university of groningen

faculty of science and engineering

Information Extraction from Contracts Using Large Language Models

Cornelis Geerligs

university of groningen

faculty of science and engineering

University of Groningen

Information Extraction from Contracts Using Large Language Models

Master's Thesis

To fulfill the requirements for the degree of
Master of Science in Artificial Intelligence
at the University of Groningen under the supervision of
Dr. M.A. Dhali (Artificial Intelligence, University of Groningen)
and
Dr. Laura Astola (Accenture (External))

Cornelis Geerligs (s4978889)

September 6, 2024

Contents

Acknowledgments

This thesis stands as a special milestone at the end of my educational journey, a time not without challenges. However, I would not have been able to do it without the guidance and support of these people:

I owe a special debt of gratitude to Maruf A Dhali for supervising my project, thanks for the amazing feedback and for helping me navigate through the project. For your patience in understanding what was going on and not too hastily make decisions.

I also want to thank Laura Astola, despite our differences, her critical insights did push me to refine my thesis to the best of my ability.

Sandra van Grond, thanks for the continual support and understanding, Accenture is a really big company with lots of rules and departments and whenever I had a question I could always ask you.

A special thank you to Maria Schaars, who provided access to contractual data and assisted me in annotating without any expectation of reward. Thank you for the time you spared even though you were busy, I really appreciate it.

To my fellow interns at Accenture, sharing and discussing our projects over lunch has been both useful and enjoyable.

To my family for always supporting and believing in me. Listening to my AI rants and encouraging me nonetheless. Supporting me throughout these past 7 years of study. I am very grateful for the opportunity that they have given me.

Lastly, to my father, Henk Geerligs, for the countless discussions and brainstorming sessions about the project's direction and details. You have been an amazing help all throughout my life, thanks for always being there for me.

To all of you many thanks. It has been a really rewarding experience, and I have learned greatly from each of you.

I hope you enjoy reading the thesis!

Responsible Use of Large Language Models Statement

Ironically, in this thesis on the capabilities and limitations of large language models (LLMs), OpenAI's ChatGPT-4 has been utilized as a supportive resource throughout the writing and coding process. This integration was carefully managed with a strong commitment to ethical standards and best practices, ensuring the integrity and originality of the academic content.

Large Language Models (LLMs), are adept at generating detailed and coherent content. It is essential, however, to clearly differentiate between machine-generated outputs and the original intellectual contributions of the researcher. Any content derived from the LLM in this thesis was substantially modified and used primarily as preliminary drafts or supplementary information. These were then critically evaluated, reworded, or restructured to align with the broader, original analysis of the work. To ensure the thesis remains accurate and credible, information provided by ChatGPT was rigorously checked against reliable sources.

The deployment of the LLM adhered to ethical guidelines recommended by academic institutions and research bodies, serving solely to enhance productivity and understanding—not as a replacement for the student's intellectual engagement. All uses conform to ethical norms in academic research, focusing on the avoidance of plagiarism and upholding the authenticity of the scholarly work.

In coding aspects of the project, special attention was given to data privacy and security. All datasets were handled in accordance with relevant data protection laws, ensuring that no sensitive or personal data was processed by the LLM.

Abstract

Abstract

This thesis explores the usability of Large Language Models (LLMs) to automate intricate contract reporting tasks, focusing on filling in the Digital Operational Resilience Act (DORA) compliance template and extracting obligations from contracts. Structured experiments evaluate the capability of LLMs to identify and extract relevant information from legal texts and format this information to meet specific compliance and reporting standards. The first experiment achieved an average accuracy of 97.71% in filling out the DORA compliance template. The second experiment, involving the extraction of contractual obligations, showed a variable performance with the highest accuracy at 70.56%. The findings demonstrate the potential of advanced language models to reshape legal document analysis and management, reducing human error and labor costs in legal processes. The research also identifies the strengths and limitations of LLMs in extracting precise and contextually relevant information, suggesting areas for further development and optimization to enhance their applicability in the legal domain.

1 Introduction

The field of natural language processing has experienced a proverbial boom with the emergence of large language models (LLMs), particularly exemplified by the release of OpenAI's GPT-3 in 2020. With 175 billion parameters, GPT-3 marked a substantial leap over its predecessor, offering unprecedented capabilities across a variety of tasks such as reading comprehension, question answering, and code generation. This model set the stage for subsequent models that not only grew in size but also showcased emergent behaviors, enhancing their ability to perform complex tasks with minimal human oversight [12, 28].

After GPT-3, the field of large language models (LLMs) saw rapid advancements with models like Gopher, PaLM, and Megatron, each introducing key innovations in architecture and training methods [53, 16, 60]. Gopher, by DeepMind, uses sparse factorizations and a mixture of expert layers to boost model efficiency. Google's PaLM increases model size and enhances task-agnostic capabilities, enabling better performance in complex reasoning and code generation. NVIDIA's Megatron improves scalability through optimized data and model parallelism, handling larger datasets and pushing parameter counts into the trillions. These developments have expanded LLMs' utility, allowing them to effectively manage complex tasks requiring deep contextual and nuanced understanding.

The rapid adoption of these technologies is evident from the growth metrics of AI applications post-GPT-3. For example, ChatGPT, launched in November 2022, reached one million users within five days and achieved a user base of 100 million by February 2023, reflecting its broad acceptance and the potential for substantial economic impact. The model's ability to adapt to various user requirements through few-shot learning and its generalization capabilities have lowered barriers to entry across numerous applications, making advanced AI more accessible. As a result, models like GPT-4 and Llama-3 have not only advanced the frontiers of natural language understanding and generation but have also paved the way for their deployment in sectors requiring stringent compliance with regulatory standards.

The finance sector has recently undergone a digital transformation, increasing both the efficiency and complexity of its operations. This transformation has been followed by the European Union's proactive stance in implementing regulations to ensure the financial sector's operational resilience in the face of growing digitalization. A key initiative in this regulatory framework is the Digital Operational Resilience Act (DORA), which mandates extensive disclosures from Financial Entities (FEs) regarding their third-party Information and Communication Technology (ICT) service providers [23]. These regulations aim to fortify the financial sector's defenses against digital threats, underscoring the EU's commitment to fostering a secure and resilient financial ecosystem.

This research started with the aim of harnessing the capabilities of Large Language Models (LLMs) to enhance DORA compliance through the automation of mandatory reporting requirements. The objective was to streamline the compliance process, minimize errors, and reduce the reporting burden on FEs. Despite the challenge of data scarcity for testing and evaluation, a proper effort was made to implement the DORA template filler automation program.

Expanding on the groundwork established by the initial DORA compliance project and seeing the potential, this research also extended into another avenue: automating the extraction of obligations in contracts.

This second experiment builds upon the initial DORA compliance project, which, while insightful, posed a relatively straightforward challenge. By launching a more nuanced study on the extraction of obligations from contracts, this work aims to more thoroughly evaluate the potential of Large Language Models (LLMs) for ambiguous information extraction tasks. This additional project not only addresses the limitations observed in the DORA compliance initiative but also expands the investigation into the practical applications of LLMs within the financial sector. By adopting this dual approach, the research offers an exploration of how LLMs can change the efficiency and accuracy of information extraction processes in legal and financial contexts.

1.1 Research Question

The onset of Large Language Models (LLMs) has revolutionized numerous applications within Natural Language Processing, particularly in tasks that involve understanding and generating human-like text. This research aims to discover the capabilities of LLMs to enhance the process of information extraction from contracts. Thus, the research question that guides this thesis is:

"How can Large Language Models (LLMs) be effectively applied to automate the extraction of information from contracts?"

This primary question gives rise to several related sub-questions, each aiming to explore a different facet of the main inquiry:

1. **What are the capabilities of LLMs in identifying and extracting specific contractual information?**
2. **What are the limitations of LLMs in identifying and extracting specific contractual information?**

Addressing these sub-questions will help us understand how LLMs can change legal document analysis and what factors contribute to their success or limitations in this field.

2 Background Literature

2.1 Evolution of Language Models in NLP

Language modeling is a well-established area within the field of Natural Language Processing (NLP). The concept of developing a probabilistic model to analyze sequences of words was first proposed by Shannon in 1949 [58] with his n-gram models. These models were eventually trained on corpora as large as 2 trillion tokens [11]. The past decade has seen a surge in language model research, driven by advances in machine learning techniques, greater availability of data, and increased computational power. Initial efforts in autoregressive language modeling, such as those by Mikolov et al. (2010)[45] and Sutskever et al. (2011)[62], using recurrent neural networks. However, these models were relatively modest in size and trained on limited datasets.
As computational capabilities expanded, so did the models. Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks emerged, enabling models to remember long input sequences, thus improving the handling of context and reducing the impact of the vanishing gradient problem [32].
These improvements reached a milestone with the introduction of the transformer architecture by Vaswani et al. (2017)[63]. This model abandoned recurrent layers in favor of self-attention mechanisms, which directly compute relationships between all words in a sentence, irrespective of their positional distance. By doing so, transformers handle long-range dependencies more effectively and scale more efficiently with larger datasets and more complex models [63]. This architecture paved the way for developing models that could learn from vast amounts of data, capturing nuanced patterns and structures of language not previously possible.

2.2 Generative Pre-trained Transformers (GPT)

Introduced by OpenAI, Generative Pre-trained Transformers (GPTs) represent an evolution in the field of NLP. These models use the transformer architecture for effective learning and generation of text, demonstrating remarkable proficiency across various language tasks. GPT models are pre-trained on a diverse range of internet text and then fine-tuned for specific tasks, allowing them to achieve unparalleled success in fields ranging from conversational agents to complex problem-solving scenarios [51].
The first iteration, GPT-1, was succeeded by GPT-2, which featured a much larger model with 1.5 billion parameters trained on an even more extensive dataset. GPT-2 demonstrated the power of scale in language models, performing well across many NLP tasks without task-specific training [52]. However, GPT-3, with its 175 billion parameters, genuinely showcased the capabilities of large-scale language models. GPT-3 excelled at tasks it was directly trained on and showed an ability to perform tasks it had never seen before, purely by tapping into its vast pre-trained knowledge base [12].
This advancement underscores a shift in AI research—towards models that can generalize from broad datasets to specific tasks, reducing the need for large annotated datasets in new applications. The success of GPT models has spurred ongoing research and development, leading to further refinements in model architecture, training methods, and application to real-world tasks [24].

2.3 Domain-Specific Adaptations of LLMs

The refinement of large language models (LLMs) to meet the specific needs of various domains has been a pivotal development in the field of NLP. Bidirectional encoder representations from trans-

formers (BERT) models have predominantly been used for these domain-specific applications such as BioBERT [39], SciBERT Beltagy, Lo, and Cohan [6], FinBERT [1] and BloombergGPT [66] serve as exemplars of this trend. BioBERT, which is pre-trained on extensive biomedical literature, excels in biomedical entity recognition tasks, significantly outperforming its predecessors by using domain-specific vocabulary and context [39]. Similarly, SciBERT, developed by Beltagy et al. (2019)[6], adapts the BERT architecture to the scientific domain by training on a corpus comprising scientific papers, making it highly effective for tasks like citation intent classification and relation extraction. FinBERT, specifically tuned for the financial sector, demonstrates its prowess in analyzing financial texts for sentiment analysis, which is applicable for market prediction and risk assessment [1].
Moreover, the field has seen innovation through models like ClinicalBERT and BioMedRoBERTa, which are fine-tuned for even more specialized sub-domains such as clinical reports and biomedical research, respectively [34, 29]. These models address even more focused tasks that require understanding complex interactions within data, such as patient outcomes prediction and treatment effect analysis.
The adaptation strategies also vary. While some models are pre-trained from scratch on specialized corpora, others are fine-tuned from general-purpose models, a method that has proven effective in maintaining the robust features of foundational models while adapting to the intricacies of domain-specific data [21, 29].
This specialization is not just about enhancing performance but also about integrating these models into operational pipelines where they can provide actionable insights specific to the needs of the industry, such as predicting market trends or understanding complex medical reports [67, 43]. The success of these domain-specific models highlights the flexibility and scalability of transformer architectures, encouraging further adaptations across various fields.

2.4 LLMs in Legal Applications

The legal domain has seen an increase in the applications of large language models (LLMs), illustrating the adaptability of these technologies to sector-specific needs. Notable examples include Legal-BERT[13] and CaseLaw-BERT[69], which have been fine-tuned from general models to address tasks such as legal judgment prediction, contract review, and other areas requiring a nuanced understanding of legal language and concepts. Legal-BERT, for instance, is trained on a diverse corpus of legal documents to tailor its capabilities to legal reasoning and statutory interpretation, showing marked improvements over its generalist predecessors in tasks such as named entity recognition and legal topic classification[13].
Similarly, CaseLaw-BERT adapts the BERT architecture to analyze and predict outcomes based on the historical case law data, proving essential for automating aspects of legal research and precedent analysis[69]. These models not only demonstrate the potential for LLMs to enhance efficiency in legal workflows but also highlight the importance of domain-specific training in achieving high accuracy and relevance in professional settings.
The development of such models addresses specific challenges in the legal field, including the need to parse and understand complex legal jargon and to maintain the integrity and interpretability of legal information. The evolution of these models has also sparked discussions around ethical considerations, such as ensuring fairness, preventing bias, and maintaining transparency in automated legal decision-making processes[9].

3 Methodology

3.1 Introduction to Large Language Models (LLMs)

Natural Language Processing (NLP) has witnessed a lot of progress during the past few decades, evolving from simple rule-based algorithms to sophisticated machine-learning models capable of understanding, generating, and interpreting human language with astounding accuracy. The advent of Large Language Models (LLMs) marks a big milestone in this journey, offering unprecedented capabilities in a wide range of text-based tasks[12].

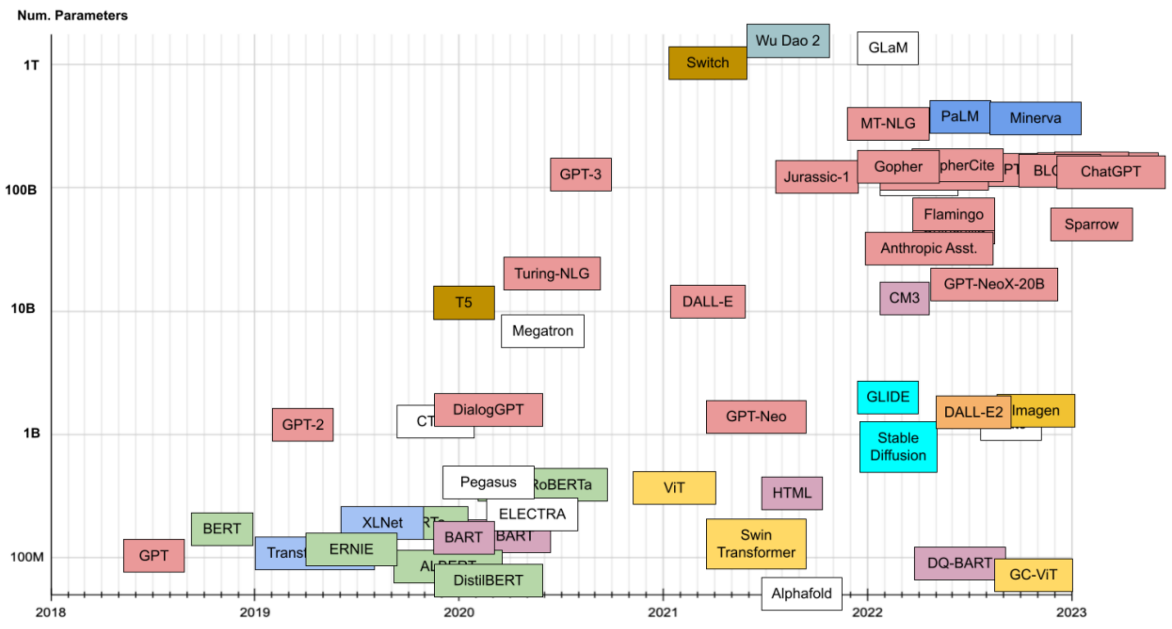


Figure 1: LLM progression in recent years[49].

Figure 1 illustrates the rapid progression of Large Language Models over recent years, highlighting key milestones in their development. From early models with relatively limited capabilities to the latest iterations that boast billions of parameters and unprecedented linguistic comprehension, the timeline underscores the exponential growth and refinement of these technologies. This evolution reflects not only advances in computational power and algorithmic sophistication but also a growing comprehension of language's complexity and nuances.

3.2 Foundations of Large Language Models

This section will delve into the fundamental components and concepts underlying LLMs, providing a solid foundation for understanding how these models are constructed and how they operate.

3.2.1 Neural Networks

Neural Networks are foundational to the operation of modern machine learning systems, particularly in the context of deep learning, which powers most state-of-the-art natural language processing models today [26]. A neural network consists of interconnected nodes or neurons, which are organized into layers. The simplest form of a neural network includes three types of layers: the input layer, hidden layers, and the output layer 2.

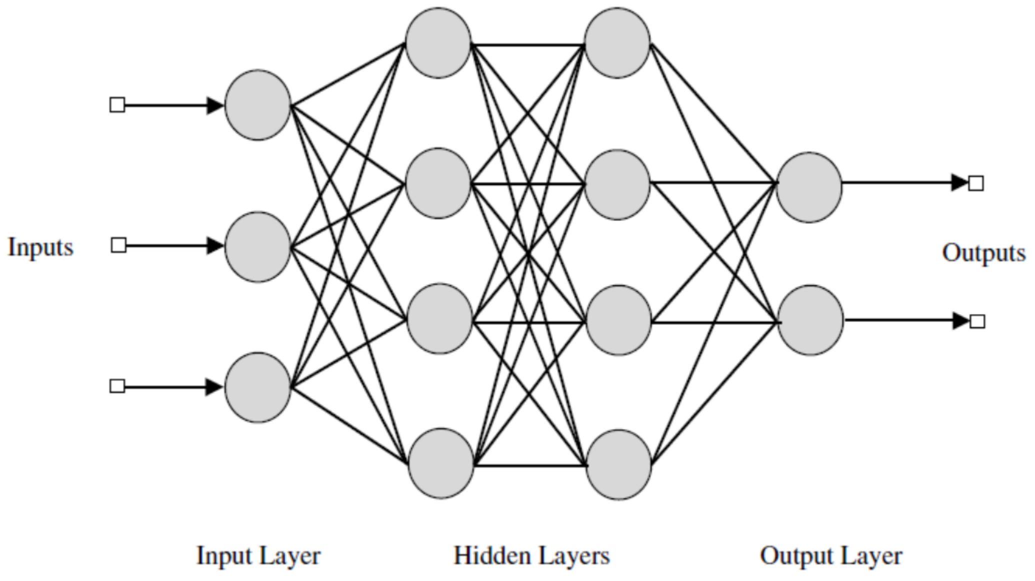


Figure 2: Two hidden layers fully connected Neural Network structure.

