus on more complex legal work.

In this master thesis, we focus on two specific applications of Large Language Models (LLMs) in the legal sector: contract analysis and the generation of clauses.

3.7 INFORMATION RETRIEVAL

In the digital age, the exponential growth of data has created a pressing need for efficient methods to access and retrieve relevant information. Information retrieval (IR) is a field within computer science that addresses this challenge by developing techniques and systems to locate the most pertinent data from vast repositories based on user queries. At its core, IR involves indexing, searching, and ranking documents or data items to

satisfy information needs effectively.

Traditional IR systems rely on keyword-based matching, where the occurrence of specific terms in both the user's query and the documents determine their relevance. These systems are foundational in various applications, including web search engines, digital libraries, e-commerce platforms, and enterprise content management systems. The objective of IR is to return a set of documents or data items that best match the user's query, ranked in order of relevance.

The process of information retrieval has several key components:

- **Query Processing**: Understanding and interpreting the user's request for information.
- **Indexing**: Organizing data to enable fast and efficient retrieval.
- **Search Algorithms**: Techniques for matching queries to documents.
- **Ranking**: Ordering the search results based on their relevance to the query.

The topic of interest to the user is referred to as the "information need." The IR process begins when the user inputs a query into the system. Queries are formal declarations of informational needs, similar to search strings used in web search engines. Unlike traditional SQL database queries, IR queries cannot pinpoint a single object within the collection. Instead, multiple objects may correspond to the query with varying degrees of relevance.

Semantic Search

Traditional IR systems predominantly depend on keyword matching,

determining relevance based on the presence of specific terms in both the query and the documents. Although this method has proven effective, it frequently fails to grasp the deeper meaning and context of user queries.

To overcome these limitations, **semantic search** has emerged as an advanced extension of information retrieval. **Semantic search** aims to improve the accuracy and relevance of search results by comprehending the context, intent, and meaning behind user queries, rather than simply matching keywords. This approach utilizes natural language processing (NLP), machine learning, and knowledge graphs to interpret the semantics of both queries and data.

By incorporating semantic search techniques, IR systems can provide more precise and contextually appropriate results, greatly enhancing user satisfaction and efficiency in locating relevant information. This advancement is especially critical in fields such as legal research where the precision and relevance of retrieved information are most important.

3.8 RETRIEVAL AUGMENTED GENERATION

Retrieval-Augmented Generation (RAG) [11]is an advanced technique in the field of Natural Language Processing (NLP) that merges the strengths of pre-trained language models with the advantages of information retrieval systems. The primary objective of RAG is to enhance the capabilities of large language models (LLMs), particularly for tasks that demand a deep understanding and generation of contextually relevant responses.

Traditional language models generate responses based solely on their input and the information they were trained on. While effective in many scenarios, these models often struggle with tasks requiring specific, factual

knowledge or the ability to reference multiple sources of information. RAG addresses these limitations by integrating retrieval mechanisms, enabling models to access and utilize external data, thereby improving accuracy and relevance in generated responses [12].

The concept of RAG involves two main components: a document retriever and a large language model (LLM) [12]. The document retriever finds relevant information from a large corpus of documents based on the input query using semantic search. This information is then passed to the LLM, which generates a response. What sets RAG apart is its joint process where document retrieval and response generation are intertwined, allowing the model to consider multiple documents simultaneously when generating a response. This results in more accurate and contextually relevant outputs.

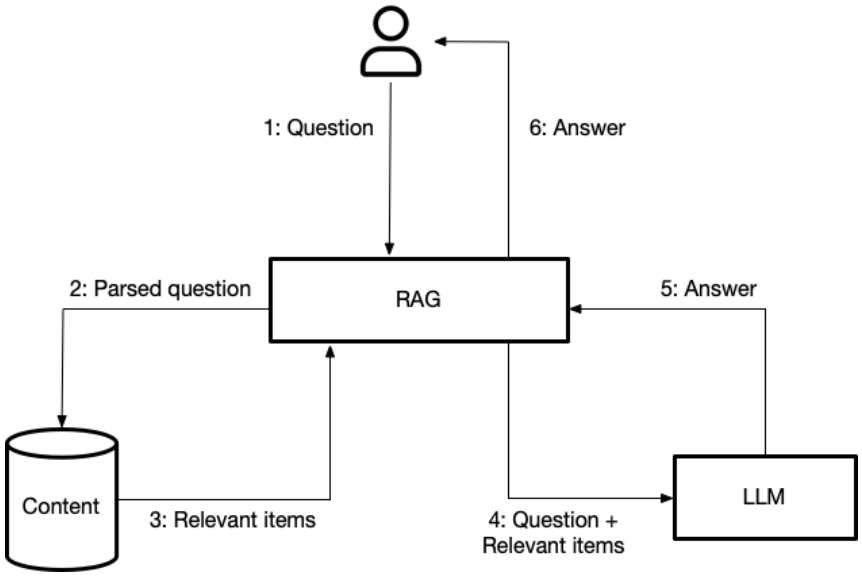


FIGURE 3.6:RAG ARCHITECTURE [13]

The RAG approach is particularly effective for tasks requiring a deep understanding of context and the ability to reference multiple sources of

information. This includes tasks such as question answering, where the model needs to consider multiple sources of knowledge and select the most appropriate one based on the context of the question.

4. LEGAL APPLICATION

In this chapter, we explore the platform, which enables Orbyta Legal employees to improve and speed up the document analysis and the draft of new contract in a more efficient and intuitive way. The proposed platform contains main components:

Web Application: The interfaces act as a bridge between the user and the system, with user-friendly interface. It allows the use to write input text and receive response in real-time.

Retrieval Augmented Generation (RAG) system allows direct interaction between text documents and deep learning models, such as Large Language Models (LLM). This methodology enhances the generative capabilities of LLMs with an information retrieval system, improving the quality of the responses generated by AI.

4.1 APPLICATION

4.1.1 FRONT-END

The front-end of this application was constructed using streamlit an open-source Python framework for data scientists and AI/ML engineers to deliver dynamic data apps with only a few lines of code. With the rise of Large Language Models (LLMs), Streamlit [14]has rapidly become the visual UI of choice for LLM-powered apps, including chatbots, sentiment analysis tools, content summarizers, and more. For this first version of this

application, we chose streamlit because offers several advantages:

Easy to use: Streamlit allows to build interactive applications with minimal effort. Developers can create complex complex UI without extensive front-end development knowledge.

Real-time Feedback: Streamlit supports real-time updates, which is crucial for applications that require dynamic interactions, such as those utilizing LLMs for tasks like text generation, question answering, or chatbots.

Visualization: For LLM applications, Streamlit can render markdown, code snippets, and styled text, making it perfect for displaying generated text, summaries, and other outputs. Streamlit can also integrates popular libraries like Matplotlib, Ploty, enabling rich visualizations of model outputs and performance metrics.

The application has three main pages and two pages of configuration:

- **Generate clauses:** On this page, users can draft their desired clause using the text input fields. The application then generates the clause, displaying it in a response box along with additional information about the resources used in the generation process

.

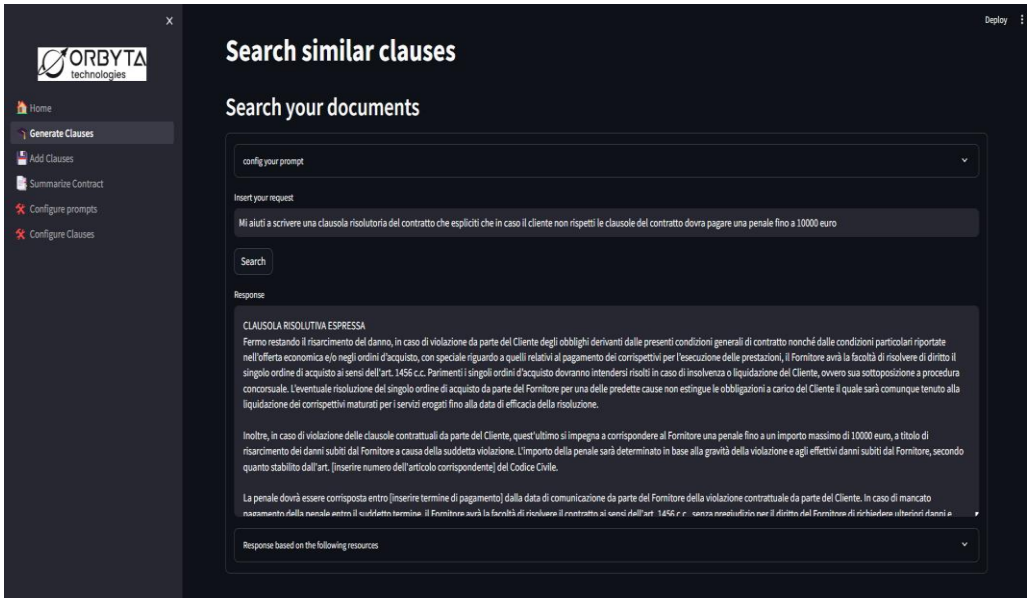


FIGURE 4.1: PAGE GENERATION CLAUSES

- **Add clause:** Users can add their clauses using the input fields. The application stores these clauses in a Redis database, allowing them to be used as resources for future clause generation.