Each neuron in a layer receives input from the neurons of the previous layer, processes this input through a non-linear function, and passes the output to the next layer. The strength of the connection between neurons, known as weights, is adjusted during the training process to minimize the error in prediction [38]. This training involves the backpropagation algorithm [54], which efficiently computes gradients of the loss function with respect to the weights.
Deep neural networks, which contain multiple hidden layers, can model complex relationships in the data. The depth of these networks is a key factor in their ability to perform feature extraction at various levels of abstraction, making them highly effective for tasks such as image recognition, speech recognition, and natural language processing [56].

3.2.2 Transformer Architecture a quick overview

The transformer model, introduced by Vaswani et al. (2017)[63] in their groundbreaking paper *"Attention Is All You Need"*, has completely changed the field of natural language processing by providing a new architecture for building more efficient and powerful LLMs. Unlike traditional sequence-to-sequence models that rely on recurrent or convolutional layers to process data, the transformer adopts a fully attention-based mechanism.

The core idea behind the transformer architecture is to use self-attention mechanisms to weigh the significance of different parts of the input data differently. This approach allows the model to focus on relevant parts of the input sequence when performing tasks, leading to improvements in processing speed and model performance, especially on tasks involving long sequences.

As illustrated in Figure 3, all transformers share three fundamental components:

- **Tokenizers** that transform text into discrete tokens.
- **An embedding layer** that maps tokens and their positions to vector representations.
- **Transformer layers** that perform successive transformations on the vectors, progressively extracting linguistic information. These layers alternate between attention mechanisms and feed-forward neural networks.

Figure 3 shows the encoder-decoder structure of the transformer. The process begins with tokenizing the input text into discrete tokens. The tokenizer identifies the boundaries of each token, which often consist of subword units like ”de” or ”to”. By dissecting the text into its most meaningful tokens, the tokenizer efficiently compresses the information. Each token is then transformed into a numerical vector through an embedding process, where it is matched with a pre-trained vector from an embedding table. To capture the sequence’s order, positional encodings are added to these embeddings, compensating for the model’s inherent lack of sequential processing capability.

The encoder consists of multiple identical layers, each featuring a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. The self-attention mechanism allows the model to weigh the importance of different words in the sentence, regardless of their positional distance, by calculating attention scores based on queries, keys, and values derived from the embeddings. This enables the encoder to capture a rich context of the input sequence. The feed-forward networks further process these representations independently for each position, adding depth and complexity to the model’s understanding.

The decoder mirrors the encoder structure but includes an additional cross-attention layer in each of its layers, which attends to the encoder’s output. It also features a masked self-attention mechanism that prevents positions from attending to subsequent positions in the sequence, preserving the auto-regressive property necessary for generating coherent text outputs. The cross-attention layers allow the decoder to focus on relevant parts of the input sequence, which is essential for tasks that require contextual awareness.

The final output of the decoder is produced by transforming the attended features into output probabilities through a linear layer followed by a softmax function. This predicts the probability distribution of the next token in the sequence, effectively generating text one token at a time.

This architecture makes use of the parallel processing of sequences, accelerating training compared to traditional sequential models like RNNs and LSTMs. The Transformer’s ability to process input data in parallel and its use of attention to dynamically focus on different parts of the input sequence allows it to manage long-range dependencies in the text more effectively.

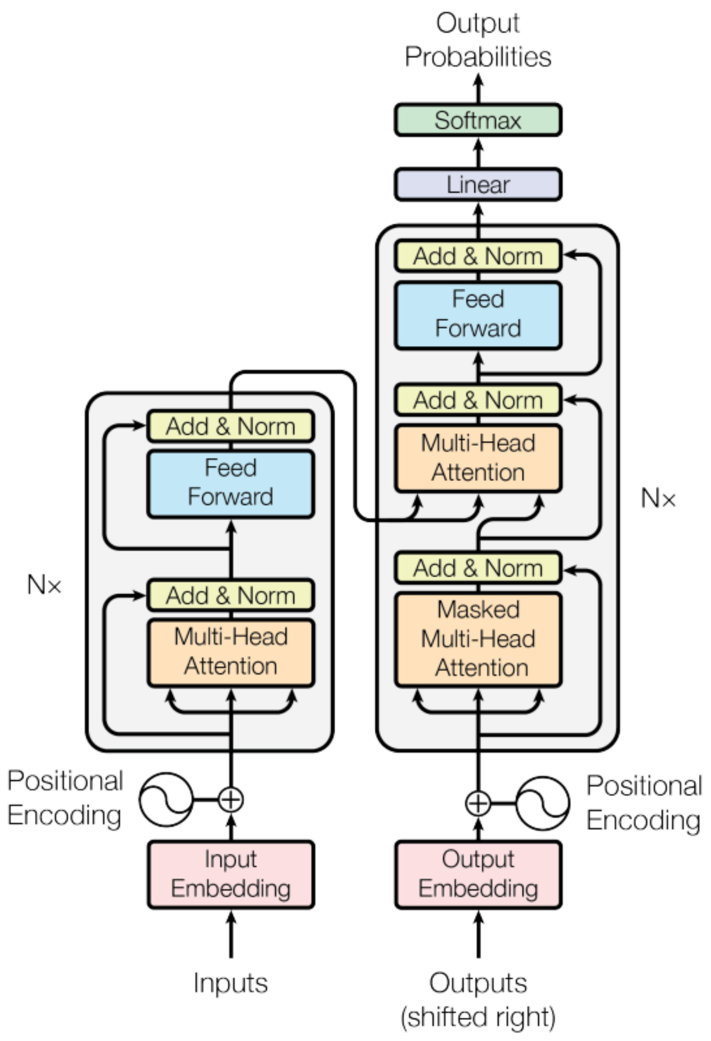


Figure 3: Schematic representation of the transformer architecture, illustrating its encoder-decoder(left and right) structure and the flow of data through various self-attention and feed-forward layers [63].

3.2.3 Input

The input text undergoes tokenization, typically using a byte pair encoding (BPE) tokenizer, which divides the text into tokens. Each token is then mapped to a vector by referencing a word embedding table. Subsequently, positional encodings are integrated with these word embeddings to incorporate the order of the tokens. The LLM now holds not only the unique information for each token in the text but also the positional data that indicates its location relative to other tokens. This information includes the specific details stored in the embedding table for each token. For example, if the token is 'Th', the embedding table might indicate that the token 'e+space' frequently follows it.

3.2.4 Encoder Decoder

In the original paper by Vaswani et al. (2017)[63], the transformer model was introduced with both an encoder and a decoder components. This dual structure was essential for the model's original purpose—translating text from one language to another. The encoder processes the input text, preparing it for translation, while the decoder then generates the corresponding text in the target language. In contrast, contemporary generative models, such as GPT, typically utilize only a decoder, focusing on generating text based on given prompts. Similarly, domain-specific models like BERT employ solely an encoder, optimizing them for tasks that involve understanding input text without the need to generate new text.

The encoder processes the input sequence to create a set of context vectors. Each element of the input sequence, such as a word in a sentence, is converted into a numerical value, often using embeddings. These numerical representations are then fed through a series of neural network layers that reduce their dimensionality while preserving essential information about their relationships within the sequence. This process results in an ”encoded” version of the input, typically represented by the final hidden state of a neural network, which captures the summary of the entire input sequence [15].

Following the encoder, the decoder’s role is to take the encoded data and reconstruct an output sequence from it, effectively translating the encoded information back into a variable-length sequence [15]. The encoder and decoder are typically trained jointly to optimize the conditional probability of the output sequence given the input sequence, thereby maximizing the relevance and accuracy of the output with respect to the input.

Each encoder and/or decoder block contains multi-head attention and feedforward neural network layers, referred to collectively as a multi-head block 3. These blocks can focus on specific aspects of grammar or logic. Stacking multiple such blocks enhances the model’s representational power, allowing it to learn complex patterns and relationships within the data.

3.2.5 Attention mechanism

In the architecture of transformers, the attention mechanism is central to modeling the dependencies between inputs and outputs, independent of their positions in the input sequence. This mechanism is operationalized through the creation of queries, keys, and values, which are vectors representing the input tokens in a sequence. Each vector is generated through learned linear transformations applied to the input embeddings, as described in Vaswani et al. (2017)[63].

Queries, keys, and values are foundational elements in computing attention weights within the transformer model. For each token in the input sequence, the model learns three distinct sets of weight matrices: the query weights W_Q, the key weights W_K, and the value weights W_V. The input token representation x_i is transformed by each of these weight matrices to produce a query vector $q_i = x_i W_Q$, a key vector $k_i = x_i W_K$, and a value vector $v_i = x_i W_V$.

The interaction between these components is akin to a retrieval system where the query vector of a token seeks to retrieve information from all other tokens. This retrieval is quantified by the dot product between the query vector corresponding to one token and the key vector of another, creating a score that represents how much focus (or attention) the token should place on other parts of the input sequence. These scores are scaled by the square root of the dimension of the key vectors ($\sqrt{d_k}$) to stabilize the gradients during training. The scores are then normalized across all tokens using a softmax function to form the final attention weights for each token, given by the formula:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

where Q, K, and V represent the matrices of queries, keys, and values, respectively.

This attention mechanism is inherently different from prior models due to its self-attention capability. It allows each token to attend to all other tokens in the sequence, thereby drawing global dependencies between input and output. The multi-head attention architecture further enhances this by allowing the

model to attend to information from different representation subspaces at different positions, effectively capturing a variety of relationships in a single processing step.
The queries, keys, and values serve distinct but interrelated roles:

- **Query:** Represents the current token for which the context is being sought; this can be thought of as asking a question about something to the other tokens.
- **Key:** Encapsulates all tokens in the sequence as potential candidates to be retrieved; this can be thought of as the answer that all the tokens will give to the query. If the query and key are a match, they will have a high dot product, and the query token will receive a lot of information from this key token.
- **Value:** Holds the actual content of the tokens, if the query and key are a match this is the value that the key-token will give to the query-token.

Through these interactions, the transformer can dynamically focus on the most relevant parts of the input data, facilitating effective translation, text generation, and many other tasks where understanding the context and relationships within the data is very important. This method of attention, is not only efficient computationally, allowing parallel processing, but also highly flexible, adapting to the needs of various sequence modeling tasks.

3.2.6 Multi-Head Attention

In the architecture of transformers, one of the most critical enhancements introduced is the multi-head attention mechanism. This feature allows the model to process the input data through multiple distinct attention mechanisms—or ”heads”—in parallel, each with its own set of learned projections for queries, keys, and values.

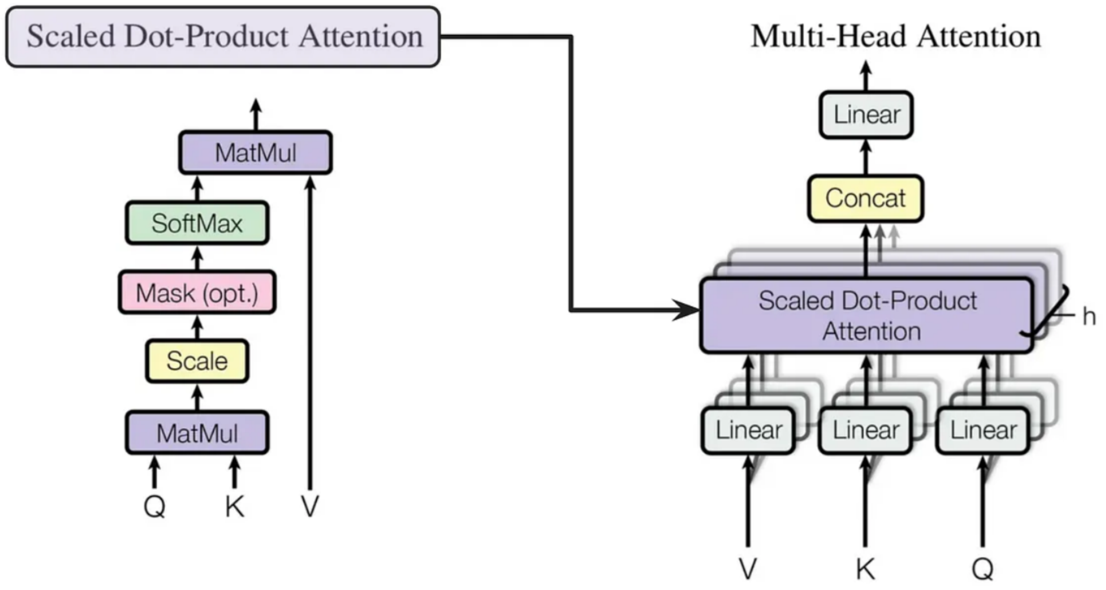


Figure 4: Single head attention(left) and Multi-Head Attention (right)[63].

Multi-head attention improves upon the single attention function by allowing the model to jointly attend to information from different representational subspaces at different positions. This is achieved by projecting the queries, keys, and values multiple times with different, learned linear projections to lower dimensions [63]. Each projection corresponds to a different ”head”, and the attention function is applied in parallel across these projections:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2}$$

where W_i^Q, W_i^K, and W_i^V are the parameter matrices for the i-th head for queries, keys, and values, respectively. This parallel processing allows each head to capture different aspects of the information contained in the input sequence.

After computing the output for each head, the results are concatenated and then projected again to form the final output of the multi-head attention layer:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \tag{3}$$

where W^O is a parameter matrix that linearly transforms the concatenated output back to the original dimensionality of the model. This final projection helps to combine the different informative signals captured by each head [63].

The advantage of using multiple heads is that the model can capture a wider range of information from the input sequence. For instance, different heads might focus on different types of syntactic structures or semantic nuances, providing a richer representation of the input data. This feature is particularly beneficial in complex tasks like machine translation or semantic role labeling, where different linguistic features can be important [63].

In practice, multi-head attention allows the transformer model to perform better on a variety of natural language processing tasks by enhancing its ability to understand and integrate information from different parts of the input data simultaneously. The parallel structure also facilitates efficient training and provides robustness to the representation power of the network[21, 12, 2].

3.2.7 Positional Encoding

Since transformers do not inherently capture the order of tokens in a sequence due to the parallel processing of tokens, positional encodings are essential to incorporate sequence order information into the model's computations. Vaswani et al. (2017)[63] introduced sinusoidal positional encodings as a method to add this information to the input embeddings . For a given position p and dimension i, the positional encoding is computed using sine and cosine functions, as shown in the following equations:

$$PE(p, 2i) = \sin\left(\frac{p}{10000^{\frac{2i}{d}}}\right) \tag{4}$$

$$PE(p, 2i+1) = \cos\left(\frac{p}{10000^{\frac{2i}{d}}}\right) \tag{5}$$

where d represents the dimensionality of the embeddings. These sinusoidal functions are designed to provide a unique encoding for each token position, enabling the model to distinguish the position of tokens within the sequence. The choice of sinusoidal functions also facilitates the model's ability to learn relative positions, as shifts in the position can be represented linearly within these encodings.

In contrast to fixed sinusoidal encodings, developments in transformer architectures include the use of learned positional encodings. These encodings are model parameters that are learned during the training process, similar to other neural network weights [21]. Learned positional encodings provide the flexibility to adapt to the specific characteristics of the training data, potentially enhancing model performance on specific tasks. However, they also introduce a limitation on the maximum sequence length that can be processed by the model due to their fixed dimensionality, unlike sinusoidal encodings, which can theoretically handle sequences of any length [63, 21].

3.3 Training and Optimization of LLMs

Transformers employ a training paradigm that typically begins with unsupervised pre-training, followed by supervised fine-tuning. The pre-training phase uses a larger dataset to develop a broad understanding of language, while fine-tuning adjusts the model's parameters for specific tasks using smaller, task-specific datasets.

3.3.1 Pre-training

Pre-training is the initial stage where the model learns general language representations. This stage involves training the model on a large corpus of text data, typically sourced from the internet, encompassing a wide range of topics and styles. The model learns to predict parts of the text, either through next-word prediction or by filling in missing words in a sentence.
The primary goal of pre-training is to enable the model to learn the general properties of a language, including syntax, semantics, grammar and other common patterns within vast amounts of text data. This foundational knowledge equips the model with the capability to perform well on a variety of downstream tasks, even before any task-specific fine-tuning [21, 51].
Pre-training typically employs unsupervised learning techniques, with two dominant methods used in the training of transformer-based models:

1. **Autoregressive Language Modeling (ALM):** In this method, the model predicts the next word in a sentence given all the previous words. It is trained to maximize the likelihood of the next word based on the preceding context. This method helps in learning a language model from scratch [52].

2. **Masked Language Modeling (MLM):** Popularized by BERT and adopted in various forms by other models, MLM involves masking a portion of the input tokens randomly and then predicting these masked tokens based on their context. This method allows the model to learn a bidirectional understanding of language context [21].

The data used in pre-training is of great influence for the model's performance. It generally includes a wide range of text sources from books, articles, websites, and other forms of written media to ensure diversity and comprehensiveness in language learning. The scale of the data can range from billions to trillions of words, depending on the model's capacity and the computational resources available [12].

3.3.2 Fine-tuning

Once a large language model (LLM) has been pre-trained, it can initially only autocomplete documents from a starting token, which isn't very useful on its own. To harness the full potential of these foundational LLMs, they must be further trained on specialized tasks. This additional training enables them to apply their general capabilities to specific applications. This process is essential for LLMs, where the shift from a generic training regime to task-specific optimizations will be important for most LLM use-cases

The fine-tuning dataset contains examples that are directly relevant to the tasks the model will perform, such as contract analysis, sentiment analysis, or legal document information extraction. The primary goal of fine-tuning is to adjust the model's weights such that it maintains its extensive language understanding while becoming highly efficient at processing and generating outputs that are task-specific [33].
For instance, when fine-tuning a model for contract reporting, the model should be trained on a dataset comprised of contracts, legal negotiations, and compliance documents. This specialized training helps the model grasp the nuanced language of legal documents and understand the structure and obligations typically outlined in contracts.
Supervised fine-tuning involves adjusting LLMs to perform well on a task by training them on a labeled dataset relevant to that task. This method fine-tunes the entire model or specific parts of it, depending on the nature of the task and the data. Most common techniques for supervised fine-tuning are:

- **Task-Specific Fine-tuning:** Focuses on optimizing model performance for a particular task, using a dataset closely related to the task's requirements. This method is highly effective but risks catastrophic forgetting if the model's parameters diverge too far from their initial settings.
- **Multi-Task Fine-tuning:** Involves training the model on multiple related tasks simultaneously. This approach not only improves the model's performance on these tasks but also helps in maintaining its generalization capabilities.
- **Sequential Fine-tuning:** Applies when a model is incrementally trained on related tasks in a sequence, which is useful for progressively adapting a model to specialized tasks within a broader domain.
- **Few-Shot Fine-tuning:** Utilizes a small number of examples to guide the model in adapting to new tasks. This technique is particularly valuable when only limited task-specific data is available.