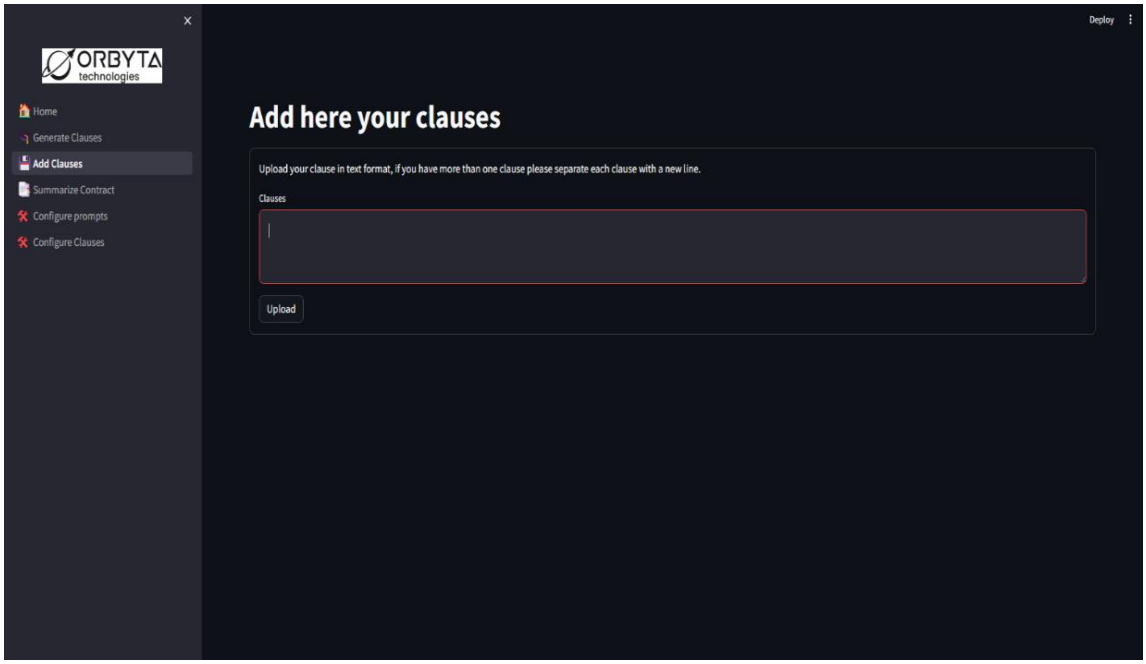


FIGURE 4.2:PAGE ADD YOUR CLAUSES

- **Summarize Contract:** Users can upload contracts for analysis using the drag-and-drop interface. The application then iteratively analyzes the contract without storing it and displays the key points of the document.

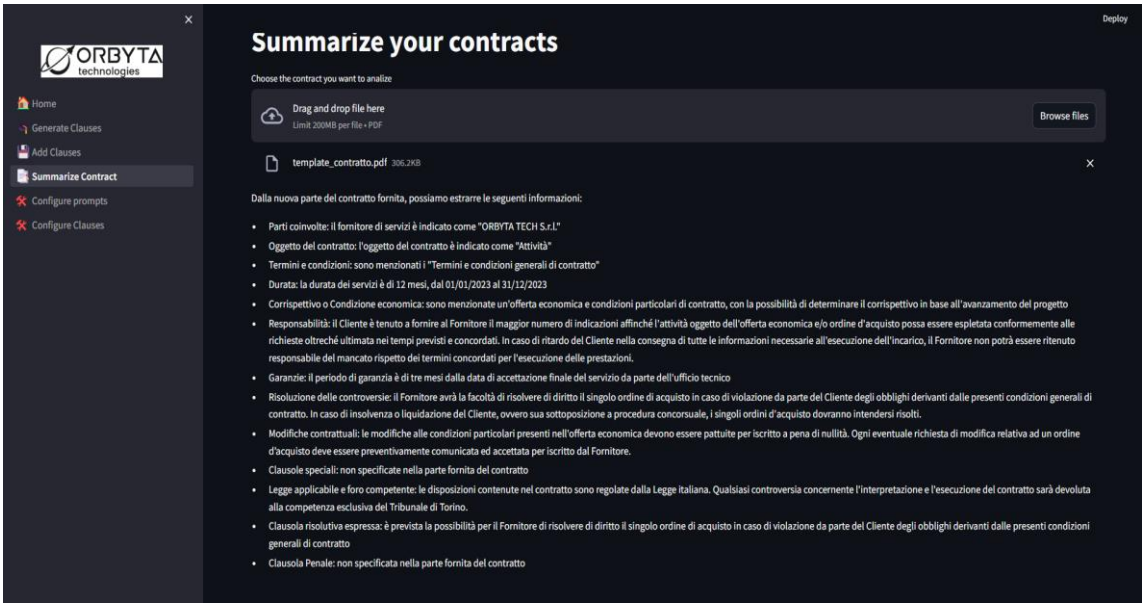


FIGURE4.1.3: PAGE SUMMARIZE CONTRACT

- **Configure Prompt:** In this page user can configure the input prompt for the generation of clauses and summarization contract.

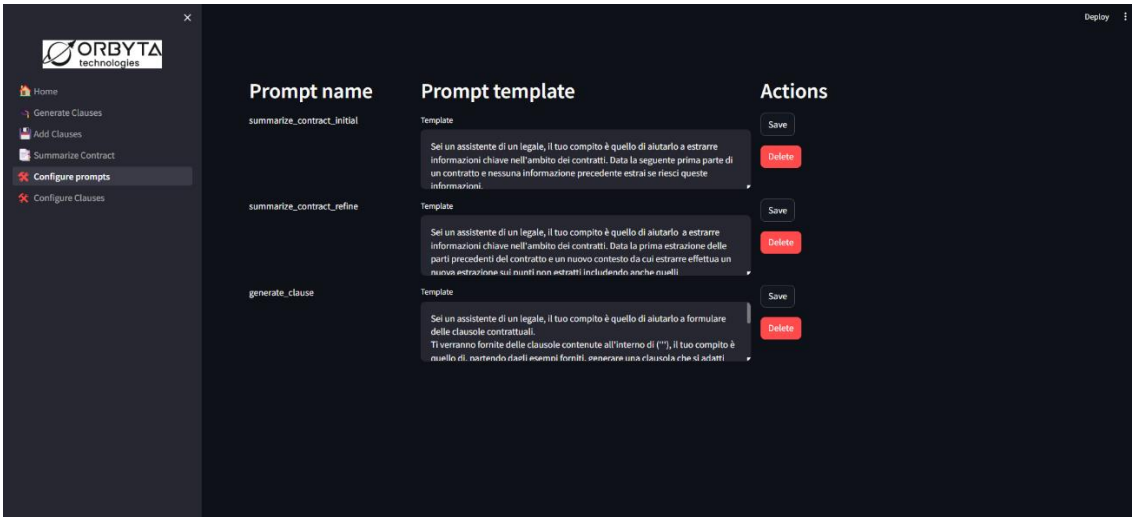


FIGURE 4.3:PAGE CONFIGURE PROMPT

- **Configure Clauses:** The user can manage the source clauses stored in Redis database. User can modify or delete clauses.