Reinforcement learning from human feedback (RLHF) is a sophisticated fine-tuning approach where a model is refined using feedback directly from human evaluators. This method involves training a reward model based on human preferences, which then guides the fine-tuning of the LLM by reinforcing outputs that align with human judgments.

- **Reward Model Training:** Develops a model that assigns numerical scores to outputs based on their quality as judged by humans, facilitating the targeted adjustment of the LLM's parameters.
- **Proximal Policy Optimization (PPO):** Utilizes the reward model's scores to train the LLM using reinforcement learning techniques, optimizing the model's outputs towards those that are most favored by human evaluators.

Parameter-efficient fine-Tuning (PEFT) addresses the resource-intensive nature of traditional fine-tuning methods by modifying a smaller subset of the model's parameters. This technique allows for computational savings and reduces the risk of catastrophic forgetting.

- **Low-Rank Adaptation (LoRA):** A specific implementation of PEFT, LoRA modifies the model by applying low-rank updates to the parameters, which allows for efficient fine-tuning on standard hardware configurations.

Introduced as an alternative to RLHF, Direct Preference Optimization (DPO) simplifies the fine-tuning process by directly optimizing the model's outputs based on human preferences without the need for a separate reward model. This method enhances efficiency and reduces the complexity involved in model training.

- **Preference-based Training:** In DPO, the LLM is adjusted based on direct feedback, where outputs are labeled as positive or negative based on human preferences, steering the model towards more desired responses.

3.3.3 State-of-the-art models

Two of the most prominent sources for evaluating large language models based on performance and user preferences are the GPT Arena[14] and the Huggingface leaderboards[4, 25, 17, 68, 31, 42, 55, 18]. Each of these platforms offers a unique approach to benchmarking the capabilities and effectiveness of LLM's.

Rationale for Focusing on Pre-Trained Models
Evaluating language models requires distinguishing between different training and deployment strategies, such as pre-trained, continuously pre-trained, fine-tuned, and merged models. Each category reflects distinct stages of model development and impacts performance and applicability differently.
The focus on pre-trained models for baseline comparison is driven by the need to assess models in their most fundamental state. Pre-trained models, not yet fine-tuned or customized for specific tasks, provide a clear view of a model's core abilities without the influence of task-specific optimizations. This approach allows for fair comparisons across different models based on their inherent capabilities.
Focusing on pre-trained models highlights advancements in foundational model capabilities and serves as a starting point for further customization and fine-tuning. In contrast, fine-tuned and iteratively trained models, while potentially superior for specific tasks, may not generalize well across diverse applications and can introduce biases or overfitting. This focus also avoids the pitfalls of evaluating models that may have artificially inflated performance on benchmark tests that do not necessarily translate to general effectiveness in real-world scenarios.

Huggingface Leaderboards
The Huggingface leaderboards employ the "EleutherAI language model evaluation harness" to assess models across six different benchmarks. These benchmarks include tasks that range from commonsense reasoning to scientific knowledge and computational skills. The aim is to test models in both zero-shot and few-shot settings to gauge their general and task-specific knowledge capabilities. Notable benchmarks include the AI2 Reasoning Challenge, HellaSwag, and GSM8k. The leaderboards focus exclusively on open-source models to ensure transparency and encourage community engagement. An overview of the current rankings on Huggingface can be found **here**.
GPT Arena
The GPT Arena, as introduced in the work by Chiang et al (2024)[14], utilizes a crowdsourced approach to model evaluation. Here, models are evaluated through pairwise comparisons in which users vote for the model that provides the more satisfactory answer. This arena has proven to be a robust method for capturing human preferences across diverse queries, reflecting a broad spectrum of real-world applications. The GPT Arena has become one of the most referenced LLM leaderboards, noted for its transparency and straightforward evaluation metrics. A summary of the current standings in the GPT Arena is presented **here**.

Top Open-Source Models
Open-source models are the main contributors to democratizing advanced AI technologies and making them accessible to everyone, allowing researchers and developers to innovate without high costs. Based on the Huggingface leaderboards, here are the top five open-source models, known for their robust performance across various benchmarks:

- **meta-llama/Meta-Llama-3.1-405B** The best open source model to date, a very large model that can be used as an instructor for smaller models like 70b and 8b
- **mistralai/Mistral-Large-Instruct-2407**, a big 123 billion parameter model trained specifically for math, code and reasoning. Outperforming Llama 3.1 405 in those areas.

These models are typically hosted on platforms like Huggingface, where they can be accessed and used at no cost, although deploying them at scale involves infrastructure expenses.

Top Closed-Source Models
Closed-source models often lead in performance but come with usage restrictions and cost implications. From the GPT Arena, the leading closed-source models include:

- **GPT-4o** from OpenAI is distinguished by its top-tier performance across various language understanding and generation benchmarks. This model is primarily deployed in enterprise environments that demand high-quality and consistent outputs. It delivers the best overall benchmark results among competing models, although it is the most expensive option.
- **Claude-3.5-Sonnet** excels in graduate-level problem-solving (GPQA), undergraduate-level knowledge (MMLU), and coding proficiency (HumanEval). It is recognized as the foremost model for advanced reasoning tasks.
- **Gemini 1.5 Pro** introduces a new context window that extends up to two million tokens—the most extensive among large-scale foundational models. This feature enables recall in long-context retrieval challenges, making it ideal for processing extensive documents, lengthy code, and extended periods of audio and video content.

These models are typically accessed through an API, with associated costs varying according to usage. For the initial experiment in this study, OpenAI's GPT-4o was chosen. As a proprietary model, GPT-4o is available exclusively via API, with charges based on the amount of tokens processed.
At the time of conducting the experiment, LLama 3.1 and Mistral Large had not yet been released, thus no open-source models were available that could handle a context window larger than 8,000 tokens. Consequently, this limitation prevented the inclusion of larger documents such as contracts in our testing, restricting our evaluation to available models.
To ensure that our findings were applicable to large language models (LLMs) in general and not restricted to a specific model, the second experiment incorporated the top three models available at that time: GPT-4o, Claude-3.5-Sonnet, and Gemini 1.5 Pro. All these models are proprietary and were accessed via API, allowing us to make requests and incur costs based on the number of tokens processed.

3.3.4 Limitations of Large Language Models

Inherent Biases in Model Outputs
LLMs are often criticized for their tendency to perpetuate and sometimes amplify existing biases found in their training data. These biases can manifest in various forms, such as racial, gender, and socio-economic prejudices, which are embedded in the vast corpora of internet text that LLMs are typically trained on[7]. This not only affects the fairness of the models but also raises concerns about their trustworthiness and the ethical implications of their deployment in real-world applications, such as recruitment, law enforcement, and loan approvals [50]. Strategies to mitigate these biases are still in development, with current approaches including diversification of training data, debiasing algorithms, and increased transparency in model workings [8].

Environmental Impact
The carbon footprint of training LLMs is substantial. These models require vast amounts of computational power, which translates to astounding energy usage. Studies have highlighted the environmental impact of training models like GPT-3, which involves substantial CO2 emissions equivalent to the lifetime emissions of several cars [61][20]. The sustainability of such practices has been questioned, with calls for more energy-efficient computing techniques and the adoption of green energy sources in AI research and operations [57].
In their paper, the developers of Meta's open-source LLama3 model disclosed details about its power consumption. While they utilized entirely green electricity, this approach addresses only the symptom rather than the core issue of high energy usage. Moreover, other LLM creators may not adhere to similar green practices[65]. . The emissions of 6 different models is shown in figure 5

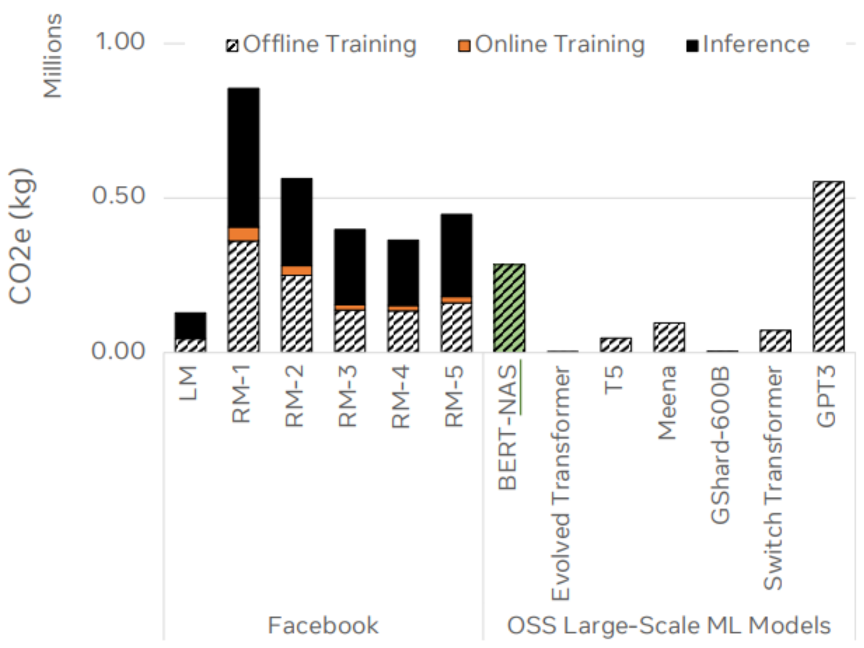


Figure 5: Operational carbon emissions associated with training and inference for six representative machine learning models in production at Facebook, across various ML models[65].

High Economic Costs of Development
The development of LLMs involves large economic investment, largely due to the costs associated

with computational resources. The training process requires state-of-the-art hardware, such as advanced GPUs, which are not only expensive to purchase but also costly to operate due to their high energy consumption. This economic barrier limits the ability of smaller entities or institutions, particularly those in developing countries, to participate in cutting-edge AI research and development. Moreover, the ongoing costs of model tuning and updating to maintain state-of-the-art performance further increase the total cost of ownership[36].

Scalability and Accessibility Challenges
While the capabilities of LLMs scale with size, this scalability brings about several challenges. Larger models require not only more computational resources but also generate higher latency in applications, which can be a critical drawback in real-time systems [7]. For instance, in real-time language translation services or interactive chatbots, a delay of even a few seconds can degrade user experience significantly. Furthermore, deploying such models in resource-constrained environments like mobile devices or in regions with limited computing infrastructure can be impractical. This often necessitates additional investments in specialized hardware or cloud services to handle the increased load. This scalability issue affects the accessibility of advanced LLM technologies, creating a divide between organizations that can afford to invest in such technologies and those that cannot. Furthermore, the complexity of deploying these models in practical applications requires engineering efforts to optimize efficiency and manage resource demands effectively.

Technical Limitations and Societal Concerns
Beyond the immediate limitations related to biases, costs, and environmental impacts, LLMs face technical challenges in understanding abstract concepts and executing complex reasoning tasks. While adept at recognizing patterns, these models frequently struggle with capturing the deeper semantic meanings that come naturally to humans [44]. A survey study further proves this by examining the behavior of LLMs across core reasoning tasks such as logical, mathematical, and causal reasoning [48]. The survey reveals that although LLMs perform well in scenarios closely aligned with their training data, they encounter considerable difficulties in out-of-distribution situations, highlighting a limitation in their reasoning abilities. These findings underscore the need for moving beyond shallow accuracy metrics and advocate for more sophisticated reasoning assessments to better evaluate the capabilities of LLMs [48].
Additionally, the societal implications of deploying LLMs in sensitive contexts—such as surveillance, military applications, and the spread of misinformation—are profound, raising concerns about privacy, security, and the potential for misuse [70]. These concerns emphasize the necessity for rigorous ethical reviews and the implementation of robust safeguards before LLMs are deployed in high-stakes environments.

3.3.5 Impact Analysis

This section discusses the potential impacts of implementing automated extraction with LLMs, focusing on operational changes within organizations, potential job role modifications, and how such technologies could shift workflow dynamics.

Operational Efficiency Gains Integrating LLMs into organizational workflows for automated information extraction offers substantial improvements in the speed and accuracy of data processing tasks. Traditionally, legal teams spend considerable amounts of time manually reviewing contracts to identify and document obligations, which are then entered into scheduling applications to ensure compliance and timely fulfillment. This manual process often requires dedicated teams of lawyers who focus exclusively on tracking these obligations. While the precise amount of time saved by implementing LLMs has not yet been quantified, but the potential for efficiency gains is evident. Automating these tasks with advanced language models not only reduces the workload on legal staff but also enhances the reliability of the data captured, leading to better compliance management and operational performance.

Workforce adaptation The automation of information extraction tasks is likely to lead to a redefinition in workforce roles and responsibilities. As noted by [40], automation does not necessarily eliminate jobs but rather evolves them, requiring new skills and competencies. Organizations must, therefore, consider strategies for workforce re-skilling and training to manage this transition effectively. Supporting employees through these changes is essential for gaining the benefits of automation while maintaining job satisfaction and organizational loyalty [59].

Technology Adoption Barriers While the benefits of adopting LLMs for automation are clear, several barriers can impede their integration into existing systems. These include technological limitations, cultural resistance to change, and the initial costs associated with deployment and training. Dasgupta et al. (2019)[19] discuss various strategies to overcome these barriers, such as phased implementation plans, stakeholder engagement, and emphasizing the long-term benefits of adoption. Understanding these potential obstacles and planning for them can facilitate smoother transitions and more successful integrations of new technologies [46].

3.3.6 Ethical Considerations

Ethical considerations play a pivotal role in the development and deployment of generative AI and large language models (LLMs). As these technologies become more integrated into society, their potential to influence public opinion, decision-making processes, and privacy norms grows, making ethical considerations paramount. Ethical AI development focuses on principles such as fairness, accountability, transparency, and the avoidance of bias, which are essential to ensure that AI technologies contribute positively to society and do not perpetuate existing inequalities or injustices[47, 22].

Fairness in AI is particularly challenging due to the diverse and often biased nature of the data used to train these models. Bias in training data can lead to outcomes that unfairly discriminate against certain groups, undermining the social acceptability of AI technologies[3]. For example, research has shown that LLMs like GPT-3 can exhibit racial and gender biases in language generation, which could perpetuate stereotypes and unfair discrimination against certain groups[7]. To combat this, developers must employ strategies such as inclusive data collection, bias mitigation algorithms, and continuous monitoring of AI systems to ensure that they operate fairly[5].

Accountability in AI involves establishing clear guidelines on who is responsible for the outcomes produced by AI systems. This is complicated by the autonomous nature of these systems, which can make tracing decisions back to human operators difficult[37]. Implementing mechanisms such as audit trails, decision logs, and transparency reports can help increase accountability. Additionally, developing standards for explainable AI can make these systems more understandable to users and regulators, thereby enhancing trust[27].

Privacy concerns are also paramount as AI systems often process vast amounts of personal data to function effectively. Protecting this data and ensuring that AI systems comply with data protection regulations like the General Data Protection Regulation (GDPR) is foundational for maintaining user trust and legal compliance[64].

The ethical deployment of AI also involves engagement with diverse stakeholders, including policymakers, technologists, and the public, to ensure that AI systems are developed and used in a manner that reflects broad societal values and norms[30]. This multi-stakeholder approach helps to balance the benefits of AI with potential ethical risks, promoting an AI ecosystem that is both innovative and responsible.

The ethical considerations surrounding AI and LLMs are complex and multifaceted. Addressing these concerns requires a concerted effort from developers, regulators, and society at large to ensure that AI technologies are used in ways that respect human rights and promote social good. As AI becomes more prevalent, the importance of robust ethical frameworks and proactive regulatory approaches will only increase, highlighting the need for ongoing vigilance and adaptation in AI governance[35].

4 Experimental Setup

This section outlines the experimental setup for evaluating the performance of the developed information extraction programs. Two separate experiments were conducted: Experiment 1 focused on the extraction of information for the DORA compliance template, and Experiment 2 targeted the extraction of obligations from contracts. Both experiments used a LLMs for the extraction tasks.

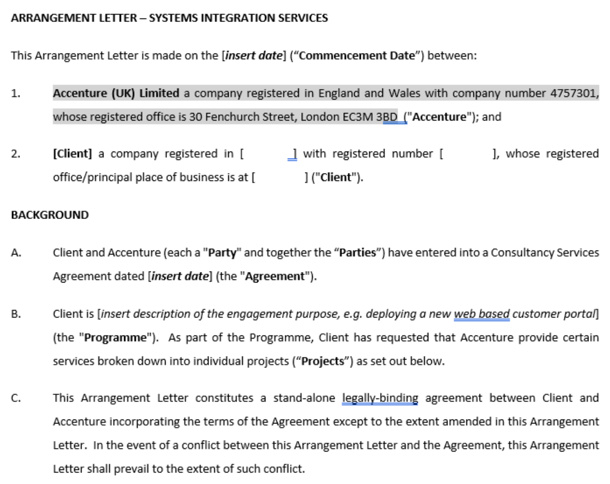
4.1 Experiment 1: DORA Compliance Template Extraction

The primary objective of Experiment 1 was to assess the effectiveness of Large Language Models (LLMs) in identifying and filling in the required information for the DORA compliance template.

4.1.1 Contract Generation

Despite extensive efforts, including contacting various individuals at Accenture, obtaining applicable contracts for filling in the DORA template proved challenging. These contracts are highly confidential, and with the EU DORA act set to commence in 2025, there is limited active work on it, making it difficult to find relevant materials. Consequently, creating custom contracts became the next viable option.

A legal professional was consulted to obtain an arrangement letter template contract. This template, however, lacked the specific information required for DORA compliance. To address this, a script was developed to generate contract information and populate the template.
The initial template (Figure 6) was modified to include placeholder fields. These placeholders, formatted as [x], represent the descriptions of the required information, for instance, [Name of the financial entity]. A total of 35 pieces of information necessary for the DORA template were identified and corresponding placeholders were incorporated into the contract, resulting in a modified template (Figure 7). To introduce more variation, five different versions of the contract template were created. In these variations, the order of texts and clauses was plausibly rearranged. When generating a contract, the starting template is randomly selected from these five variations, ensuring a diverse set of contracts.