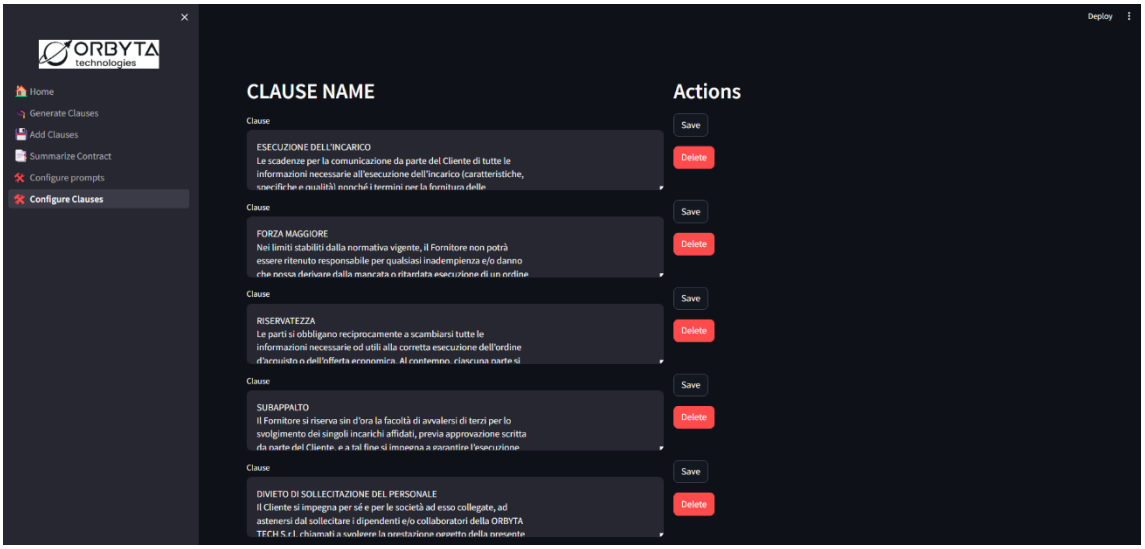


FIGURE 4.4: PAGE CONFIGURE CLAUSES

In conclusion, the last two pages are configuration pages where the administrator can manage the clause sources and configure the prompts used by the LLM.

4.1.2 BACK-END

The application leverages Redis as a database for storing initial prompts and as a vector database to store clause embeddings. At the startup, the application injects the clauses and prompts into Redis. For implementing the pipeline and manipulating chains, the application uses LangChain, an open-source framework designed for building applications based on large language models (LLMs). Chains, which are series of automated actions from the user's query to the model's output, are the fundamental principles in LangChain that ensure context-aware responses.

Main components of the application:

- **Redis Database**: Redis is used for storing initial prompts and

clause embeddings, making it a crucial component for fast and efficient data retrieval and manipulation. As Vector DB stores clauses embeddings, nabling quick semantic searches and similarity matching.

- **Langchain**: LangChain is utilized for creating and managing the AI-driven workflows within the application. It provides several modules to build robust, context-aware language model systems.
    - **APIs**: LangChain provides APIs for developers to connect and query various LLMs, including public and proprietary models like GPT, Bard, and PaLM. This simplifies integration by allowing simple API calls instead of complex code.
    - **Prompt Templates**: Developers can create and use prompt templates to consistently format queries for AI models. These templates can be reused across different applications and language models, ensuring consistency and precision.
    - **Custom Chains:** LangChain provides tools and libraries to compose and customize chains for complex applications. An agent in LangChain prompts the language model to determine the best sequence of actions in response to a query.
    - **RAG Systems:** LangChain supports the development of Retrieval-Augmented Generation (RAG) systems, offering tools to transform, store, search, and retrieve information

that refines language model responses.

The integration of Redis for efficient data storage and LangChain for managing AI workflows makes this application robust and contextually aware.

4.2 CHUNKING

In the world of information retrieval and machine learning, document chunking plays a critical role, particularly in the context of Retriever-Augmented Generation (RAG) systems [15]. Document chunking involves breaking down large texts into smaller, manageable pieces or "chunks" that are easier for computational models to process and analyze. This is crucial in scenarios where the answer to a query might span different sections of a document or when handling large datasets that exceed the processing capacity of standard models. Chunking is essential for several reasons:

- **Efficiency**: Processing smaller sections of text can dramatically speed up computation time, making applications more efficient.
- **Accuracy**: By focusing on smaller text segments, models can more accurately associate queries with relevant text pieces, improving the accuracy of the responses.
- **Scalability**: Chunking allows systems to scale by handling larger documents.

DOCUMENT READER

To perform document retrieval, we need to load the documents into our system and for this purpose we use Document Reader. In this application, we manage two types of document readers offered by LangChain:

- **PyPDFLoader**: used to read contracts before chunking them. It efficiently handles the complexities of PDF files and extracts the text content for further processing

- **TextLoader:** used to manage the injection of clauses and prompts. It ensures that textual data is loaded efficiently for subsequent chunking and processing.

TEXT SPLITTER

LangChain offers multiple splitters, each suited for specific types of text and applications. For this purpose, we use the Character Text Splitter. It is used in the loading of the source clause with a chunk size of 800 and separator "\n\n" to ensure that each clause is treated as a separate chunk, avoiding cutoffs in the middle of clauses.

4.2.1 TEXT EMBEDDER

The text embedding model transforms the chunks extracted by the Document Reader or the user's query into vector representations. Significant advancements over the past decade have led to the development of models capable of creating vector representations of words or phrases that encode semantic relationships. This ability to represent semantic connections is crucial for assessing the relevance of specific text portions in relation to the user's inquiry.

The following features were considered when selecting the model to be used:

- **Number of Input Tokens**: A model capable of handling a large number of input tokens offers significant benefits. It minimizes the need for additional text segmentation, optimizing the amount of data stored in the database. Additionally, it accelerates search

operations while maintaining high accuracy and relevance in the responses.

- **Supported Language**: Given the project's nature and target audience, it was crucial that the chosen model adequately supported the Italian language. This ensures that linguistic and cultural nuances specific to Italian are accurately captured and represented.

- **Cost and Performance**: Beyond evaluating the model's effectiveness, practical aspects such as cost, speed, and reliability were also considered. The balance between cost and performance led to the selection of a model that provides the best value for the project's requirements.

Text-Embedding-Ada-002

Text-Embedding-Ada-002 (TEA-002) is a general-purpose text embedding model released by OpenAI in late 2022. This model integrates and enhances the performance of all previously released OpenAI embedding models, excelling in tasks such as search, similarity, and retrieval. In addition to performance improvements, it supports larger input sizes and restricts output length, making it suitable for embedding very long text sequences while still producing low-dimensional vectors.

Since it is a closed model, details about its training and the datasets used have not been disclosed. However, a significant advantage of using this model is its availability as a service through dedicated APIs, which eliminates infrastructure and maintenance costs. This also reduces long-term costs and avoids dependency on potential malfunctions of OpenAI's proprietary systems. The model is cost-effective, with a pricing of €0.00001 per 1,000 tokens. Estimating that a typical document page

includes approximately 8,000 tokens, converting 10,000 document pages would amount to €1.

Moreover, converting documents into vectors incurs a one-time expense, with the only recurring cost being the conversion of user queries. This cost structure makes Text-Embedding-Ada-002 an economical choice for embedding large volumes of text efficiently.

Text-Embedding-Ada-002 can be utilized with OpenAi API or Microsoft Azure OpenAI's APIs but there is different performance [16].

The comparison between OpenAI and Azure on text-embedding-ada-002 is easy to characterize Azure's outputs are identical given the same input, whereas OpenAI's outputs are noisy. In other words, don't expect to get the same embedding vector back from OpenAI's ada-002 implementation. As can be seen in Figure 4.2.1, OpenAI produces about 10 or so unique embeddings per 100 trials of the same input sentence, whereas Azure produces 1 in each case.