ARRANGEMENT LETTER – SYSTEMS INTEGRATION SERVICES

This Arrangement Letter is made on the [***insert date***] ("**Commencement Date**") between:

1. **Accenture (UK) Limited** a company registered in England and Wales with company number 4757301, whose registered office is 30 Fenchurch Street, London EC3M 3BD ("**Accenture**"); and

2. **[Client]** a company registered in [ ] with registered number [ ], whose registered office/principal place of business is at [ ] ("**Client**").

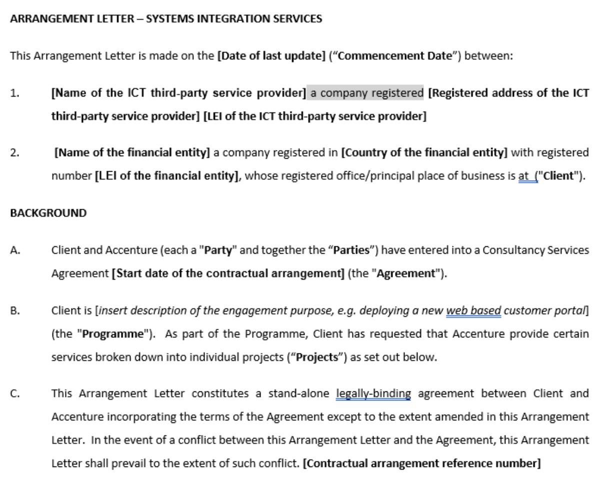
BACKGROUND

A. Client and Accenture (each a "**Party**" and together the "**Parties**") have entered into a Consultancy Services Agreement dated [***insert date***] (the "**Agreement**").

B. Client is [*insert description of the engagement purpose, e.g. deploying a new web based customer portal*] (the "**Programme**"). As part of the Programme, Client has requested that Accenture provide certain services broken down into individual projects ("**Projects**") as set out below.

C. This Arrangement Letter constitutes a stand-alone legally-binding agreement between Client and Accenture incorporating the terms of the Agreement except to the extent amended in this Arrangement Letter. In the event of a conflict between this Arrangement Letter and the Agreement, this Arrangement Letter shall prevail to the extent of such conflict.

Figure 6: Initial Template Contract



ARRANGEMENT LETTER – SYSTEMS INTEGRATION SERVICES

This Arrangement Letter is made on the **[Date of last update]** ("**Commencement Date**") between:

1. **[Name of the ICT third-party service provider]** a company registered **[Registered address of the ICT third-party service provider] [LEI of the ICT third-party service provider]**

2. **[Name of the financial entity]** a company registered in **[Country of the financial entity]** with registered number **[LEI of the financial entity]**, whose registered office/principal place of business is at ("**Client**").

BACKGROUND

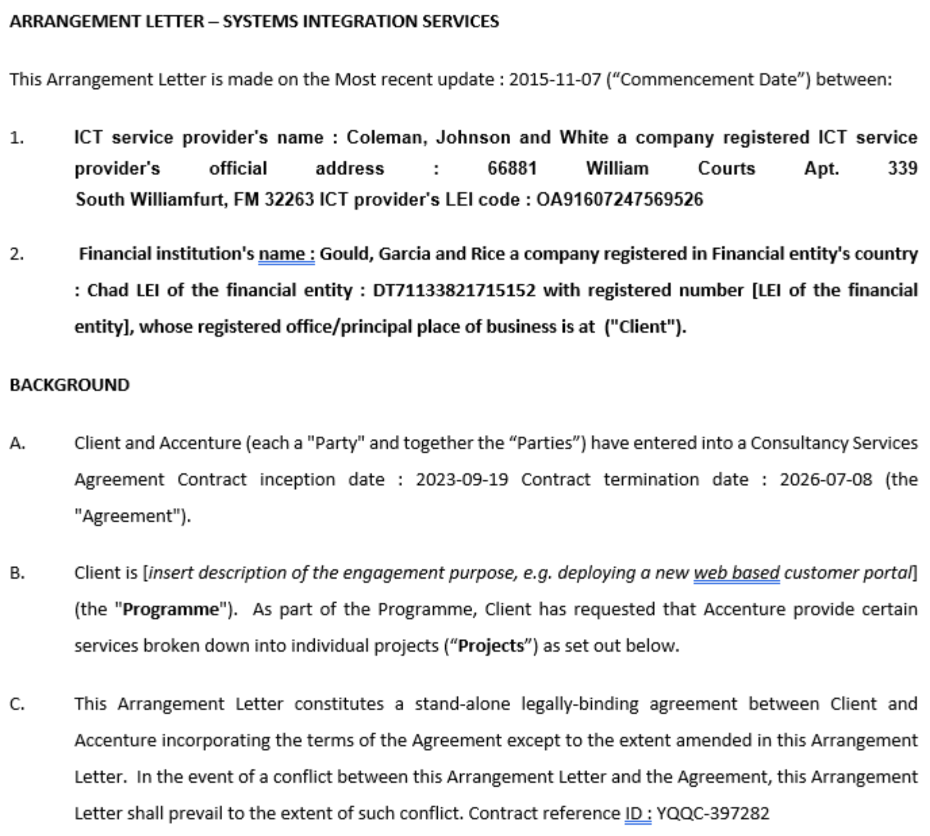
A. Client and Accenture (each a "**Party**" and together the "**Parties**") have entered into a Consultancy Services Agreement **[Start date of the contractual arrangement]** (the "**Agreement**").

B. Client is [*insert description of the engagement purpose, e.g. deploying a new web based customer portal*] (the "**Programme**"). As part of the Programme, Client has requested that Accenture provide certain services broken down into individual projects ("**Projects**") as set out below.

C. This Arrangement Letter constitutes a stand-alone legally-binding agreement between Client and Accenture incorporating the terms of the Agreement except to the extent amended in this Arrangement Letter. In the event of a conflict between this Arrangement Letter and the Agreement, this Arrangement Letter shall prevail to the extent of such conflict. **[Contractual arrangement reference number]**

Figure 7: Template Contract with Placeholders

A script was then created to look up these placeholders and insert generated information. Inserting raw data without context would result in an incoherent document, with seemingly random data scattered throughout. To mitigate this, each piece of information was prefixed with a contextual sentence derived from a DORA template description. For example, for the placeholder [Name of the financial entity], the script would insert ”The Name of the financial entity Brett & Brockly,” where ”The Name of the financial entity” provides context and ”Brett & Brockly” is the generated data. Figure 8 shows an example of the front page of one such generated contract.



ARRANGEMENT LETTER – SYSTEMS INTEGRATION SERVICES

This Arrangement Letter is made on the Most recent update : 2015-11-07 (“Commencement Date”) between:

1. **ICT service provider's name : Coleman, Johnson and White a company registered ICT service provider's official address : 66881 William Courts Apt. 339 South Williamfurt, FM 32263 ICT provider's LEI code : OA91607247569526**

2. **Financial institution's name : Gould, Garcia and Rice a company registered in Financial entity's country : Chad LEI of the financial entity : DT71133821715152 with registered number [LEI of the financial entity], whose registered office/principal place of business is at ("Client").**

BACKGROUND

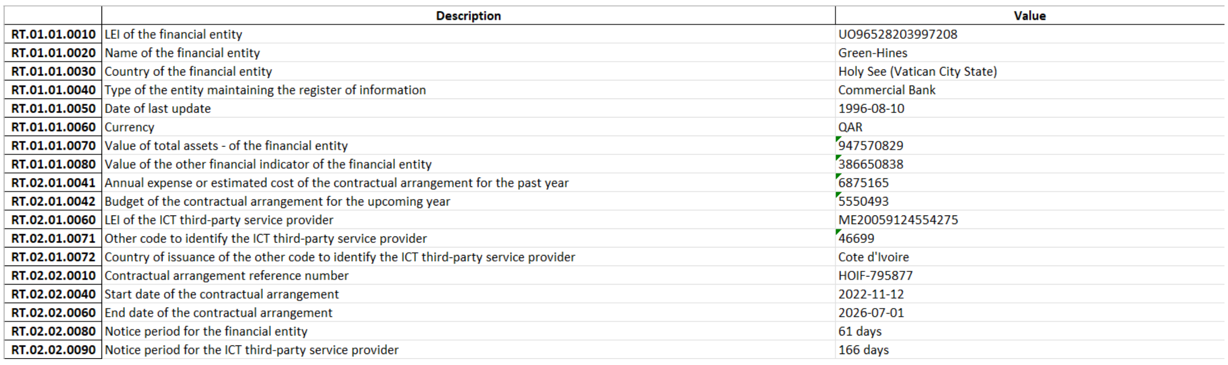
A. Client and Accenture (each a "Party" and together the “Parties”) have entered into a Consultancy Services Agreement Contract inception date : 2023-09-19 Contract termination date : 2026-07-08 (the "Agreement").

B. Client is [*insert description of the engagement purpose, e.g. deploying a new web based customer portal*] (the "**Programme**"). As part of the Programme, Client has requested that Accenture provide certain services broken down into individual projects (“**Projects**”) as set out below.

C. This Arrangement Letter constitutes a stand-alone legally-binding agreement between Client and Accenture incorporating the terms of the Agreement except to the extent amended in this Arrangement Letter. In the event of a conflict between this Arrangement Letter and the Agreement, this Arrangement Letter shall prevail to the extent of such conflict. Contract reference ID : YQQC-397282

Figure 8: Example of a Generated Contract

Additionally, the inserted generated information was saved in an Excel file as the ground truth data to facilitate later quantification of the results. An example of this is shown in Figure 9.



	Description	Value
RT.01.01.0010	LEI of the financial entity	UO96528203997208
RT.01.01.0020	Name of the financial entity	Green-Hines
RT.01.01.0030	Country of the financial entity	Holy See (Vatican City State)
RT.01.01.0040	Type of the entity maintaining the register of information	Commercial Bank
RT.01.01.0050	Date of last update	1996-08-10
RT.01.01.0060	Currency	QAR
RT.01.01.0070	Value of total assets - of the financial entity	947570829
RT.01.01.0080	Value of the other financial indicator of the financial entity	386650838
RT.02.01.0041	Annual expense or estimated cost of the contractual arrangement for the past year	6875165
RT.02.01.0042	Budget of the contractual arrangement for the upcoming year	5550493
RT.02.01.0060	LEI of the ICT third-party service provider	ME20059124554275
RT.02.01.0071	Other code to identify the ICT third-party service provider	46699
RT.02.01.0072	Country of issuance of the other code to identify the ICT third-party service provider	Cote d'Ivoire
RT.02.02.0010	Contractual arrangement reference number	HOIF-795877
RT.02.02.0040	Start date of the contractual arrangement	2022-11-12
RT.02.02.0060	End date of the contractual arrangement	2026-07-01
RT.02.02.0080	Notice period for the financial entity	61 days
RT.02.02.0090	Notice period for the ICT third-party service provider	166 days

Figure 9: Example of Ground Truth Data for DORA Compliance

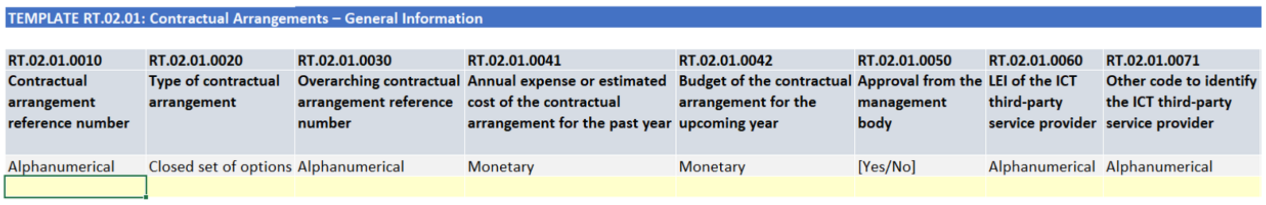
4.1.2 Input data

The input data for the LLM consisted of the generated contracts and the DORA compliance template entry descriptions.

Contract The documents are initially in DOCX formats and need to be converted into plain text-string for further processing. For these DOCX files, the `Document` class from the `docx` library is used, which allows for straightforward extraction of raw text from the documents. Once the text is extracted, it is concatenated into a single string so that it can be inserted into the prompt.

Template As input we also used the DORA compliance template descriptions. The DORA compliance excel file was loaded into our system using the `pandas` library and contains these fields: ”Template Code”, ”Description” and ”Type”. We use the ”Description” information in the prompt to describe the information that the LLM needs to extract for this entry from the stringified contract.

Moreover, the details from the Excel file are stored in a python dictionary, which maps template codes to their corresponding descriptions. This dictionary serves as a reference for the Large Language Model (LLM) during the text processing phase, enabling it to accurately identify and correlate the extracted information with the appropriate fields in the compliance template. Figure 10 illustrates a piece of the Excel file, showing the organization of the template.



TEMPLATE RT.02.01: Contractual Arrangements – General Information

RT.02.01.0010 Contractual arrangement reference number	RT.02.01.0020 Type of contractual arrangement	RT.02.01.0030 Overarching contractual arrangement reference number	RT.02.01.0041 Annual expense or estimated cost of the contractual arrangement for the past year	RT.02.01.0042 Budget of the contractual arrangement for the upcoming year	RT.02.01.0050 Approval from the management body	RT.02.01.0060 LEI of the ICT third-party service provider	RT.02.01.0071 Other code to identify the ICT third-party service provider
Alphanumerical	Closed set of options	Alphanumerical	Monetary	Monetary	[Yes/No]	Alphanumerical	Alphanumerical

Figure 10: Part of the Excel file containing the DORA compliance template, outlining the template codes, descriptions, information types, fill-in instructions, and whether the information is mandatory.

4.1.3 Prompt

The extraction task is guided by a prompt, which directs the LLM to process the contract text and systematically extract information according to the entry descriptions of the DORA template. The prompt is structured to facilitate the creation of a structured output in the form of a Python dictionary, where each key corresponds to a template code from the DORA compliance template, and the value is a list containing the information description and extracted information. The prompt is presented below:

```
prompt = ChatPromptTemplate.from_messages([
    ("user", '''
    Given a dictionary with template codes as keys and template entry descriptions as
    values: '''+ template_dict_str + '''. Extract the entries from the provided
    document: ''' + contract + '''.

    Create a dictionary with the following structure:
    - Key: Template code
    - Value: A list containing:
        1. Entry description from the input dictionary
        2. The entry found in the document

      Exclude any entry descriptions from the output if
      their corresponding entry cannot be found in the document.
      Ensure all dictionaries and lists are properly closed with brackets.

      Your task is to structure the output as a python dictionary
      and include only the relevant details as specified.

      An example of the output is: ''' + example_output)
])
```

The variable `template_dict_str` represents the DORA compliance template, which includes template codes and entry descriptions as a dictionary string, while the variable `contract` contains the concatenated contract documents. Finally the variable `example_output` contains a string of an example output in order to clarify how the output should look.

1. **Dictionary as Input:** The prompt begins by defining a dictionary structure where each key is a template code, and the value is a description. This tells the model what to look for in the contract.

2. **Structured Output:** The model is instructed to output its findings in a structured Python dictionary format, where each entry corresponds to a template code. For each code, the model must provide:

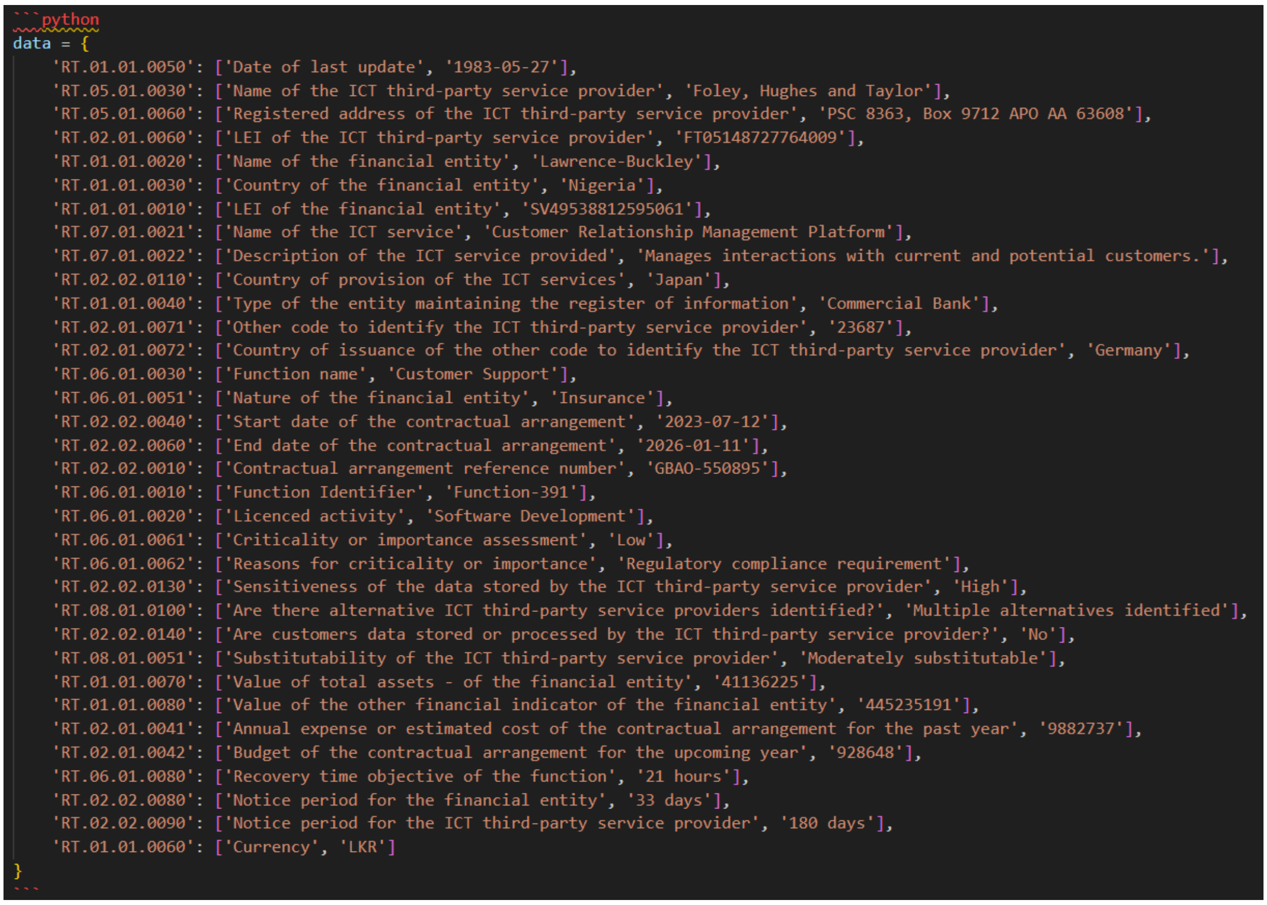
   - **Entry description:** The description of the entry.
   - **Extracted Answer:** Information extracted from the contract.

   This prompt output structure is necessary to properly parse the output as a data frame in the next step.

3. **Conditional Instructions:** The model is further guided to attempt extraction for all template codes and to structure its responses precisely as outlined. If the model cannot find or deduce information for a given template code, it is instructed to leave that entry blank, ensuring that only relevant entries are included.

4.1.4 Parsing the Output String

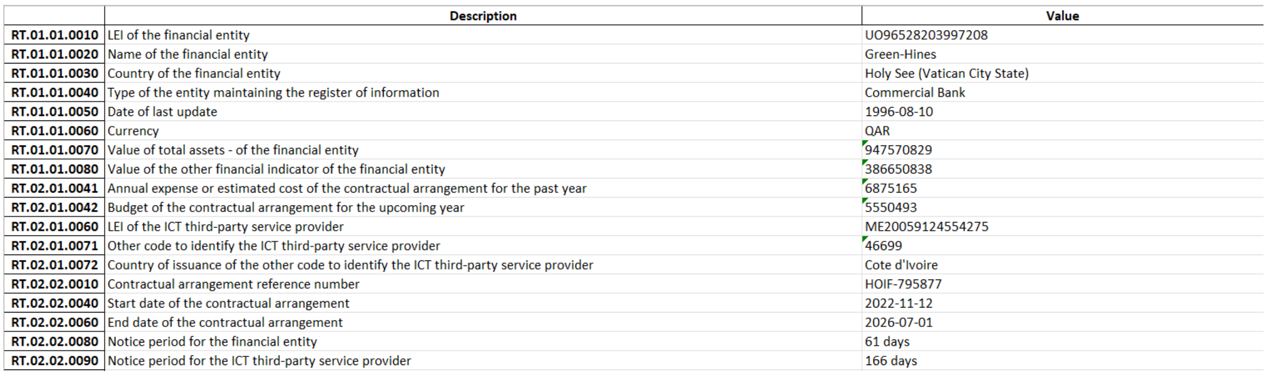
The prompt is passed to the GPT-4o API, which generates an output string. This output string is shown in Figure 11.



~~~
```python
data = {
    'RT.01.01.0050': ['Date of last update', '1983-05-27'],
    'RT.05.01.0030': ['Name of the ICT third-party service provider', 'Foley, Hughes and Taylor'],
    'RT.05.01.0060': ['Registered address of the ICT third-party service provider', 'PSC 8363, Box 9712 APO AA 63608'],
    'RT.02.01.0060': ['LEI of the ICT third-party service provider', 'FT05148727764009'],
    'RT.01.01.0020': ['Name of the financial entity', 'Lawrence-Buckley'],
    'RT.01.01.0030': ['Country of the financial entity', 'Nigeria'],
    'RT.01.01.0010': ['LEI of the financial entity', 'SV49538812595061'],
    'RT.07.01.0021': ['Name of the ICT service', 'Customer Relationship Management Platform'],
    'RT.07.01.0022': ['Description of the ICT service provided', 'Manages interactions with current and potential customers.'],
    'RT.02.02.0110': ['Country of provision of the ICT services', 'Japan'],
    'RT.01.01.0040': ['Type of the entity maintaining the register of information', 'Commercial Bank'],
    'RT.02.01.0071': ['Other code to identify the ICT third-party service provider', '23687'],
    'RT.02.01.0072': ['Country of issuance of the other code to identify the ICT third-party service provider', 'Germany'],
    'RT.06.01.0030': ['Function name', 'Customer Support'],
    'RT.06.01.0051': ['Nature of the financial entity', 'Insurance'],
    'RT.02.02.0040': ['Start date of the contractual arrangement', '2023-07-12'],
    'RT.02.02.0060': ['End date of the contractual arrangement', '2026-01-11'],
    'RT.02.02.0010': ['Contractual arrangement reference number', 'GBAO-550895'],
    'RT.06.01.0010': ['Function Identifier', 'Function-391'],
    'RT.06.01.0020': ['Licenced activity', 'Software Development'],
    'RT.06.01.0061': ['Criticality or importance assessment', 'Low'],
    'RT.06.01.0062': ['Reasons for criticality or importance', 'Regulatory compliance requirement'],
    'RT.02.02.0130': ['Sensitiveness of the data stored by the ICT third-party service provider', 'High'],
    'RT.08.01.0100': ['Are there alternative ICT third-party service providers identified?', 'Multiple alternatives identified'],
    'RT.02.02.0140': ['Are customers data stored or processed by the ICT third-party service provider?', 'No'],
    'RT.08.01.0051': ['Substitutability of the ICT third-party service provider', 'Moderately substitutable'],
    'RT.01.01.0070': ['Value of total assets - of the financial entity', '41136225'],
    'RT.01.01.0080': ['Value of the other financial indicator of the financial entity', '445235191'],
    'RT.02.01.0041': ['Annual expense or estimated cost of the contractual arrangement for the past year', '9882737'],
    'RT.02.01.0042': ['Budget of the contractual arrangement for the upcoming year', '928648'],
    'RT.06.01.0080': ['Recovery time objective of the function', '21 hours'],
    'RT.02.02.0080': ['Notice period for the financial entity', '33 days'],
    'RT.02.02.0090': ['Notice period for the ICT third-party service provider', '180 days'],
    'RT.01.01.0060': ['Currency', 'LKR']
}
```
~~~

Figure 11: Example of Generated Output String

To parse the output string, a regular expression (r"{(.*?)}") is used to extract the content between the brackets. The extracted string is then interpreted as a dictionary using `ast.literal_eval(output_stripped)`. This resulting dictionary is subsequently saved as an Excel file for further analysis.
The extracted information output Excel file appears as shown in Figure 12.



	Description	Value
RT.01.01.0010	LEI of the financial entity	UO96528203997208
RT.01.01.0020	Name of the financial entity	Green-Hines
RT.01.01.0030	Country of the financial entity	Holy See (Vatican City State)
RT.01.01.0040	Type of the entity maintaining the register of information	Commercial Bank
RT.01.01.0050	Date of last update	1996-08-10
RT.01.01.0060	Currency	QAR
RT.01.01.0070	Value of total assets - of the financial entity	947570829
RT.01.01.0080	Value of the other financial indicator of the financial entity	386650838
RT.02.01.0041	Annual expense or estimated cost of the contractual arrangement for the past year	6875165
RT.02.01.0042	Budget of the contractual arrangement for the upcoming year	5550493
RT.02.01.0060	LEI of the ICT third-party service provider	ME20059124554275
RT.02.01.0071	Other code to identify the ICT third-party service provider	46699
RT.02.01.0072	Country of issuance of the other code to identify the ICT third-party service provider	Cote d'Ivoire
RT.02.02.0010	Contractual arrangement reference number	HOIF-795877
RT.02.02.0040	Start date of the contractual arrangement	2022-11-12
RT.02.02.0060	End date of the contractual arrangement	2026-07-01
RT.02.02.0080	Notice period for the financial entity	61 days
RT.02.02.0090	Notice period for the ICT third-party service provider	166 days

Figure 12: Example of Extracted Information Output in Excel

Evaluation To evaluate the performance of the LLM in extracting the required entries from the contracts, an evaluation script was created. This script compares the LLM-extracted entries with the ground truth entries inserted into the contracts. The evaluation considers three types of errors:

- **False Positives (FP):** Entries that the LLM extracted but do not exist in the ground truth.
- **No-Match:** Entries where the LLM extracted a value, but it does not match the ground truth.
- **Missing Entries:** Required entries that the LLM failed to extract.

The accuracy of the LLM is calculated as the ratio of correctly extracted entries to the total number of required entries. Specifically, if out of the 35 ground truth template entries, x entries were correctly extracted, then:

$$\text{Accuracy} = \frac{x}{35}$$

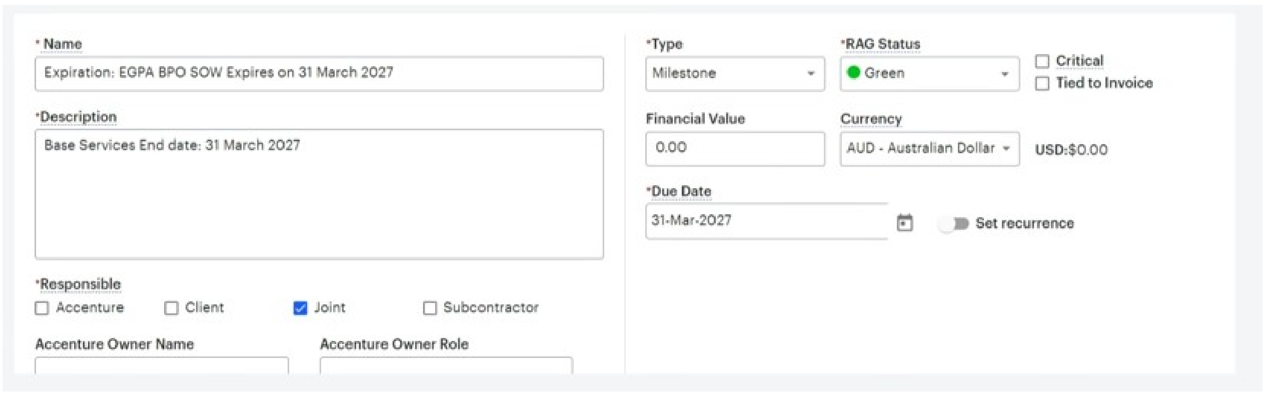
The comparison script performs the following steps for each document:

1. **Normalization:** Strings are normalized by removing non-alphanumeric characters, extra spaces, and converting to lowercase to ensure a fair comparison.
2. **Shared Keys Identification:** Identifies keys (template codes) that are shared between the ground truth and LLM-extracted entries.
3. **String Similarity Check:** Compares the normalized strings to determine if they match.
4. **Result Compilation:** Compiles results for false positives, missing entries, wrong entries, and calculates the accuracy.

The analysis included several statistical measures. Firstly, the mean accuracy was calculated to represent the average accuracy across all documents. To assess the consistency of this accuracy, the standard deviation was measured, providing insight into the variability of the results. Additionally, a 95% confidence interval was determined, offering a range within which the true mean accuracy is likely to fall.

4.2 Experiment 2: Extraction of Deliverables and Obligations

While the first project demonstrated promising results, it was relatively straightforward. The information that the LLM needed to identify was clearly indicated with a single context sentence, making it easy to locate. This project confirmed that the LLM could perform this simple task with high accuracy.
To further test the LLM's ability to extract information from contracts, a second project was undertaken. The goal of Experiment 2 is to evaluate the extraction program's capability to identify and extract obligations from contractual documents.
What are obligations? "Obligations in contracts refer to duties or commitments that parties are legally required to fulfill as per the terms of the contract."
Examples of obligations include payment deadlines, performance metrics, reports, and termination dates.
After a contract is signed, both parties are required to adhere to all the agreements stipulated within it. To ensure compliance, companies list all obligations and due dates in a data frame. At Accenture, the tool used for this purpose is called Manage My Contract (MMC). Once a contract is signed, a



* Name	Expiration: EGPA BPO SOW Expires on 31 March 2027
*Description	Base Services End date: 31 March 2027
*Responsible	☐ Accenture ☐ Client ✓ Joint ☐ Subcontractor
Accenture Owner Name	
Accenture Owner Role	
*Type	Milestone
*RAG Status	● Green
☐ Critical	
☐ Tied to Invoice	
Financial Value	0.00
Currency	AUD - Australian Dollar
USD:$0.00	
*Due Date	31-Mar-2027
Set recurrence	

Figure 13: Manage My Contract (MMC) Interface with a termination obligation filled in.

legal professional from Accenture reviews it and annotates all obligations using the MMC interface, as shown in Figure 13.
For each obligation, the legal professional must provide the name, description, responsible party, type, RAG status, and due date. This is a manually intensive task, and overlooking an important obligation can have severe consequences.

4.2.1 Dataset

After an extensive search spanning several months and even signing a second NDA, getting contract data wasn't possible. Consequently, the next best option was to use a publicly available dataset. The chosen dataset was the Contract Understanding Atticus Dataset (CUAD), a corpus of 510 commercial legal contracts. These contracts do not come with obligation annotations, therefore they had to be self-annotated.
The initial plan involved inserting these contracts into the LLM prompt and requesting the LLM to identify all obligations along with the secondary information required by the MMC UI (name, description, responsible party, type, RAG status, and due date). This would produce an Excel file containing all obligations and their secondary information, which could then be compared to a ground truth excel list of obligations. However, a challenge arose: how to compare obligations in the LLM-predicted Excel list to those in the ground truth Excel list. The lists are not indexed or ordered, making direct comparison difficult.
This issue is referred to as the comparison problem. Attempting to link obligations based on due dates can lead to mismatches due to coincidental date similarities, and using descriptions can be problematic because the same obligation might be described differently in both lists. The same applies to names. Figure 14 illustrates this problem and its solution. In the figure, the same obligation is described differently in the LLM-predicted obligation description and the human-annotated obligation description.
The solution is to request the verbatim text from the contract along with the currently required information. This allows obligations to be identified based on matching verbatim texts.

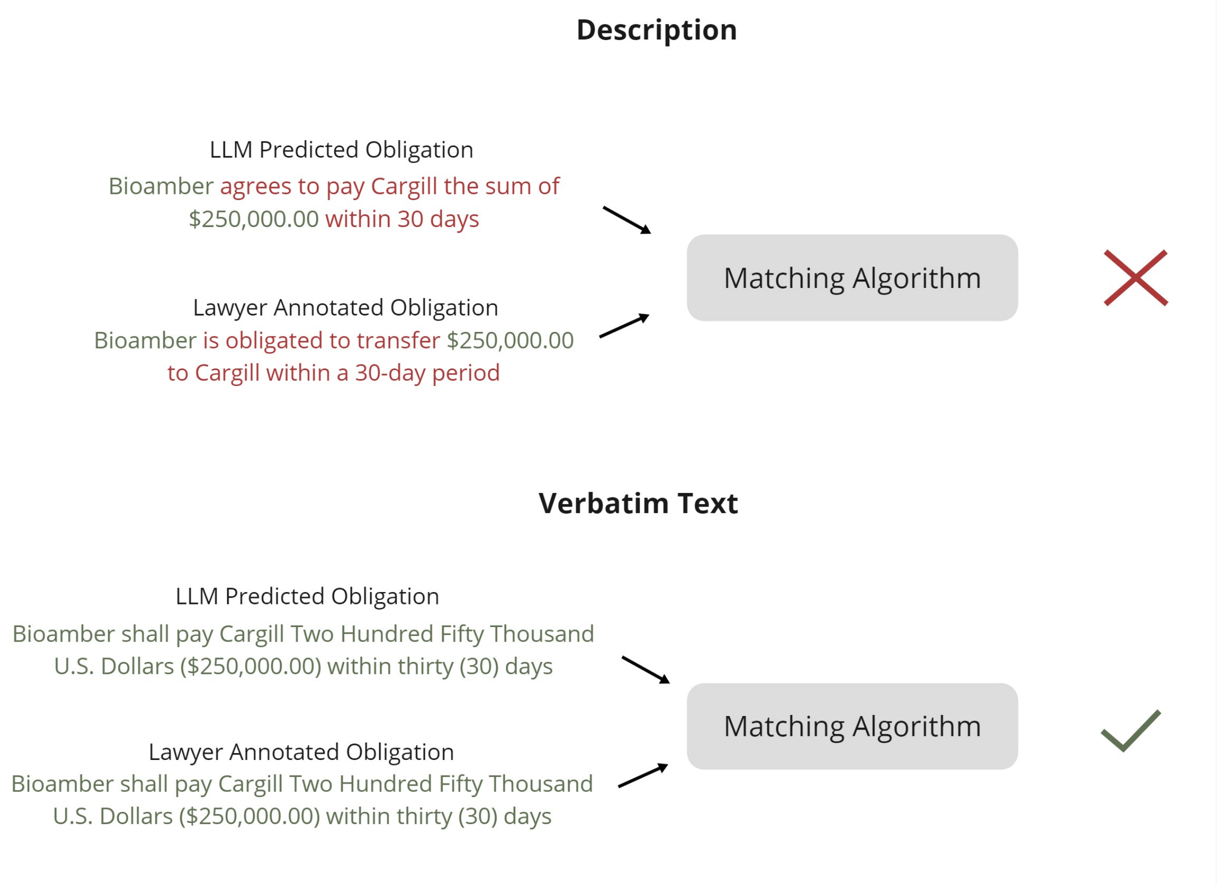
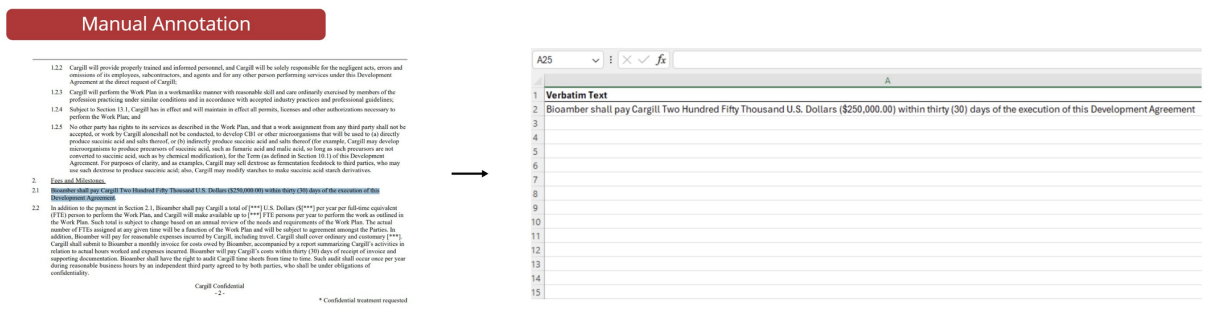

Description
LLM Predicted Obligation
Bioamber agrees to pay Cargill the sum of $250,000.00 within 30 days
Lawyer Annotated Obligation
Bioamber is obligated to transfer $250,000.00 to Cargill within a 30-day period
Matching Algorithm
Verbatim Text
LLM Predicted Obligation
Bioamber shall pay Cargill Two Hundred Fifty Thousand U.S. Dollars ($250,000.00) within thirty (30) days
Lawyer Annotated Obligation
Bioamber shall pay Cargill Two Hundred Fifty Thousand U.S. Dollars ($250,000.00) within thirty (30) days
Matching Algorithm

Figure 14: Illustration of the Comparison Problem and its Solution

4.2.2 Annotation

The next step involved annotating this data. Six contracts were manually selected from the total 510 legal contracts, consisting of four consulting agreements and two license development agreements. These contracts were chosen based on the advice of a consulted legal professional and the consulting nature of Accenture. Each contract was carefully reviewed, and the verbatim text for each identified obligation was copied into an Excel file, as illustrated in Figure 15.