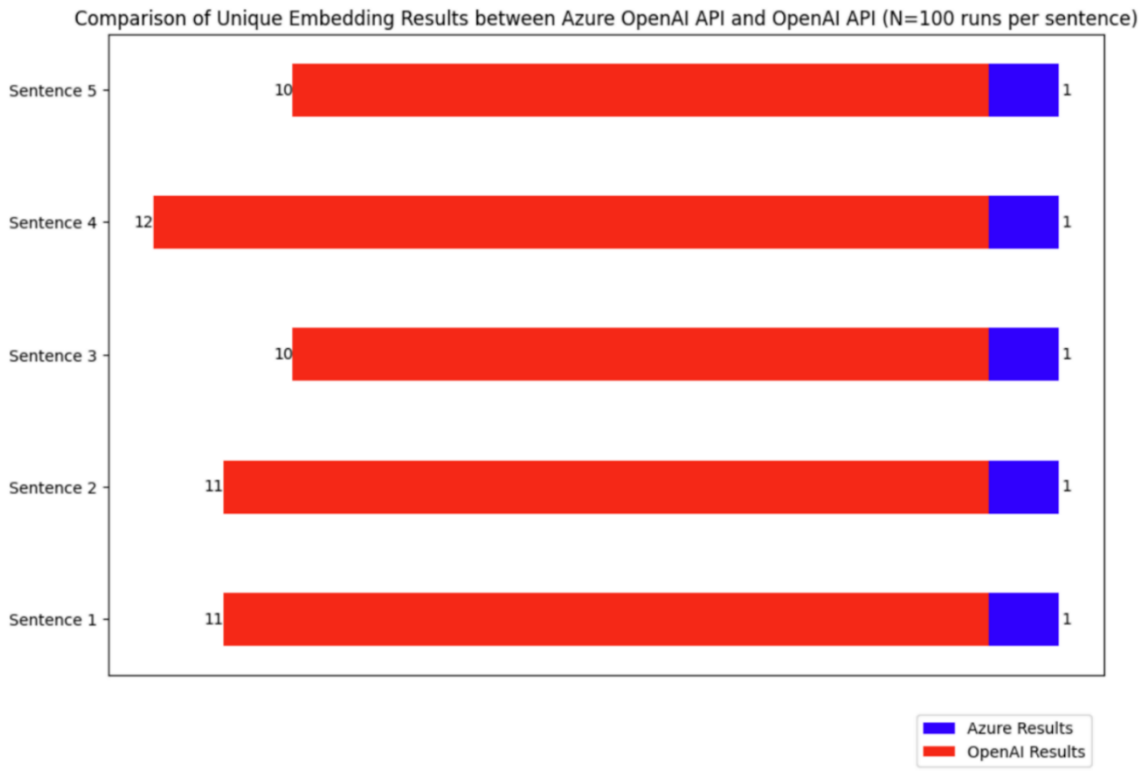


FIGURE 4.5:COMPARISON OPENAI AND AZURE EMBEDDING [3]

How this impacts a retrieval augmented generation (RAG) system or another system reliant on text embeddings depends critically on the use case and data in question. In order to have a stable application we use the AZURE OpenAI endpoint.

4.3 VECTOR DATABASE

Databases play a crucial role in the architecture of any computer system, particularly in those involving machine learning or artificial intelligence models. Their effectiveness and efficiency have a direct impact on the model's performance, especially in terms of response speed.

Vector databases, a subtype of non-relational databases, utilize vectorized data representation. Unlike conventional keyword-based databases that

search for exact matches, vector databases perform searches based on the semantic meaning of the data. This capability allows them to find matches that are contextually or semantically related to the query, which is particularly advantageous for tasks like similarity search or classification functions. Furthermore, due to their non-relational nature, vector databases can easily store associated metadata, such as file names, resource links, or contained images.

4.3.1 REDIS

Redis [17] is renowned for its exceptional performance, attributed to its in-memory architecture which provides rapid response times for both read and write operations. This makes it highly suitable for tasks requiring the processing of large amounts of vector data, such as similarity searches. Redis supports two types of vector indexing: Flat and Hierarchical Navigable Small Worlds (HNSW).

- **Flat Indexing**: This method involves a brute-force search, examining every vector in the database to find the one most like the query vector. Although straightforward, this technique can be inefficient and slow with large datasets.

- **Hierarchical Navigable Small World (HNSW) Indexing:** HNSW is an advanced indexing and searching algorithm that creates a hierarchical structure of vectors. This structure allows for much more efficient similarity searches compared to flat indexing, especially in large datasets. HNSW traverses this hierarchical structure to quickly locate the most similar vectors, significantly reducing the number of comparisons needed.

Redis's in-memory capabilities are particularly beneficial for operations

involving vectors, such as similarity searches, due to the exceptional speed in read and write operations. Additionally, Redis's scalability is a notable feature; it can be easily scaled up as datasets grow to accommodate larger workloads, ensuring consistent response times and maintaining optimal performance.

4.4 MODEL CHOICE

Large Language Models (LLMs) are sophisticated artificial intelligence systems trained using vast datasets and deep learning methods, especially the transformer architecture. These models excel at comprehending and generating human-like text, allowing them to tackle a wide array of natural language processing (NLP) tasks with high accuracy and fluency. By employing advanced algorithms, LLMs process and analyze text efficiently, deriving valuable insights, producing coherent responses, and enhancing human-machine interactions through natural language. Their applications span diverse sectors such as content creation, customer service, healthcare documentation, and beyond.

Howewer, the process of training such models entails significant financial and computational expenditures, making it a viable option primarily for well-resourced entities. Consequently, only a select group of major corporations, including industry leaders such as Google, Meta, Microsoft, and OpenAI, can afford to engage in the intensive research and development necessary for these advanced models. This economic barrier effectively limits participation to those with substantial capital, highlighting a disparity in the technological advancement capabilities among different sized entities within the field.

When assessing LLMs, it is crucial to consider several important factors:

- **Task**: Identification of the task that LLM needs to perform. Different

models excel in different areas.

- **Model Size**: LLMs vary in size, from smaller models like GPT-2 to massive architectures like GPT-4 or Claude. Consider the trade-off between model size and computational resources. Larger models generally provide superior performance but are more expensive and demand significant computational power and memory, especially if you manage the hosting internally. This balance is crucial for effectively deploying LLMs, as it impacts both the feasibility and sustainability of various applications.

- **Resource Constraints**: Evaluation of the computational resources, including GPU availability, memory capacity, and the speed required for generating inferences. The selection of an LLM that fits within the resource limits while still maintaining effective performance. Latency is a critical consideration—if immediate responses are necessary, choosing for a faster, more streamlined model may be required, which might involve reducing model complexity. Additionally, consider the availability of **APIs** that can facilitate integration and streamline operations, offering a balance between performance and resource management.

- **Open-Source and closed-source models:** Numerous open-source models utilize architectures like LLama and LLama 2, which were developed and released by Meta. These foundational models facilitate extensive experimentation and optimization, allowing for specific adaptations such as Alpaca and Vicuna. While these fine-tuned models exhibit strong performance, they often strive to reach the efficacy of leading proprietary models such as Bard or GPT-4.

- **Data Privacy**: Consider ethical implications such as bias, fairness, and data privacy when selecting an LLM. Ensure that the model

aligns with ethical guidelines and principles to mitigate potential risks.

The market now offers APIs that allow for the use of models without the necessity of local installation. This arrangement means that the costs for machine management and maintenance are incorporated into the service fees, eliminating the need for users to manage these aspects independently. Another significant advantage is the immediacy with which these services can be utilized; unlike with open models, there is no delay. With open models, one must account for the time required for fine-tuning. Additionally, the performance of these API-based models is exceptionally high, with those provided by OpenAI being among the best available.

For the privacy maintenance Azure OpenAI guarantees that the data submitted remains within Microsoft Azure and is not passed to OpenAi for model predictions. Azure has sole control and governance of the data and OpenAI. Azure OpenAi [18]is the best choice for data company.

For the initial launch of this application, we have opted to use GPT-3.5-Turbo. In future, after an evaluation by the user, we will use gpt4.

4.5 PROMPT ENGINEERING

Prompt engineering [19] is the process where you guide generative artificial intelligence (generative AI) solutions to generate desired outputs. Even though generative AI attempts to mimic humans, it requires detailed instructions to create high-quality and relevant output. In prompt engineering, you choose the most appropriate formats, phrases, words, and symbols that guide the AI to interact with your users more meaningfully. Prompt engineers use creativity plus trial and error to create a collection of input texts, so an application's generative AI works as

expected.

The Prompt engineering bridge the gap between the end users and the large language model. Prompt engineering makes AI applications more efficient and effective.

Prompt engineering [20] is a dynamic and evolving field. It requires both linguistic skills and creative expression to fine-tune prompts and obtain the desired response from the generative AI tools.

A good prompt is composed by:

- **Role**: specifies the position or persona that the prompt assigns to an individual, aiding the AI in crafting responses that are pertinent to that specific character. For instance, if the prompt says: “You are a layer specialized in the writing contract.” Using the term “layer specialized in..” allows the AI to create a response in a legal tone appropriate for contract support.

- **Istruction/task**: This refers to a clear outline of what specific action or response the AI is expected to generate. For example, “Compose a rescission clause for a furniture contract” is a prompt asking the AI to generate a rescission clause by taking in consideration the type of contract

- **Context**: Adding further contextual information significantly enhances the AI-generated response by making it more relevant and accurate for the specific scenario. In Rag system the contextual information is given by the result of the retrieval of stored documents.

- **Example**: An effective learning strategy can be adding examples to the prompts, which further attracts the AI’s attention and sets clear

expectations for the type of information required.