1.2.2 Cargill will provide properly trained and informed personnel, and Cargill will be solely responsible for the negligent acts, errors and omissions of its employees, subcontractors, and agents and for any other person performing services under this Development Agreement at the direct request of Cargill;

1.2.3 Cargill will perform the Work Plan in a workmanlike manner with reasonable skill and care ordinarily exercised by members of the profession practicing under similar conditions and in accordance with accepted industry practices and professional guidelines;

1.2.4 Subject to Section 13.1, Cargill has in effect and will maintain in effect all permits, licenses and other authorizations necessary to perform the Work Plan; and

1.2.5 No other party has rights to its services as described in the Work Plan, and that a work assignment from any third party shall not be accepted, or work by Cargill aloneshall not be conducted, to develop CB1 or other microorganisms that will be used to (a) directly produce succinic acid and salts thereof, or (b) indirectly produce succinic acid and salts thereof (for example, Cargill may develop microorganisms to produce precursors of succinic acid, such as fumaric acid and malic acid, so long as such precursors are not converted to succinic acid, such as by chemical modification), for the Term (as defined in Section 10.1) of this Development Agreement. For purposes of clarity, and as examples, Cargill may sell dextrose as fermentation feedstock to third parties, who may use such dextrose to produce succinic acid; also, Cargill may modify starches to make succinic acid starch derivatives.

2. Fees and Milestones

2.1 Bioamber shall pay Cargill Two Hundred Fifty Thousand U.S. Dollars ($250,000.00) within thirty (30) days of the execution of this Development Agreement.

2.2 In addition to the payment in Section 2.1, Bioamber shall pay Cargill a total of [***] U.S. Dollars ($[***] per year per full-time equivalent (FTE) person to perform the Work Plan, and Cargill will make available up to [***] FTE persons per year to perform the work as outlined in the Work Plan. Such total is subject to change based on an annual review of the needs and requirements of the Work Plan. The actual number of FTEs assigned at any given time will be a function of the Work Plan and will be subject to agreement amongst the Parties. In addition, Bioamber will pay for reasonable expenses incurred by Cargill, including travel. Cargill shall cover ordinary and customary [***]. Cargill shall submit to Bioamber a monthly invoice for costs owed by Bioamber, accompanied by a report summarizing Cargill's activities in relation to actual hours worked and expenses incurred. Bioamber will pay Cargill's costs within thirty (30) days of receipt of invoice and supporting documentation. Bioamber shall have the right to audit Cargill time sheets from time to time. Such audit shall occur once per year during reasonable business hours by an independent third party agreed to by both parties, who shall be under obligations of confidentiality.

Cargill Confidential
- 2 -

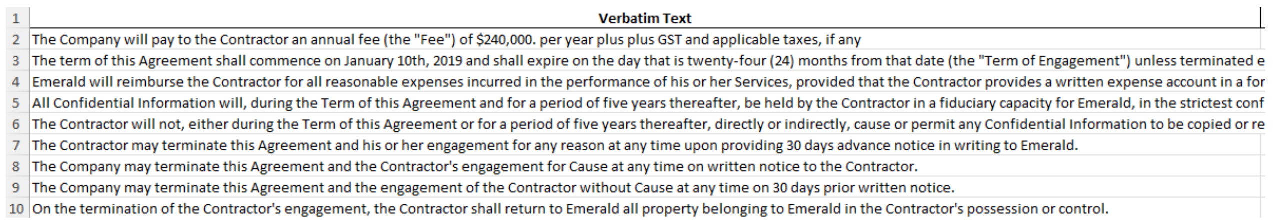
* Confidential treatment requested

A25

	A
1	**Verbatim Text**
2	Bioamber shall pay Cargill Two Hundred Fifty Thousand U.S. Dollars ($250,000.00) within thirty (30) days of the execution of this Development Agreement
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

Figure 15: Manual Annotation of Contract Obligations

The final ground truth Excel sheet appeared as shown in Figure 16.



1	Verbatim Text
2	The Company will pay to the Contractor an annual fee (the "Fee") of $240,000. per year plus plus GST and applicable taxes, if any
3	The term of this Agreement shall commence on January 10th, 2019 and shall expire on the day that is twenty-four (24) months from that date (the "Term of Engagement") unless terminated e
4	Emerald will reimburse the Contractor for all reasonable expenses incurred in the performance of his or her Services, provided that the Contractor provides a written expense account in a for
5	All Confidential Information will, during the Term of this Agreement and for a period of five years thereafter, be held by the Contractor in a fiduciary capacity for Emerald, in the strictest conf
6	The Contractor will not, either during the Term of this Agreement or for a period of five years thereafter, directly or indirectly, cause or permit any Confidential Information to be copied or re
7	The Contractor may terminate this Agreement and his or her engagement for any reason at any time upon providing 30 days advance notice in writing to Emerald.
8	The Company may terminate this Agreement and the Contractor's engagement for Cause at any time on written notice to the Contractor.
9	The Company may terminate this Agreement and the engagement of the Contractor without Cause at any time on 30 days prior written notice.
10	On the termination of the Contractor's engagement, the Contractor shall return to Emerald all property belonging to Emerald in the Contractor's possession or control.

Figure 16: Example of Annotated Data Point in Ground Truth Excel Sheet

4.2.3 Prompt

The extraction is guided by a prompt, which directs the LLM to process the contract text and extract information according to the description of important obligations. The prompt is designed to make the LLM output a structured Python dictionary, where each key corresponds to an index number, and the value is a list containing the verbatim text of an obligation.
Due to the limited output token space of an LLM the output can cutoff prematurely. To prevent the LLM from cutting off its response before listing all found obligations, the task was divided into two parts: one for extracting the verbatim text and another for acquiring the secondary information (name, description, responsible party, type, RAG status, and due date).
The prompt is divided into three parts: the general task description, the formatting instructions, and the obligation description.
General Task Description:

```
prompt = ChatPromptTemplate.from_messages([
    ("user", '''
        Extract verbatim text of each important deliverable, obligation, milestone,
            or governance agreement from the provided contract: "{contract}".
        Focus only on items critical for compliance and tracking within the agreement.
        Only include deliverables, obligations, milestones, or governance agreements
            that are pivotal.
        Typically, these will have specified due dates and may carry consequences if not met.
        Ensure that the text extracted is verbatim. If an item does not specify a due date,
            consider its importance before deciding to include it.
        Essentially, these are the most important elements within the contract,
            the items both parties must not overlook.
    ''')
])
```

Here, the `contract` variable represents the contract text as a string. The general task description is outlined to ensure that the LLM focuses on extracting only the most important obligations from the contract.

Formatting Instructions:

```
    '''
        Create a dictionary with the following structure:
        - Key: Index number for the deliverable, obligation,
            milestone, or governance agreement
        - Value: A list containing:
            1. A verbatim quotation of the deliverable, obligation,
               milestone, or governance agreement as stated in the contract.

        Ensure all dictionaries and lists are properly closed with brackets.
        Avoid multiple entries for very similar obligations or milestones.
        For example, different clauses leading to contract termination
            can be consolidated into a single entry.
        Anticipate finding approximately 10 to 30 critical items in
            the contract.
        If there are double quotations (") in the verbatim text,
            replace them with <<< to prevent parsing issues.
     '''
```

In this part of the prompt, the LLM is instructed to output the string in a dictionary format so that it can later be parsed and saved as an Excel file. The instruction to replace double quotations with ¡¡¡ ensures that the parsing step is not disrupted.

Obligation Descriptions:

```
    '''
        Important deliverables, obligations, milestones, and governance
            agreements include but are not limited to:
        1. Contract end or extension dates.
        2. Details of promotional activities or lists.
        3. Volume discounts related to annual expenditures.
        4. Hiring decisions from contract-to-hire channels.
        5. Baseline performance metrics of parties involved.
        6. Requirements for continuous improvement or cost reduction.
        7. Audit reports.
        8. Schedules for mandatory meetings (monthly, quarterly, etc.).
        9. Reporting deadlines.
        10. Price indexation clauses.

        Some correct examples of verbatim texts would be:
        "X shall pay Y a non-refundable, non-creditable license
        issue fee of $500,000 within three (3) Business Days
        following the date the Parties enter into this
        Agreement and the Supply Agreement.",

        "X shall keep the Y informed regarding the progress
        and results of such Commercialization.
        Such progress reports shall be provided at least quarterly
        and in a form reasonably acceptable to Y.",

        "The Y shall hold meetings as mutually agreed by the Parties,
        but in no event less than quarterly unless X and Y
        mutually agree in writing (which may include email),
        no later than thirty (30) days in advance of any
        meeting following the initial meeting of the Y."
    '''
```

A legal professional provided a contract with its corresponding obligations. Although this contract could not be included in the dataset due to confidentiality, it offered valuable insight into what obligations are. From this obligations list, ten types of obligations were extracted and listed in the prompt to clarify what the LLM should extract from the contract. Additionally, three literal examples of verbatim obligation texts were provided to guide the LLM.

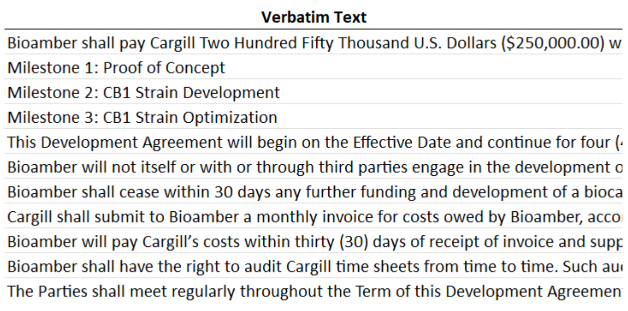
4.2.4 Parsing the Output String

The prompt is passed to either the GPT-4o, Claude 3.5 Sonnet, or Gemini 1.5 Pro API, which generates an output string.
To parse the output string, the following steps are used:

1. **Clean the text:** Remove all newline characters (\n) to prevent interference during parsing.
2. **Fix incomplete output:** If the LLM cuts off its response and the dictionary string is not properly finished, search for the last completed entry in the dictionary string output. Remove everything after it and append the correct closing brackets `"]}`.
3. **Extract content with regular expression:** Use a regular expression (r"{(.*?)}") to extract the content between the brackets.
4. **Interpret as a dictionary:** Convert the extracted string into a dictionary using `ast.literal_eval(output_stripped)`.
5. **Replace placeholders:** Convert all instances of <<<back to double quotes (").
6. **Save as an Excel file:** The resulting dictionary is subsequently saved as an Excel file for further analysis.

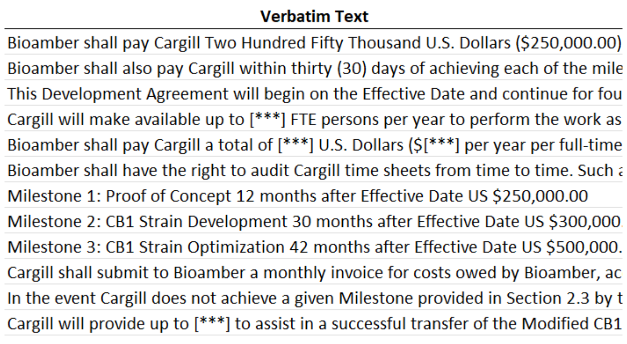
4.2.5 Matching LLM Predictions with Ground Truth

The extracted information output Excel file and the ground truth Excel file are shown in Figures 17 and 18, respectively.



Verbatim Text
Bioamber shall pay Cargill Two Hundred Fifty Thousand U.S. Dollars ($250,000.00) w
Milestone 1: Proof of Concept
Milestone 2: CB1 Strain Development
Milestone 3: CB1 Strain Optimization
This Development Agreement will begin on the Effective Date and continue for four (
Bioamber will not itself or with or through third parties engage in the development o
Bioamber shall cease within 30 days any further funding and development of a bioca
Cargill shall submit to Bioamber a monthly invoice for costs owed by Bioamber, acco
Bioamber will pay Cargill's costs within thirty (30) days of receipt of invoice and supp
Bioamber shall have the right to audit Cargill time sheets from time to time. Such au
The Parties shall meet regularly throughout the Term of this Development Agreemen

Figure 17: LLM-predicted obligations



Verbatim Text
Bioamber shall pay Cargill Two Hundred Fifty Thousand U.S. Dollars ($250,000.00)
Bioamber shall also pay Cargill within thirty (30) days of achieving each of the mile
This Development Agreement will begin on the Effective Date and continue for fou
Cargill will make available up to [***] FTE persons per year to perform the work as
Bioamber shall pay Cargill a total of [***] U.S. Dollars ($[***] per year per full-time
Bioamber shall have the right to audit Cargill time sheets from time to time. Such a
Milestone 1: Proof of Concept 12 months after Effective Date US $250,000.00
Milestone 2: CB1 Strain Development 30 months after Effective Date US $300,000
Milestone 3: CB1 Strain Optimization 42 months after Effective Date US $500,000.
Cargill shall submit to Bioamber a monthly invoice for costs owed by Bioamber, ac
In the event Cargill does not achieve a given Milestone provided in Section 2.3 by t
Cargill will provide up to [***] to assist in a successful transfer of the Modified CB1

Figure 18: Manually annotated ground-truth obligations

A fuzzy matching technique was introduced to align the verbatim texts. Each obligation in the ground truth list was compared to the obligations extracted by the LLM.
Initially, the verbatim texts were matched within the contract using the `fuzzysearch` method, providing the index location of each ground truth obligation and each predicted obligation. This method prioritizes the text's location within the contract over the exact start or end position of the verbatim text.
Once matches were identified in the contract, the start and end indices of the matched texts were analyzed to determine if there was an overlap between the predicted obligations and the ground truth obligations. Any overlap in these indices was considered a match.
To evaluate the fuzzy matching method's accuracy, the Levenshtein ratio between the matched text in the contract and the LLM-predicted text was calculated. Figure 19 shows a histogram of the Levenshtein scores for all matches.

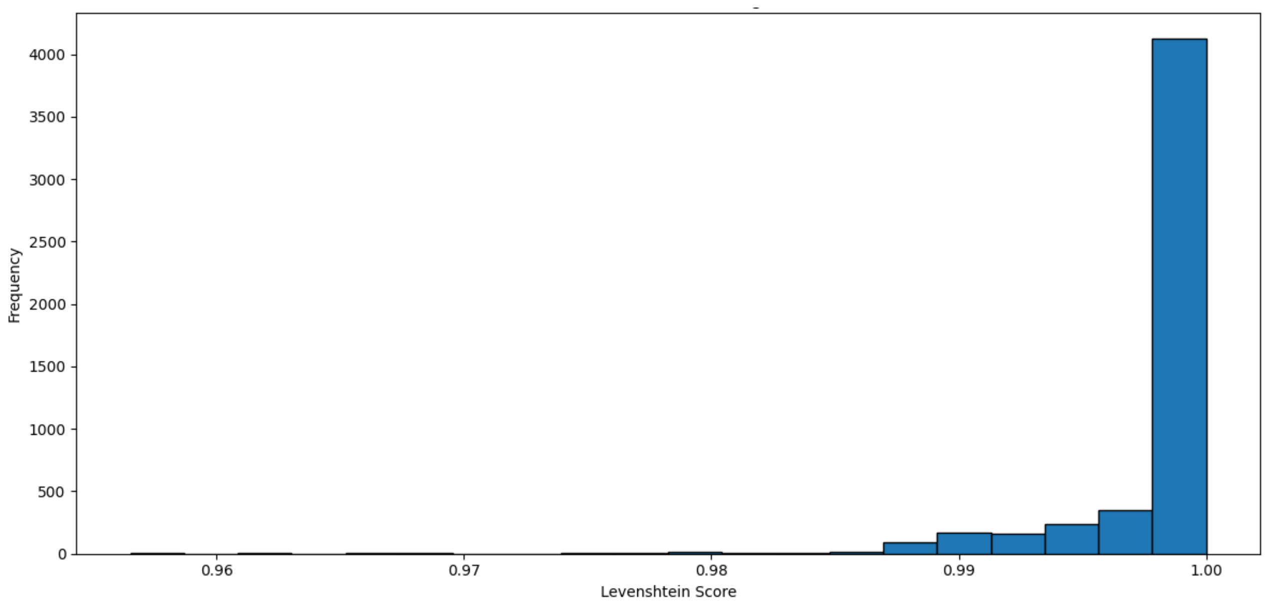


Figure 19: Histogram of Levenshtein scores for matched texts

Additionally, the top 10 worst-performing text matches were identified and are displayed in Figure 1. The differences between the matched texts and the predicted/ground truth texts are mainly due to the normalization process of the contract.

Score	**Searched Text**	**Matched Text**
0.9565	The executive acknowledges that the position of President and CEO will involve significant travel for business development and for investor relations.	the executive acknowledges that the position of president and ceo will involve significant travel for business development and for investor relations
0.9585	Vyera shall make the Equity Investment within seven (7) days of the Effective Date, as required by Section 8.13.	vyera fails to make the equity investment within seven 7 days of the effective date as required by section 813
0.9600	Time to achieve: Month 30	time to achieve: month 30
0.9610	This Agreement is made effective as of December 17, 2019 (the ”Effective Date”)	this “agreement” is made effective as of december 17 2019 the “effective date”

0.9620	Emerald will pay to the Contractor an annual fee (the ”Fee”) of $240,000. per year plus plus GST and applicable taxes, if any.	company will pay to the contractor an annual fee the ”fee” of 240000 per year plus plus gst and applicable taxes if any
0.9652	Vyera shall pay to CytoDyn [***] (the ”[***] Milestone Payment”) in the event that [***] (a ”[***]”) results in a [***].	vyera shall pay to cytodyn [***] the “[***] milestone payment” in the event that [***] a “[***]” results in a [***]
0.9680	Vyera shall enter into a Supply Agreement(s) for the commercial supply of Licensed Product on the Effective Date.	ties shall enter into a supply agreements for the commercial supply of licensed product on the effective date
0.9690	This Commercialization and License Agreement (this ”Agreement”) is made effective as of December 17, 2019 (the ”Effective Date”) by and between Vyera Pharmaceuticals, LLC, a Delaware limited liability company (”Vyera”), and CytoDyn Inc., a Delaware corporation (”CytoDyn”).	this commercialization and license agreement this “agreement” is made effective as of december 17 2019 the “effective date” by and between vyera pharmaceuticals llc a delaware limited liability company “vyera” and cytodyn inc a delaware corporation “cytodyn”
0.9718	The Consultant’s engagement shall terminate upon the Consultant’s death.	the consultant’s engagement shall terminate upon the consultant’s death

Table 1: Comparison of Search and Match Texts with the corresponding Levenshtein score

As illustrated, these matches are generally accurate, with only minor differences often caused by parsing adjustments.
Out of the total 5266 LLM-predicted obligations, 92 could not be found in the contract using the `fuzzysearch` method. The breakdown is as follows: GPT-4: 52, Gemini: 40, Claude: 0. Most of these discrepancies occurred because they involve lists that are truncated. For example, the contract states:

> The Chief Executive Officer of Emerald Health Naturals will:
>
> - Lead all aspects of the business in terms of strategic planning, product development, and operational execution on its annual and long-term objectives.
> - Actively manage the company’s P&L performance to ensure that its financial performance is in line with its budget projections and will enact any necessary changes to ensure that the business meets or exceeds such projections.

However, the LLM may output the obligations as:

> ”The Chief Executive Officer of Emerald Health Naturals will:
> Actively manage the company’s P&L performance to ensure that its financial performance is in line with its budget projections and will enact any necessary changes to ensure that the business meets or exceeds such projections.”

This discrepancy highlights the need for improved prompt engineering to address such mismatches.

4.2.6 Evaluation

To evaluate the performance of the LLM in extracting the verbatim obligation texts from the contracts, an evaluation script was created. This script compares the LLM-extracted verbatim text obligations with the ground truth verbatim text obligations. The evaluation considers two types of errors:

- **False Positives (FP):** Verbatim obligation texts that the LLM extracted but do not exist in the ground truth.
- **Missing:** Verbatim obligation texts in the ground truth that the LLM failed to extract.

The accuracy of the LLM is calculated as the ratio of correctly extracted verbatim obligation texts to the total number of ground truth verbatim obligation texts in a specific contract. Specifically, if out of the y ground truth verbatim obligation texts, x verbatim obligation texts were correctly extracted, then:

$$\text{Accuracy} = \frac{x}{y}$$

The comparison script performs the following steps for each document:

1. Cleaned the text data.
2. Identified and matched verbatim texts between predictions and ground truth.
3. Calculated accuracy and false positives for each iteration.
4. Summarized results and computed overall statistics, including mean accuracy, standard deviation, and confidence intervals.

The following statistics were calculated for each contract:

- **Average Accuracy:** The mean accuracy over multiple iterations.
- **Standard Deviation of Accuracy:** The variability of the accuracy across iterations.
- **95% Confidence Interval (CI) for Accuracy:** The range within which the true accuracy lies with 95% confidence.
- **Average False Positives per Iteration:** The mean number of false positives identified across iterations.

Each model was evaluated using x iterations per contract. For each iteration, the accuracy and false positives were computed. Then the average accuracy and false positives were calculated for each contract for each model. The final overall contract statistics for each model were determined by summarizing the individual contract iteration averages and calculating the overall standard deviation and confidence intervals.
Calculating the standard deviation of the overall contracts is different from merely averaging the contract average standard deviations. In order to accurately determine the combined standard deviation, we need to consider the variances and the number of observations. Here is the process:
First, for each dataset, calculate the variance by squaring the standard deviation:

$$\text{Variance} = (\text{Std Dev})^2 \tag{6}$$

Next, sum these variances across all datasets. Because each dataset has the same number of observations, we can directly average these variances. The combined variance, as shown in Equation 7, is calculated by dividing the sum of variances by the number of datasets:

$$\text{Combined Variance} = \frac{\sum_{i=1}^{n} \text{Variance}_i}{n} \tag{7}$$

Finally, take the square root of the combined variance to obtain the combined standard deviation:

$$\text{Combined Std Dev} = \sqrt{\text{Combined Variance}} \tag{8}$$

This method, known as pooling variances (Equation 8), ensures that the overall standard deviation accurately reflects the variability in the data across all contracts. This approach is recommended for cases where data sets are assumed to be from the same distribution [10].
The results were saved in an Excel file, containing detailed information on the performance of the LLM for each contract and overall performance across all contracts.

5 Results

5.1 Experiment 1: DORA Compliance Template Extraction

The objective of Experiment 1 was to evaluate the effectiveness of the Large Language Models (LLMs) in accurately filling out the DORA compliance template based on information extracted from contractual documents. This section presents the results of this experiment.
The extraction program was tasked with identifying and extracting information for a set of predefined fields within the DORA compliance template. The input consisted of contract documents alongside a structured prompt designed to guide the LLM. The expected output was a series of key-value pairs corresponding to the template fields and their extracted information. The extracted information was then compared to the ground truth to determine the accuracy in completing this task.

5.1.1 Sample size calculations

We can generate any number of contracts, but the essential question is: How many do we actually need? This depends on the desired precision for predicting average performance. After 15 contracts processed we got the following table 2

Table 2: Performance metrics of the contracts. In this table, the false positives are the entries that the LLM created. Wrong entries indicate entries that matched the template codes but not the values. Missing entries are those present in the ground truth but not provided by the LLM prediction. Accuracy is the percentage of correctly predicted entries by the LLM divided by the total number of entries.

Document	**False Positives**	**Wrong Entries**	**Missing Entries**	**Accuracy (%)**
Document 0	0	0	3	91.43
Document 1	0	0	0	100.00
Document 2	0	0	1	97.14
Document 3	1	0	1	97.14
Document 4	0	0	0	100.00
Document 5	0	0	1	97.14
Document 6	1	0	1	97.14
Document 7	0	0	0	100.00
Document 8	0	0	0	100.00
Document 9	0	0	0	100.00
Document 10	1	0	1	97.14
Document 11	0	1	3	88.57
Document 12	0	0	0	100.00
Document 13	0	0	0	100.00
Document 14	0	0	0	100.00
Average	0.2	0.067	0.73	97.71
Standard Deviation	0.41	0.26	1.03	3.45
95% Confidence Interval	0.00 - 0.43	0.00 - 0.21	0.16 - 1.31	95.80 - 99.62

A maximum permissible error of 1% at a 95% confidence level was chosen. After evaluating 15 contracts, the observed average accuracy was 97.71% with a standard deviation of 3.45%. The resulting error margin was calculated as follows:
To achieve a reduced error margin of 1%, we need to adjust the sample size. The sample size n required can be calculated using the formula:

$$n = \left(\frac{Z \times \sigma}{E} \right)^2 \tag{9}$$

where Z is the z-score corresponding to the 95% confidence level (1.96), σ is the standard deviation of the measurements, and E is the desired error margin. This calculation is based on the formula for estimating the minimum sample size necessary to achieve a specific margin of error in a proportion, given a confidence interval, which is derived from the central limit theorem [41]. Plugging in the values, we get:

$$n = \left(\frac{1.96 \times 3.45}{1} \right)^2$$

Solving for n, we find:

$$n = \left(\frac{6.76}{1} \right)^2 = 45.72 \tag{10}$$

Thus, approximately 46 contracts are required to achieve an error of 1% with 95% confidence, as shown in Equation 10. To ensure thoroughness, we generated 50 contracts.

5.1.2 Performance of the DORA Template Entry Filler

The performance of the extraction process was evaluated based on its ability to accurately identify and extract the required information for each field within the template. The results of processing 50 contracts with the model are presented in 3.

Table 3: Performance metrics of the contracts. False positives are entries incorrectly created by the LLM. Wrong entries are those that matched the template codes but not the values. Missing entries are those present in the ground truth but not predicted by the LLM. Accuracy is the percentage of correctly predicted entries by the LLM divided by the total number of entries.

Document	**False Positives**	**Wrong Entries**	**Missing Entries**	**Accuracy (%)**
Document 0	0	0	3	91.43
Document 1	0	0	0	100.00
Document 2	0	0	1	97.14
Document 3	1	0	1	97.14
Document 4	0	0	0	100.00
Document 5	0	0	1	97.14
Document 6	1	0	1	97.14
Document 7	0	0	0	100.00
Document 8	0	0	0	100.00
Document 9	0	0	0	100.00
Document 10	1	0	1	97.14
Document x	x	x	x	x
Document 40	0	0	1	97.14
Document 41	0	0	0	100.00
Document 42	0	0	0	100.00
Document 43	0	0	0	100.00
Document 44	0	0	1	97.14
Document 45	0	0	0	100.00
Document 46	0	0	1	97.14
Document 47	0	0	0	100.00
Document 48	0	0	0	100.00
Document 49	0	0	0	100.00
Average	0.1	0.04	0.76	97.71
Standard Deviation	0.36	0.20	1.10	3.32
95% Confidence Interval	0.00 - 0.20	0.00 - 0.10	0.45 - 1.07	96.77 - 98.66

The results indicate a high overall accuracy of 97.71%, with a standard deviation of 3.32%. The 95% confidence interval for accuracy ranges from 96.77% to 98.66%, demonstrating the reliable performance of the LLM in this extraction task. These findings showcase the potential of an LLM when the information is straightforward and easily identifiable.

5.2 Experiment 2: Extraction of Deliverables and Obligations

This experiment aimed to evaluate the performance of various Large Language Models (LLMs) in extracting contractual obligations. The models were assessed based on their accuracy and amount of false positives (FP)

5.2.1 Sample Size Calculations

Initially, all models were run for 15 iterations on each contract, yielding an average accuracy. A question arises: How many iterations are actually needed? The required number depends on the desired precision for predicting average performance. While an ideally large sample size would be preferable, practical constraints such as cost, environmental impact, and time limit feasibility. Our objective is to distinguish whether model X performs better than model Y. For meaningful comparison, it is essential that the confidence intervals (CIs) for the overall accuracies of the models do not overlap.
After conducting 15 iterations, the observed average accuracies are presented in Table 4.

Table 4: Performance Comparison of LLMs on Contract Obligations Extraction

Contract	Obligations	Iterations	Model	Avg FP	Avg Accuracy (%)	Std Dev (%)	95% CI (%)
BIOAMBERINC	12	15	GPT4o	16.10	77.22	3.81	[75.29, 79.15]
		15	Gemini	10.49	71.11	8.83	[66.64, 75.58]
		15	Claude	7.54	59.44	2.93	[57.96, 60.93]
CORALGOLD-RESOURCES,LTD	7	15	GPT4o	12.25	66.67	6.97	[63.14, 70.19]
		15	Gemini	5.17	60.00	12.31	[53.77, 66.23]
		15	Claude	4.94	71.43	0.00	[71.43, 71.43]
CytodynInc	27	15	GPT4o	16.71	44.94	7.39	[41.20, 48.68]
		15	Gemini	18.28	47.65	7.90	[43.66, 51.65]
		15	Claude	4.67	29.38	0.96	[28.90, 29.87]
EMERALDHEALTH THERAPEUTICSINC	9	15	GPT4o	12.90	75.56	9.58	[70.71, 80.40]
		15	Gemini	8.72	85.19	22.87	[73.61, 96.76]
		15	Claude	0	88.15	2.87	[86.70, 89.60]
SLINGERBAGINC	10	15	GPT4o	13.95	80.00	9.26	[75.31, 84.69]
		15	Gemini	9.27	66.00	9.10	[61.39, 70.61]
		15	Claude	6.00	57.33	4.58	[55.02, 59.65]
TRUENORTHENERGYCORP	13	15	GPT4o	14.22	78.97	6.80	[75.53, 82.41]
		15	Gemini	6.88	35.90	20.28	[25.63, 46.16]
		15	Claude	5.30	38.46	0.00	[38.46, 38.46]
Overall	-	-	GPT4o	14.37	70.56	8.26	[68.85, 72.27]
		-	Gemini	11.93	60.97	16.18	[57.63, 64.32]
		-	Claude	5.73	57.37	2.78	[56.79, 57.94]

The highest CI was from Gemini, causing it to overlap with Claude's CI. Currently, the standard error for Claude is 0.58%, and for Gemini, it is 5.08%. The difference in average accuracy is $60.97 - 57.37 = 3.6$. This implies that for the CIs to not overlap, the cumulative standard error must be at most 3.6. Considering Claude's already small standard error, attention shifts to Gemini.
The standard error needed from Gemini for the CIs to not overlap is calculated as:

$$\text{Required SE} = 3.6 - 0.58 = 3.02 \tag{11}$$

To determine the required sample size for Gemini, we use the formula:

$$n = \left(\frac{Z \times \sigma}{E} \right)^2 \tag{12}$$

where Z is the z-score corresponding to the 95% confidence level (1.96), σ is the standard deviation of the measurements (16.18), and E is the desired error margin (3.02). The formula used here also uses the central limit theorem to determine the smallest sample size needed to ensure a specified margin of error at a certain confidence level, as discussed in Lehmann (1999) [41]. Calculating the sample size, we find:

$$n = \left(\frac{1.96 \times 16.18}{3.02} \right)^2 \approx 110.26$$

When this value is divided by the number of contracts (6), the result is approximately 18.37. Rounding this up, we determine that 19 iterations are needed. To ensure thoroughness, we conducted 30 iterations of Gemini on all contracts.

5.2.2 Performance Comparison of LLMs on Contract Obligations Extraction

The table 5 provides a comparison of the performance of language models in extracting obligations from various contracts. The metrics include the number of obligations extracted, the number of iterations, average false positives (Avg FP), average accuracy (Avg Accuracy %), standard deviation (Std Dev %), and the 95% confidence interval (95% CI).

Table 5: Performance Comparison of LLMs on Contract Obligations Extraction

Contract	**Obligations**	**Iterations**	**Model**	**Avg FP**	**Avg Accuracy (%)**	**Std Dev (%)**	**95% CI (%)**
BIOAMBERINC	12	15	GPT4o	16.10	77.22	3.81	[75.29, 79.15]
		30	Gemini	10.11	64.44	20.17	[57.23, 71.66]
		15	Claude	7.54	59.44	2.93	[57.96, 60.93]
CORALGOLDRESOURCES,LTD	7	15	GPT4o	12.25	66.67	6.97	[63.14, 70.19]
		30	Gemini	5.23	60.95	13.49	[56.12, 65.78]
		15	Claude	4.94	71.43	0.00	[71.43, 71.43]
CytodynInc	27	15	GPT4o	16.71	44.94	7.39	[41.20, 48.68]
		30	Gemini	19.72	48.40	7.21	[45.82, 50.97]
		15	Claude	4.67	29.38	0.96	[28.90, 29.87]
EMERALDHEALTH	9	15	GPT4o	12.90	75.56	9.58	[70.71, 80.40]
THERAPEUTICSINC		30	Gemini	8.42	92.22	17.54	[85.94, 98.50]
		15	Claude	0	88.15	2.87	[86.70, 89.60]
SLINGERBAGINC	10	15	GPT4o	13.95	80.00	9.26	[75.31, 84.69]
		30	Gemini	9.72	69.67	9.99	[66.09, 73.24]
		15	Claude	6.00	57.33	4.58	[55.02, 59.65]
TRUENORTHENERGYCORP	13	15	GPT4o	14.22	78.97	6.80	[75.53, 82.41]
		30	Gemini	5.66	31.28	15.64	[25.69, 36.88]
		15	Claude	5.30	38.46	0.00	[38.46, 38.46]
Overall	-	-	GPT4o	14.37	70.56	8.26	[68.85, 72.27]
		-	Gemini	12.81	61.16	16.08	[58.81, 63.51]
		-	Claude	5.73	57.37	2.78	[56.79, 57.94]

The table reveals several insights:
Claude consistently showed the lowest average false positives, suggesting it is the most conservative model with fewer incorrect extractions. However, its accuracy varied, achieving the highest accuracy of 88.15% for Emerald Health Therapeutics Inc but only 29.38% for Cytodyn Inc.
Gemini exhibited moderate performance, with a higher number of false positives than Claude but fewer than GPT4o. Its accuracy was inconsistent, performing best with 92.22% accuracy for Emerald Health Therapeutics Inc but showing high variability as indicated by the standard deviation.
GPT4o demonstrated the highest average accuracy overall at 70.56%. Despite having the highest number of false positives, it maintained superior accuracy across most contracts, notably achieving 80.00% for Slingerbag Inc.
GPT4o achieves the highest accuracy but at the cost of more false positives, suggesting a trade-off between precision and recall. Claude's conservative approach results in fewer false positives but lower accuracy in some cases, while Gemini strikes a balance between the two but with greater variability in performance.
The sensitivity-specificity trade-off presents a central challenge in optimizing these models. Enhancing the model's sensitivity increases its ability to capture as many relevant obligations as possible but also raises the likelihood of false positives. On the other hand, increasing specificity minimizes false positives but at the risk of missing some obligations. This balance between sensitivity and specificity often comes down to subjective preference among legal professionals. Some may prioritize accuracy and are willing to manage more false positives, while others may prefer fewer interruptions by inaccuracies, even if it means potentially overlooking some obligations.

6 Discussion

This thesis explored the application of Large Language Models (LLMs) for automating the extraction of information from contracts. Two primary experiments were conducted: one focused on filling in the Digital Operational Resilience Act (DORA) compliance template, and the other on extracting obligations from contracts. The results from these experiments demonstrated varying levels of accuracy and highlighted the strengths and limitations of different LLMs in this context.

Experiment 1: DORA Compliance Template Extraction
The first experiment evaluated the ability of LLMs to extract information necessary for the DORA compliance template from contracts. The results indicated a high overall accuracy of 97.71%, with a 95% confidence interval ranging from 96.77% to 98.66%. This demonstrates the LLM's capability to perform well when the information is clearly identifiable and contextually straightforward. The low number of false positives and wrong entries further supports the reliability of the model in extracting precise information when guided by well-defined templates.

Experiment 2: Extraction of Obligations
The second experiment assessed the performance of LLMs in identifying and extracting obligations from contracts. This task proved more challenging due to the diverse and complex nature of contractual language. Among the models tested, GPT-4o achieved the highest average accuracy of 70.56%, though it also produced the most false positives. Claude 3.5 Sonnet showed the lowest number of false positives but had a lower overall accuracy. Gemini displayed moderate performance with higher variability.

The results revealed that while LLMs can effectively identify obligations, there remains a substantial variability in performance across different contracts and contract types.

6.1 Strengths

6.1.1 Strength of LLMs

The obligation extraction method has shown that Large Language Models (LLMs) possess a foundational ability to extract obligations from contractual documents, achieving a baseline accuracy of 70.56%. However, it is noteworthy that these results were achieved without any specific optimization of the models or the prompts used for extraction. This wasn't possible due to the limited dataset available—only six contracts were used, which constrained the scope for fine-tuning and prompt engineering. Optimizing a model or its prompts based on such a small dataset could lead to overfitting, where the model is tailored too closely to the dataset characteristics rather than learning to generalize from broader patterns.