The combination of this elements helps to obtain the desired response by the LLM.

4.6 FINE-TUNING

Fine-tuning involves training a pre-trained model on a specific dataset or task. This process fine-tunes the model’s parameters to adapt it to a particular task, making it more specialized. For example, you could fine-tune a GPT model on a dataset of text summarization examples. This would train the model to generate summaries that are more accurate and relevant than summaries generated by a model that has not been fine-tuned. Fine-tuning allows you to optimize the model for a specific task, resulting in better performance and reduce the prompt dependency. In other ways, it is difficult to search a task-specific dataset in order to train the model.

4.7 RAG VS FINETUNING VS PROMPT ENGINEERING

"Finetuning”, “prompt engineering" and “Rag” are three approaches used to adapt and optimize language models, particularly Large Language Models (LLMs), to specific tasks.

Finetuning is when you take the language model and make it learn something new or special. Think of it like updating an app on your phone to get better features. But in this case, the app (the model) needs a lot of new information and time to learn everything properly. It's a bit like going back to school for the model.

Finetuning needs a lot of computer power and time, it can be expensive. It offers the advantage of being able to adapt the model to very specific scenarios, thereby greatly improving its performance, it also has critical issues. Indeed, training these models requires significant computational resources and, in the presence of limited data sets, can reduce overall performance and expose the model to the risk of overfitting.

In contrast, "prompt engineering" focuses on the curation and optimization of the prompt or the initial input in the form of an instruction or question given to the model to guide its response. This technique requires no additional computational resources for training and offers considerable flexibility, allowing the model to be adapted to different tasks without changing its structure. However, finding the ideal prompt may require iterative experimentation and may not guarantee the same effectiveness as ’fine-tuning’ in some contexts.

Retrieval Augmented Generation, or RAG, mixes the usual language model stuff with something like a knowledge base. When the model needs to answer a question, it first looks up and collects relevant information from a knowledge base, and then answers the question based on that information. It's like the model does a quick check of a library of information to make sure it gives you the best answer.

In practice, these techniques can be combined or used in tandem to achieve optimal results. In our application we create a system could employ RAG to retrieve relevant information, and then use prompt engineering to guide the model's generation for a specific task.

Criteria	Retrieval-Augmented Generation (RAG)	Fine-Tuning
Training Time	Faster than fine-tuning[8][18]	Slower, depends on model size, dataset size, and compute resources[1][3]
Computational Cost	Lower than fine-tuning[2][8]	Higher, especially for large models[1][3]
Accuracy Improvement	Good for leveraging existing knowledge and domain expertise[5][18]	Better for complex tasks and learning patterns[5][18]
Model Size	No significant change in model size	Depends on the fine-tuning algorithm and model size
Training time	Shorter, as it only requires updating the query encoder and the generator	Longer, as it requires updating the entire model parameters
Training data	Smaller and more general, as it can leverage external knowledge sources	Larger and more specific, as it needs to cover the task or domain of interest
Memory usage	Higher, as it also stores the external knowledge sources	Lower, as it only stores the model parameters
Scalability	Higher, as it can handle different tasks or domains by querying different knowledge sources	Lower, as it needs to retrain the model for each new task or domain

FIGURE 4.6:RAG VS FINE-TUNING [21]

4.8 DEPLOYMENT

The deployment process is a critical phase in an application's lifecycle, shaping how and when users can access and engage with the application.

Docker is an open-source platform that assists developers in creating, shipping, and running distributed applications through containers. These containers offer a lightweight and portable method to package an application with all its dependencies, including libraries and configuration files, into a single consolidated image. On the other hand, Azure, Microsoft's cloud computing platform, provides a wide range of services, including computing, storage, and networking capabilities. The first step involves creating an Azure Container Registry (ACR) which will serve as a

repository for Docker images. ACR is a private registry that allows you to store and manage Docker images specific to an organization. A Dockerfile is a text file containing instructions for building a Docker image, outlining the base Docker image to use, the location of the application source code to be included in the image, as well as the libraries, packages, and other dependencies necessary for the application's operation. Docker interprets and follows the instructions in this file to construct the image.

The Dockerfile sets up a Python environment tailored for running a Streamlit application and install the libraries contained in the requirements.txt file

FIGURE 4.7:DOCKER FILE

Docker Compose is a tool that simplifies the process of defining and managing multi-container Docker applications. Through a single YAML file, developers can configure all the necessary services for an application, manage volumes, networks, and inter-service dependencies. This allows for starting and stopping all services with a single command, greatly facilitating the implementation of complex environments.

In the docker compose was setting-up a service that includes a web

application and a Redis instance using the redis/redis-stack image.

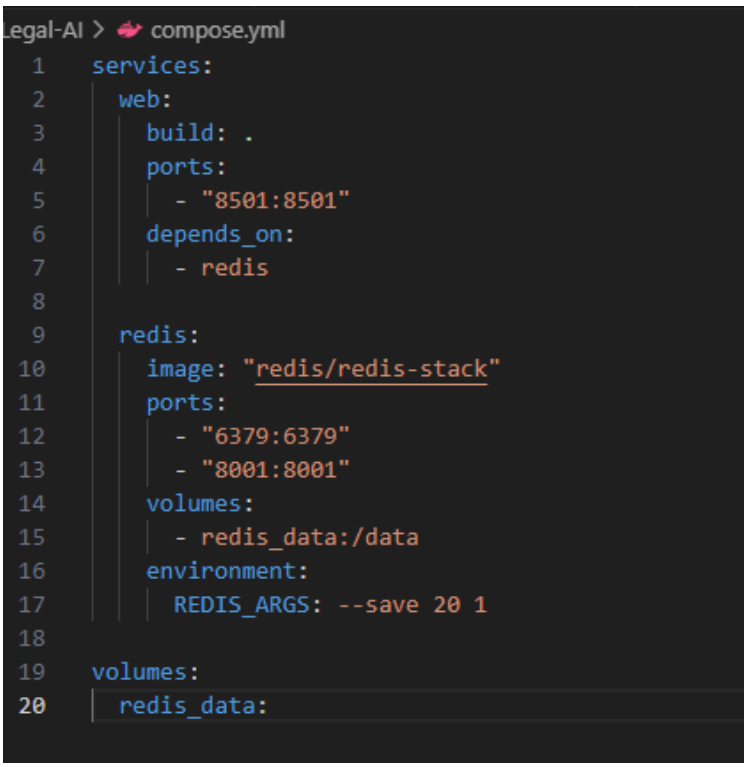


FIGURE 4.8:DOCKER FILE

After the image is created, it is pushed to the registry

Access to the application is limited to the company network and is restricted to the developer and legal-office.

5. METHODOLOGIES

The aim of this research was to provide support to lawyers through the use of large language models (LLM) to speed up time-consuming processes such as reading contracts and drafting model clauses. In this chapter, we will analyse the methodologies used to achieve this goal. We will explore both the prompt configurations used and other parameters for the various tasks in order to understand how LLMs can be effectively adapted to the specific needs of legal professionals.

5.1 CONTRACT SUMMARIZATION

In the legal field, extracting information from contracts is challenging primarily due to the scarcity of annotated data. Utilizing advanced models such as the Generative Pretrained Transformer (GPT) offers a promising solution. However, these models face limitations with their inherent token capacity, which can hinder the processing of extensive legal documents.

The main challenges in contract analysis using state-of-the-art models include not only the limited data available for training or fine-tuning to achieve high accuracy but also the substantial size of many contracts. This often exceeds the processing capabilities of current transformer architectures. Transformer-based models are constrained by a maximum sequence length, and contracts exceeding this limit may need to be segmented into smaller parts, complicating the analysis process.

Following the methodology outlined in the paper, we decided to implement summarization using AzureOpenAI. However, we encountered a limitation due to the model's context length.

The **context length limitation** is a significant hurdle: while large language models (LLMs) have many capabilities, they struggle to

synthesize large documents or process extensive text effectively due to this constraint. Current models are capable of handling input and output lengths ranging from 4,096 to 16,384 tokens, corresponding approximately to 6.4 to 26.5 pages of text. "Tokens" are the basic units of processing for LLMs, typically representing about three-quarters of a word. Various models use different tokenization methods, and each model has a specified "input context length" which indicates the total number of tokens it can handle at once, with a designated portion reserved for generating output.

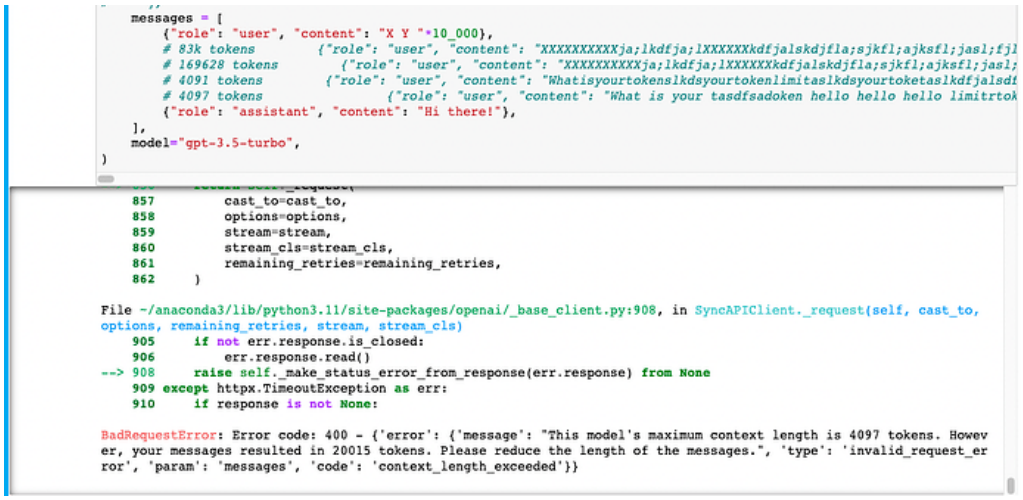


FIGURE 5.1:ERROR CONTEXT LENGHT

Contract can vary significantly in length, typically ranging from 10 to 100 pages. Analysing such extensive documents is a substantial undertaking due to their size. In our initial attempt to automate this process in our application, we implemented a summarization feature. However, we encountered constraints related to the context length that Large Language Models can handle, as they are limited to a certain number of tokens per session.

To address this issue, we adopted a preprocessing approach, dividing the

document into smaller segments, each containing a single page of the contract, to keep the number of tokens within the model's allowable limits. Initially, we configured our system to send only a portion of the document along with a carefully designed prompt for summary generation to the first API call. In subsequent API calls, we integrated the previously generated summaries into the prompt, aiming to preserve context and enhance the consistency of the generated content.

```python
def refine_chain(
        initial_prompt: BasePromptTemplate,
        refine_prompt: BasePromptTemplate,
        chunks: List[Document],
        model: BaseLanguageModel,
        output_parser: BaseOutputParser):

    chain = initial_prompt | model | output_parser

    output = chain.invoke(input={'context': chunks[0].page_content})

    chain = refine_prompt | model | output_parser
    for chunk in chunks[1::]:
        output = chain.invoke(input={'context': chunk.page_content, 'existing_summary': output})

    return output
```

FIGURE 5.2:CALL CHAIN

This method is successfully to avoid context limitations but was inefficient in terms of computational and token cost.

While effective in meeting token constraints, this method has shown inefficiencies. Segmenting the document into isolated parts may lead to a loss of overall context, potentially reducing the accuracy and relevance of the summaries produced. Furthermore, the accumulation of summaries in successive queries increases the consumption of tokens for each subsequent call, which may become onerous and less manageable as the length of the original document increases. These problems highlight the need to explore further improvements or alternative technologies to efficiently handle large volumes of text in legal contexts.

SOLUTION

After careful analysis of the documents and discussion with the legal team, we identified that, in reviewing the contracts, the lawyers were looking for specific information that tended to be recurring. Consequently, we decided to focus on extracting key points from the documents. Key points typically required in a contract include:

- **Parties involved**: identification of the legal entities participating in the contract.
- **Subject matter of the contract**: description of the agreement and the objectives of the contract.
- **Terms and conditions**: details of the rules and regulations governing the contract.
- **Duration**: period for which the contract is valid.
- **Consideration or economic condition**: details of financial aspects, such as payments or fees.
- **Liability**: obligations and duties of the parties.
- **Guarantees:** assurances given by one or both parties.
- **Dispute resolution**: procedures for handling disputes related to the contract.
- **Contract changes**: conditions under which the contract may be changed.
- **Special clauses**: unique or specific stipulations necessary for certain situations.

- **Applicable law and jurisdiction**: law under which the contract is interpreted and jurisdiction for disputes.
- **Terms and conditions**: details of the rules and regulations governing the contract.
- **Express termination clause**: conditions allowing termination of the contract.
- **Penalty Clause**: penalties for non-compliance with the terms of the contract.

This strategy aims to simplify the contract review process, improving efficiency and reducing the time needed to review legal documents.

Algorithm 1 PSEUDO-CODE CONTRACT EXTRACTION

```
procedure Load(contract)
    chunks ← split(contract,by pages)
    chain←InitializeChain(initial prompt,model,output parser)
    extraction ← InvokeChain(chunks[0].content)
    for i = 1 to length(chunks) - 1 do
        extraction ← RefineChain(chunks[i].content,extraction)
    end for
    return extraction
end procedure
```

The prompts are designed for a legal application that acts as a digital assistant for lawyers, with the specific purpose of facilitating the extraction of key information from contracts. This AI tool uses a system based on Large Language Models (LLMs) to analyse the contract text and identify relevant components such as parties involved, subject matter of the contract, terms and conditions and others.

The first part of the prompt can be divided into:

- **Role**: “You are a legal assistant. Your main function is to help the lawyer in extracting relevant information from contracts”.
- **Task:** “Given the first part of a contract below and no previous information, extract the following essential information”.
- **Context:** The first page of the loaded contract
- **Key Extraction**: the list of information to be extracted. This list can be modifying in the prompt configuration as mentioned in the paragraph 4.1.

After the first extraction, we implement an iterative loop in which we create a new chain [Algorithm 1]. In this chain, we progressively include the previous extraction in the prompt. This process allows us to continuously update the extracted information as we proceed with the analysis of subsequent pages of the document. This methodology ensures that each new page analysed enriches the overall context, improving the accuracy and completeness of the information extracted from the contract in subsequent stages. This methodology ensures that each new parsed page enriches the overall context, improving the accuracy and completeness of the information extracted from the contract in subsequent steps.

5.2 GENERATION CLAUSES

The advent of artificial intelligence (AI) has led to a significant transformation in the legal services landscape, introducing new possibilities for automation and process optimisation. In particular, contract management benefits greatly, with considerable time savings and a reduction in manual errors.

To take full advantage of these opportunities, we have developed an innovative virtual assistant based on Large Language Model (LLM). This intelligent tool enables the automatic generation of draft contract clauses using archived data, significantly speeding up the drafting process and minimising errors associated with manual procedures.

The lawyer can request the system to create a contract clause. Starting from a set of standard and previously used clauses, the system can modify and adapt them to the lawyer's needs, providing flexibility and customisation in the drafting of the contract.

Solution

The first step in implementing the clause generator was to collect the clauses commonly used by the law firm. These were carefully categorised by type, allowing for systematic and organised management. After extracting them from an Excel file, the clauses were saved in the Redis database.

When a user submits a request, the system transforms this request into embeddings, i.e. vector representations that facilitate semantic comparison. It then performs a similarity search among the stored clauses, selecting and returning the three most relevant results. These serve as

context for a subsequent interaction with a Language Model (LLM), for which a specific prompt has been prepared detailing:

- USER REQUEST: {query}
- **Role**: You are a legal assistant; your task is to assist in the formulation of contractual clauses.
- **Task**: "You will be provided with clauses contained within (""), your task is, from the examples provided, to generate a clause that fits the user's request."
- **Context**: The result of the similarity search.

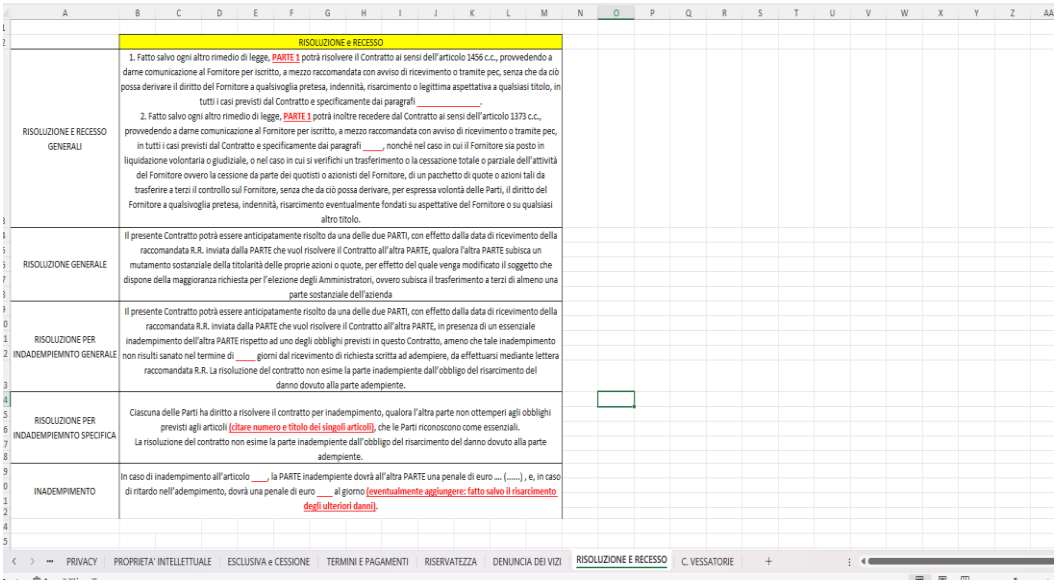
Using Langchain to generate the required clause, we set the model temperature to 0.3 to balance the creativity and accuracy of the generated clauses.

The ability to expand or modify the assistant's knowledge by adding or editing clauses in the dedicated section further increases the versatility of the system.

Benefits of this virtual assistant include:

- **Time savings**: Automation in the draft and clause generation process dramatically reduces the time needed to draft contracts, freeing lawyers for more value-added activities.
- **Reduced errors**: Minimising manual work reduces the risk of human error, resulting in more accurate and reliable contracts.
- **Increased efficiency**: Automation in the contract drafting process significantly improves law firm efficiency, optimising the use of resources and increasing productivity.

- **Customisation**: The ability to generate customised clauses gives lawyers the flexibility to tailor contracts to each client's specific needs, ensuring a highly personalised service.



	RISOLUZIONE e RECESSO
RISOLUZIONE E RECESSO GENERALI	1. Fatto salvo ogni altro rimedio di legge, PARTE 1 potrà risolvere il Contratto ai sensi dell'articolo 1456 c.c., provvedendo a darne comunicazione al Fornitore per iscritto, a mezzo raccomandata con avviso di ricevimento o tramite pec, senza che da ciò possa derivare il diritto del Fornitore a qualsivoglia pretesa, indennità, risarcimento o legittima aspettativa a qualsiasi titolo, in tutti i casi previsti dal Contratto e specificamente dai paragrafi _____. 2. Fatto salvo ogni altro rimedio di legge, PARTE 1 potrà inoltre recedere dal Contratto ai sensi dell'articolo 1373 c.c., provvedendo a darne comunicazione al Fornitore per iscritto, a mezzo raccomandata con avviso di ricevimento o tramite pec, in tutti i casi previsti dal Contratto e specificamente dai paragrafi ____, nonché nel caso in cui il Fornitore sia posto in liquidazione volontaria o giudiziale, o nel caso in cui si verifichi un trasferimento o la cessazione totale o parziale dell'attività del Fornitore ovvero la cessione da parte dei quotisti o azionisti del Fornitore, di un pacchetto di quote o azioni tali da trasferire a terzi il controllo sul Fornitore, senza che da ciò possa derivare, per espressa volontà delle Parti, il diritto del Fornitore a qualsivoglia pretesa, indennità, risarcimento eventualmente fondati su aspettative del Fornitore o su qualsiasi altro titolo.
RISOLUZIONE GENERALE	Il presente Contratto potrà essere anticipatamente risolto da una delle due PARTI, con effetto dalla data di ricevimento della raccomandata R.R. inviata dalla PARTE che vuol risolvere il Contratto all'altra PARTE, qualora l'altra PARTE subisca un mutamento sostanziale della titolarità delle proprie azioni o quote, per effetto del quale venga modificato il soggetto che dispone della maggioranza richiesta per l'elezione degli Amministratori, ovvero subisca il trasferimento a terzi di almeno una parte sostanziale dell'azienda
RISOLUZIONE PER INDADEMPIEMNTO GENERALE	Il presente Contratto potrà essere anticipatamente risolto da una delle due PARTI, con effetto dalla data di ricevimento della raccomandata R.R. inviata dalla PARTE che vuol risolvere il Contratto all'altra PARTE, in presenza di un essenziale inadempimento dell'altra PARTE rispetto ad uno degli obblighi previsti in questo Contratto, ameno che tale inadempimento non risulti sanato nel termine di ____ giorni dal ricevimento di richiesta scritta ad adempiere, da effettuarsi mediante lettera raccomandata R.R. La risoluzione del contratto non esime la parte inadempiente dall'obbligo del risarcimento del danno dovuto alla parte adempiente.
RISOLUZIONE PER INDADEMPIEMNTO SPECIFICA	Ciascuna delle Parti ha diritto a risolvere il contratto per inadempimento, qualora l'altra parte non ottemperi agli obblighi previsti agli articoli (citare numero e titolo dei singoli articoli), che le Parti riconoscono come essenziali. La risoluzione del contratto non esime la parte inadempiente dall'obbligo del risarcimento del danno dovuto alla parte adempiente.
INADEMPIMENTO	In caso di inadempimento all'articolo ___, la PARTE inadempiente dovrà all'altra PARTE una penale di euro ... (.....) , e, in caso di ritardo nell'adempimento, dovrà una penale di euro ___ al giorno [eventualmente aggiungere: fatto salvo il risarcimento degli ulteriori danni].

PRIVACY | PROPRIETA' INTELLETTUALE | ESCLUSIVA e CESSIONE | TERMINI E PAGAMENTI | RISERVATEZZA | DENUNCIA DEI VIZI | RISOLUZIONE E RECESSO | C. VESSATORIE

FIGURE 5.3:FILE CLAUSES

5.3 EVALUATION

For the monitoring and evaluation of our system, we employ Langsmith, developed by Langchain. This tool supports the debugging, monitoring and evaluation of applications using Large Language Models (LLM). Langsmith offers execution logging and visualisation of pipeline components, integrated with Langchain. When analysing the architecture of a Retrieval-Augmented Generation (RAG) system, it is crucial to examine both the document retrieval and generation components. This method makes it possible to assess the quality of the model and identify areas for improvement to optimise performance. In particular, to assess the effectiveness of our clause generator, we focus on the retriever. The choice of the most appropriate retriever is based on the analysis of the feedback provided by lawyers regarding the correctness of the generated clauses, collected through Langsmith [22]. For the evaluation of the retriever, we use the following metrics:

- Context Relevance: This metric assesses how relevant the retrieved context is to the question asked. Using an LLM, it determines how well the context supports the statements needed to formulate an appropriate response.

- Contextual recall: Measures the system's ability to retrieve all essential information to answer the question. An LLM verifies that each element of the answer is supported by the retrieved context.

In addition, we analyse the speed at which the vector database provides the appropriate context by assessing the impact of the type of index used on overall system performance.

The following table illustrates the variations in the similarity score with respect to the following query: “Help me write a termination clause for a

contract that specifies that if the customer fails to comply with the contractual terms, the customer will have to pay a penalty of up to €10,000."

Similarity Score	Total Tokens	Context relevance	Context Recall	Latency(s)	User Fedback
0.9	1350	0.86	0.85	4.01	Positive
0.75	1400	**0.90**	**0.92**	**3.83**	Positive
0.5	1520	0.60	0.65	4.31	Negative
0.3	1600	0.57	0.60	5.31	Negative

The results obtained show how the parameters change as the similarity score of Redis differs. As can be seen, the latency changes very little as the parameter changes; however, the total number of tokens in the context varies significantly. We also considered this parameter with a view to future developments and upgrading to better models. As regards the type of Retriever, we decided to experiment only with Flat Indexing due to the very small dataset of clauses at our disposal.

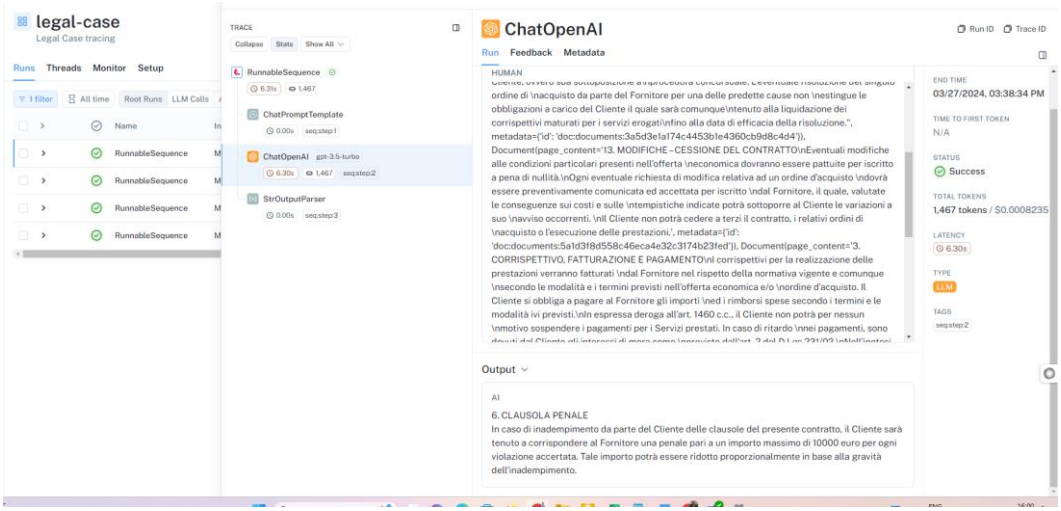


FIGURE 5.4:: FEDDBACK FROM LANGSMITH

FIGURE 5.5:TRACING LANGSMITH

6. CONCLUSION

This report explores the application of Large Language Models (LLM) in the legal sector, focusing on their use for the optimisation of contract analysis and the generation of contract clauses. The introduction of artificial intelligence technologies in this field marks a remarkable transformation, opening the way to solutions previously considered unfeasible. Specifically, we analysed the effectiveness of an LLM-based virtual assistant to automate and improve procedures traditionally handled manually by lawyers. This tool not only speeds up the process of analysing and creating customised clauses, but also helps minimise errors and increase the operational efficiency of law firms. We showed how, by supplementing the capabilities of OpenAi with an appropriate set of clauses and the use of retriever techniques and specific prompts, it is possible to develop a RAG system that effectively meets the needs of the law firm. In addition, we addressed contract privacy challenges by implementing AzureOpenAi, which meets the GDPR security standards provided by Azure.

The automated innovations in the legal industry allow lawyers to focus more on high-quality activities, greatly improving the firm's productivity.

Another crucial aspect to consider is the essential need for human supervision. Despite the significant benefits brought by automation through Large Language Models, it is essential to maintain an active, supervisory role on the part of lawyers.

The active presence of a legal professional ensures that technology is used as a supporting tool and not as a complete substitute for human decision-making, preserving the integrity of the legal process.

However, some limitations have emerged: the current system is only able

to process documents in text format and not scanned documents. In addition, it lacks a feedback mechanism to guide the model in generating clauses more precisely.

In conclusion, this pilot project offers legal professionals a safe and efficient tool, representing a valuable starting point for further development

6.1 FUTURE WORK

In this section, we look at possible future developments that can improve the system, planned for the second phase of the project.

One of the current critical points is the Document Reader, which is currently limited to reading files in PDF format. To increase the versatility of Contract Analyser, it is essential to extend support for other common file formats. This would allow users to upload and analyse contract documents independently of the original format. Considering that many legal documents are paper-based and digitised by scanning, the integration of OCR (Optical Character Recognition) technology would allow these digitised documents to be analysed as well, avoiding omitting vital information for analysis. This improvement would significantly increase the accessibility and usability of the system.

Another potential improvement is the implementation of a Feedback System integrated into the web application. This system would allow users to evaluate and comment on the answers generated by the system, improving the accuracy and relevance of the answers. The collected feedback could be used to create an annotated dataset, useful for fine-tuning the model, adapting it to the specific needs of legal users and improving the quality of the generated clauses.

In the initial phase, a relatively outdated model such as ChatGPT-3.5 was used. In the future, an upgrade to more advanced models available on

Azure is planned, evaluating the associated costs. In addition, the implementation of a local model will be considered, which would offer advantages such as increased data security and reduced dependence on external providers. This would be particularly useful for law firms with stringent requirements in terms of confidentiality and compliance with privacy regulations.

Finally, in order to improve the user interface and make the system more user-friendly, a chatbot library could be developed that is integrated with a front-end realised with frameworks other than the Streamlit library, currently used for simple front-ends. Using a more advanced framework, it would be possible to create dynamic and responsive web applications, significantly improving the user experience. An improved front-end would allow greater customisation, allowing users to configure the interface according to their specific needs and ensuring better accessibility, making the system usable by a larger number of users.

By implementing these improvements, POC would not only further optimise contract management, but also offer more sophisticated and adaptable legal support, improving the efficiency and accuracy of lawyers' work.

7. BIBLIOGRAFIA

[1] **K. C. G. C. a. J. D. Tomas Mikolov, «Efficient Estimation of Word Representations in Vector Space.,» arXiv:1301.3781 [cs.CL], 2013.**

[2] **A. G. P. Andreas Zelios, «Recursive neural Networks: recent results and application,» 2022.**

[3] **S. H. a. J. Schmidhuber, «Long short-term memory. Neural computation,» 1997.**

[4] **B. V. M. C. G. D. B. F. B. H. S. K. Cho, «. Learning phrase representations using rnn encoder-decoder for statistical machine translation,» arXiv preprint arXiv:1406.1078, 2014.**

[5] **N. S. N. P. J. U. L. J. Ashish Vaswani, «Attention Is All,» CoRR abs/1706.03762 (2017). arXiv: 1706 . 03762., 2017.**

[6] **M.-W. C. K. L. K. T. Jacob Devlin, «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,» arXiv:1810.04805, 2018.**

[7] **J. W. R. C. D. L. D. A. I. S. Alec Radford, «Language Models are Unsupervised Multitask Learners,» 2019.**

[8] **OpenAI, «Language Models are Few-Shot Learners,» 2020.**

[9] **OpenAI, «GPT-4 Technical Report,» 2024.**

[10] **A. U. K. Q. M. S. A. U. N. A. B. M. Humza Naveed, «A Comprehensive Overview of Large Language Models,» arXiv:2307.06435v9, 2024.**

[11] **E. P. A. P. F. P. V. K. N. G. H. K. M. L. W.-t. Y. T. R. S. R. D. K. Patrick Lewis, «Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,» arXiv:2005.11401 , 2020.**

[12] **Redis, «Introduction to Retrieval Augmented Generation (RAG),» [Online]. Available: https://redis.io/glossary/retrieval-augmented-generation/.**

[13] **J. Coenradie, «Medium,» 2023. [Online]. Available: https://jettro.dev/question-answering-through-retrieval-augmented-generation-9b54806c214e.**

[14] **streamlit, «streamlit,» [Online]. Available: https://docs.streamlit.io/.**

[15] **P. cs, «Understanding Document Chunking in LangChain for Enhanced RAG Applications,» [Online]. Available: https://www.linkedin.com/pulse/understanding-document-chunking-langchain-enhanced-rag-praveen-cs-otvmc/.**

[16] **M. L. Michael Freenor. [Online]. Available: https://www.willowtreeapps.com/craft/openai-or-azure-openai-can-models-be-more-deterministic-depending-on-api.**

[17] **Redis, «Redis Documentation,» [Online]. Available: https://redis.io/docs/latest/develop/interact/search-and-query/advanced-concepts/vectors/.**

[18] **Microsoft, [Online]. Available: https://learn.microsoft.com/en-us/legal/cognitive-services/openai/data-privacy.**

[19] **Amazone, «AWS,» [Online]. Available: https://aws.amazon.com/it/what-is/prompt-engineering/?nc1=h_ls.**

[20] **Prompt Engineering Guide, «Prompt Engineering Guide,» [Online]. Available: https://www.promptingguide.ai/introduction/examples.**

[21] **W. Glantz, «Medium,» [Online]. Available: https://betterprogramming.pub/fine-tuning-gpt-3-5-rag-pipeline-with-gpt-4-training-data-49ac0c099919.**

[22] **M. F. Alam, «DataSciencedojo,» 2023. [Online]. Available: https://datasciencedojo.com/blog/llm-evaluation-with-langsmith/.**

[23] **K. C. G. C. J. D. T Mikolov, «Efficient estimation of word representations in vector space,» ICLR (Workshop Poster), 2013.**

ACKNOWLEDGEMENTS

I am deeply grateful to Daniele Sabetta for his continuous support and guidance. His involvement in various projects I supervised was crucial. I must also thank Prof. Morisio for his remarkable flexibility and availability. I extend my heartfelt appreciation to all the environment in Orbyta. This thesis is a testament to your support, guidance, and confidence in my abilities.