Despite these constraints, the LLMs were able to achieve a commendable level of accuracy. This highlights the inherent potential of these models to perform even better with proper optimization. With access to a more extensive and varied dataset, it would be possible to fine-tune the models specifically for the task of obligation extraction and to refine the prompts to enhance their sensitivity and specificity. This tailored approach could significantly improve the accuracy and reliability of the models, making them more effective tools in legal document analysis.

6.1.2 Strength of the method

In our exploration of LLMs for extracting obligations from contracts and filling out DORA templates, the system demonstrated promising levels of accuracy. However, it is of consequence that there will still be a human in the loop. The role of the LLMs, in this case, is not to replace legal professionals but to augment their efficiency by pre-filling and highlighting identified obligations. This hybrid approach ensures that while the LLM handles the bulk of the initial data extraction, legal professionals maintain oversight to verify the accuracy and relevance of the information, providing a safety net that enhances reliability.
One of the advantages of using this LLM tool is the ability to meticulously parse every word of a contract, minimizing the risk of overlooking important but obscure obligations—a task that might challenge even experienced professionals due to human error or oversight fatigue. This dual-check system, where both human and machine collaborate, reduces the risk of consequential oversights.
The strengths of our approach to extracting and analyzing contract obligations with Large Language Models (LLMs) lie in several innovative techniques that enhance the accuracy and reliability of the system, particularly in legal applications. Firstly, the use of verbatim text as a linking identifier between predicted and ground truth obligations facilitates clear and objective comparisons. This method ensures that the evaluations of the LLM's output are based on exact text segments from the contracts, enhancing the reliability of automated systems in legal contexts, as you can now robustly quantify performance and train an algorithm on specific and accurate data.
Additionally, the approach of prompting the model to generate outputs in dictionary formatted strings enables the consistent parsing of stochastic string data into a structured, deterministic data frame. This technique ensures that the output remains organized and easy to analyze, despite the inherent variability in model predictions.
Moreover, the process of matching the predicted verbatim text and ground truth verbatim text to their corresponding locations in the contract, rather than directly to each other, introduces a robust layer of validation. This method effectively addresses scenarios where the starts of the annotated and predicted texts slightly differ but still refer to the same contractual obligation. By anchoring the text to specific locations within the contract, the likelihood of mismatching similar phrases that belong to different sections is reduced, thus enhancing the overall accuracy of the extraction process.

6.2 Limitations

6.2.1 Limitations of data

A notable limitation is the quality of datasets. For the DORA project, the information in the contracts was neither ambiguous nor well hidden, making the extraction task relatively straightforward. This raises questions about the LLM's efficacy in real-world scenarios where contract complexities and nuances play a larger role. Similarly, for the obligations extraction, annotating the data without being an authority on the subject compromises the reliability of the ground truth dataset. Furthermore, the small sample size of only six contracts, annotated by a non-expert, limits the generalizability of our claims.

6.2.2 Limitations of LLMs

Another challenge arises from the inherent variability in outputs caused by the stochastic nature of Large Language Models (LLMs) and the clarity of the prompts provided. Each time a prompt is issued, the LLM must discern the obligation importance cutoff, leading to inconsistencies in the

outputs. This variability is evident in the results, where the standard deviation significantly fluctuates by model, ranging from 2.78% to 16.08%.
The limited output token space causes truncation of responses. This typically occurs in cases involving longer documents with numerous obligations, where the model might not have enough space to include all relevant information. Such truncation can result in incomplete data capture, affecting the overall reliability of the information extracted by the LLM. These observations underscore the importance of tuning the model and managing its output space effectively to handle documents of various lengths and complexities without compromising the quality of the output.
Finally, the issue of generated false positives presents a challenge. The LLMs sometimes extracted text that was either not sufficiently important or not actually an obligation. The average number of false positives produced by each model ranged from 5.73 to 14.37 per contract. In some instances, particularly in contracts with a low count of actual obligations—such as those containing only 7 or 9—false positives outnumbered the correctly identified obligations.

6.3 Future Work

The exploration of Large Language Models (LLMs) for extracting contractual obligations and filling out compliance templates has demonstrated promising potential. However, several avenues for improvement and development remain open. This section outlines the steps that could be undertaken to refine and enhance the obligations extraction tool and its application.

6.3.1 Dataset Enhancement and Tool Improvement

A critical step forward would be addressing the dataset limitations encountered in the current study. To generate large volumes of accurate data, the Manage My Contract (MMC) tool used by Accenture should be modified. Instead of requiring legal professionals to fill in detailed forms, the tool could be repurposed into a highlighting utility. Legal professionals would simply read through a contract and highlight sections they identify as obligations. This approach would automatically capture verbatim texts of obligations directly from the contract, streamlining the data collection process.
Building on the improved data collection process, a secondary tool could be developed to extract and pre-fill secondary information for each highlighted obligation. This tool would use an LLM to predict secondary details (such as due dates, responsible parties, etc.) based on the context of the highlighted text. The predicted information would then be presented to legal professionals through a user interface, allowing them to quickly verify and adjust the pre-filled data. This system would reduce the manual workload while ensuring high data accuracy.

6.3.2 Model Training and Optimization

With a robust dataset in place, the next step would be to fine-tune an LLM specifically for the task of obligation extraction. This process would involve training the model on a set of prompts paired with the correct outputs, typically formatted in a JSONL file. Fine-tuning would help the model learn the nuances of contractual language and improve its ability to identify and extract relevant information accurately.
In parallel, an examination of common errors made by the LLM could reveal insights into its operational biases or weaknesses. Addressing these through targeted prompt engineering could further enhance the model's performance. By refining how the model is prompted, it is possible to guide its focus and improve its output accuracy.

6.3.3 Handling Output Limitations and Model Biases

One of the issues encountered was the cutoff of LLM outputs when dealing with long contracts. To mitigate this, the output token size could be dynamically adjusted based on the length of the contract, under the assumption that longer contracts contain more obligations. For exceptionally lengthy contracts, developing a method to segment the text without losing contextual integrity would be desirable. This approach also addresses the potential bias of LLMs towards generating outputs of a certain length, which can skew the balance between true positives and false positives.
Another innovative approach would involve instructing the LLM to not only output all detected obligations but also rank them based on perceived importance. By assigning a significance score to each obligation (e.g., on a scale from 1 to 100), the model provides an additional layer of analytical depth. Legal professionals could then set a threshold score, above which obligations are considered critical. This method increases the sensitivity of the extraction process, capturing a broader array of obligations at the risk of higher false positives, which can be controlled by adjusting the importance threshold.

6.4 Implications for Legal and Compliance Sectors

The findings of this thesis hold implications for automating tasks within the legal and compliance sectors. First, automating information extraction from contracts can substantially reduce the time and effort involved in contract management, resulting in notable efficiency gains. Second, employing automation as a supplementary verification method enhances the reliability of compliance processes, minimizing human error and ensuring strict adherence to regulatory standards. Finally, the deployment of large language models (LLMs) facilitates scalability across extensive volumes of contracts, delivering consistent analysis that would be impractical through manual efforts alone.
The future development of LLM-based tools for legal document analysis promises not only to enhance the accuracy and efficiency of obligation extraction but also to redefine how legal professionals interact with contractual data. By continuously refining the models, improving the data collection tools, and implementing innovative solutions to handle the complexities of legal texts, these tools can become indispensable assets in the legal industry. This iterative improvement process, fueled by advancements in machine learning and natural language processing, will drive the evolution of legal tech towards more intelligent, responsive, and reliable systems.

7 Conclusion

This thesis investigated the automation of information extraction from contracts using Large Language Models (LLMs), specifically focusing on how these models can be applied for legal document analysis. The research explored the effectiveness of state-of-the-art LLMs, such as GPT-4o, Claude 3.5 Sonnet, and Gemini 1.5 Pro, in parsing and extracting data from legal documents. The extraction tasks were designed to reflect practical applications used in legal workflows, including compliance with regulatory templates and identification of contractual obligations.

The capabilities of LLMs were demonstrated through accuracy levels in structured extraction tasks. The first experiment, which involved filling out the DORA compliance template, saw the models achieving an average accuracy of 97.71%, indicating that LLMs are highly capable of managing tasks where the required information is explicitly defined and accessible. In tasks involving the extraction of contractual obligations—a more complex challenge—the models displayed promising results, with GPT-4o achieving the highest accuracy at 70.56%. This variability in performance across different models and contract complexities underscores the potential and adaptability of LLMs in handling complex extraction tasks typical in legal settings.

However, the research also uncovered several limitations inherent to the current generation of LLMs. The output inconsistency, a byproduct of the stochastic nature of these models, poses a challenge as it can lead to variable results for identical inputs. This inconsistency can complicate the reliability of data extraction in legal processes where precision is paramount. Another limitation is the generation of false positives—instances where models incorrectly identify and extract irrelevant or inaccurately interpreted information. Such errors necessitate further checks and can complicate the use of extracted data in sensitive legal contexts. Finally, the constraints imposed by maximum token space limit the models' ability to return all information from longer documents, potentially leading to truncated or incomplete data.

Despite these challenges, this research paves the way for further refinement of LLM applications in legal contexts. With ongoing advancements in technology and methodologies, there is significant potential to enhance the reliability, accuracy, and utility of LLMs, making them indispensable tools in the automation of legal information extraction.

References

[1] Dogu Araci. "Finbert: Financial sentiment analysis with pre-trained language models". In: *arXiv preprint arXiv:1908.10063* (2019). DOI: `https://doi.org/10.48550/arXiv.1908.10063`.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *arXiv preprint arXiv:1409.0473* (2015). DOI: `https://doi.org/10.48550/arXiv.1409.0473`.

[3] Solon Barocas and Andrew D Selbst. "Big data's disparate impact". In: *Calif. L. Rev.* 104 (2016), page 671.

[4] Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. *Open LLM Leaderboard*. `https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard`. 2023.

[5] Rachel KE Bellamy et al. "AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias". In: *IBM Journal of Research and Development* 63.4/5 (2019), pages 4–1. DOI: `10.1147/JRD.2019.2942287`.

[6] Iz Beltagy, Kyle Lo, and Arman Cohan. "SciBERT: A pretrained language model for scientific text". In: *arXiv preprint arXiv:1903.10676* (2019).

[7] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. "On the dangers of stochastic parrots: Can language models be too big?" In: *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*. 2021, pages 610–623. DOI: `https://doi.org/10.1145/3442188.3445922`.

[8] Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. "Language (technology) is power: A critical survey of" bias" in nlp". In: *arXiv preprint arXiv:2005.14050* (2020). DOI: `https://doi.org/10.48550/arXiv.2005.14050`.

[9] Michael J Bommarito II, Daniel Martin Katz, and Eric M Detterman. "LexNLP: Natural language processing and information extraction for legal and regulatory texts". In: *Research handbook on big data law*. Edward Elgar Publishing, 2021, pages 216–227. DOI: `https://doi.org/10.4337/9781788972826.00017`.

[10] Michael Borenstein, Larry V. Hedges, Julian P. T. Higgins, and Hannah R. Rothstein. *Introduction to Meta-Analysis*. John Wiley & Sons, 2009.

[11] Thorsten Brants, Ashok Popat, Peng Xu, Franz Josef Och, and Jeffrey Dean. "Large language models in machine translation". In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. 2007, pages 858–867.

[12] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pages 1877–1901. DOI: `10.48550/arXiv.2005.14165`.

[13] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. "LEGAL-BERT: The muppets straight out of law school". In: *arXiv preprint arXiv:2010.02559* (2020). DOI: `https://doi.org/10.48550/arXiv.2010.02559`.

[14] Wei-Lin Chiang et al. "Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference". In: *arXiv preprint arXiv:2403.04132* (2024). DOI: `https://doi.org/10.48550/arXiv.2403.04132`.

[15] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Edited by Alessandro Moschitti, Bo Pang, and Walter Daelemans. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pages 1724–1734. DOI: 10.3115/v1/D14-1179. URL: https://aclanthology.org/D14-1179.

[16] Aakanksha Chowdhery et al. "Palm: Scaling language modeling with pathways". In: *Journal of Machine Learning Research* 24.240 (2023), pages 1–113.

[17] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. *Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge*. 2018. arXiv: 1803.05457 [cs.AI].

[18] Karl Cobbe et al. *Training Verifiers to Solve Math Word Problems*. 2021. arXiv: 2110.14168 [cs.CL].

[19] Abhishek Dasgupta and Steven Wendler. "AI adoption strategies". In: *Retrieved September* 18 (2019), page 2022.

[20] Alex de Vries. "The growing energy footprint of artificial intelligence". In: *Joule* 7.10 (2023), pages 2191–2194. ISSN: 2542-4351. DOI: https://doi.org/10.1016/j.joule.2023.09.004. URL: https://www.sciencedirect.com/science/article/pii/S2542435123003653.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018). DOI: https://doi.org/10.48550/arXiv.1810.04805.

[22] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. "Fairness through awareness". In: *Proceedings of the 3rd innovations in theoretical computer science conference*. 2012, pages 214–226. DOI: https://doi.org/10.1145/2090236.2090255.

[23] European Commission. *Digital Operational Resilience Act*. https://www.eiopa.europa.eu/digital-operational-resilience-act-dora_en.

[24] Luciano Floridi and Massimo Chiriatti. "GPT-3: Its nature, scope, limits, and consequences". In: *Minds and Machines* 30.4 (2020), pages 681–694. DOI: 10.1007/s11023-020-09548-1.

[25] Leo Gao et al. *A framework for few-shot language model evaluation*. Version v0.0.1. Sept. 2021. DOI: 10.5281/zenodo.5371628. URL: https://doi.org/10.5281/zenodo.5371628.

[26] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[27] Bryce Goodman and Seth Flaxman. "European Union regulations on algorithmic decision-making and a "right to explanation"". In: *AI magazine* 38.3 (2017), pages 50–57. DOI: https://doi.org/10.1609/aimag.v38i3.2741.

[28] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. *Large Language Model based Multi-Agents: A Survey of Progress and Challenges*. 2024. arXiv: 2402.01680 [cs.CL]. URL: https://arxiv.org/abs/2402.01680.

[29] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. "Don't stop pretraining: Adapt language models to domains and tasks". In: *arXiv preprint arXiv:2004.10964* (2020). DOI: https://doi.org/10.48550/arXiv.2004.10964.

[30] Thilo Hagendorff. "The ethics of AI ethics: An evaluation of guidelines". In: *Minds and machines* 30.1 (2020), pages 99–120. DOI: https://doi.org/10.1007/s11023-020-09517-8.

[31] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. *Measuring Massive Multitask Language Understanding*. 2021. arXiv: 2009.03300 [cs.CY].

[32] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pages 1735–1780.

[33] Jeremy Howard and Sebastian Ruder. *Universal Language Model Fine-tuning for Text Classification*. cite arxiv:1801.06146Comment: ACL 2018, fixed denominator in Equation 3, line 3. 2018. URL: http://arxiv.org/abs/1801.06146.

[34] Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. "Clinicalbert: Modeling clinical notes and predicting hospital readmission". In: *arXiv preprint arXiv:1904.05342* (2019). DOI: https://doi.org/10.48550/arXiv.1908.10063.

[35] Anna Jobin, Marcello Ienca, and Effy Vayena. "The global landscape of AI ethics guidelines". In: *Nature machine intelligence* 1.9 (2019), pages 389–399. DOI: https://doi.org/10.1038/s42256-019-0088-2.

[36] Jared Kaplan et al. "Scaling laws for neural language models". In: *arXiv preprint arXiv:2001.08361* (2020). DOI: https://doi.org/10.48550/arXiv.2001.08361.

[37] Joshua A Kroll. *Accountability in computer systems*. Oxford University Press New York, 2020.

[38] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pages 436–444. DOI: 10.1038/nature14539.

[39] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. "BioBERT: a pre-trained biomedical language representation model for biomedical text mining". In: *Bioinformatics* 36.4 (2020), pages 1234–1240.

[40] Natalie Leesakul, Anne-Marie Oostveen, Iveta Eimontaite, Max L. Wilson, and Richard Hyde. "Workplace 4.0: Exploring the Implications of Technology Adoption in Digital Manufacturing on a Sustainable Workforce". In: *Sustainability* 14.6 (2022). ISSN: 2071-1050. DOI: 10.3390/su14063311. URL: https://www.mdpi.com/2071-1050/14/6/3311.

[41] E. L. Lehmann. *Elements of Large-Sample Theory*. Springer New York, 1999.

[42] Stephanie Lin, Jacob Hilton, and Owain Evans. *TruthfulQA: Measuring How Models Mimic Human Falsehoods*. 2022. arXiv: 2109.07958 [cs.CL].

[43] Yinhan Liu et al. "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692* (2019). DOI: https://doi.org/10.48550/arXiv.1907.11692.

[44] Gary Marcus and Ernest Davis. *Rebooting AI: Building artificial intelligence we can trust*. Vintage, 2019. DOI: https://doi.org/10.48550/arXiv.2005.14050.

[45] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. "Recurrent neural network based language model." In: *Interspeech*. Volume 2. 3. Makuhari. 2010, pages 1045–1048.

[46] Seoungkwon Min and Boyoung Kim. “Adopting Artificial Intelligence Technology for Network Operations in Digital Transformation”. In: *Administrative Sciences* 14.4 (2024). ISSN: 2076-3387. DOI: 10.3390/admsci14040070. URL: https://www.mdpi.com/2076-3387/14/4/70.

[47] Brent Daniel Mittelstadt, Patrick Allo, Mariarosaria Taddeo, Sandra Wachter, and Luciano Floridi. “The ethics of algorithms: Mapping the debate”. In: *Big Data & Society* 3.2 (2016), page 2053951716679679. DOI: https://doi.org/10.1177/205395171667967.

[48] Philipp Mondorf and Barbara Plank. *Beyond Accuracy: Evaluating the Reasoning Behavior of Large Language Models – A Survey*. 2024. arXiv: 2404.01869 [cs.CL]. URL: https://arxiv.org/abs/2404.01869.

[49] Minh-Tien Nguyen, Duy-Hung Nguyen, Shahab Sabahi, Hung Le, Jeff Yang, and Hajime Hotta. “When Giant Language Brains Just Aren’t Enough! Domain Pizzazz with Knowledge Sparkle Dust”. In: (May 2023). DOI: https://doi.org/10.48550/arXiv.2305.07230.

[50] Safiya Umoja Noble. “Algorithms of oppression: How search engines reinforce racism”. In: *Algorithms of oppression*. New York university press, 2018. DOI: https://doi.org/10.18574/nyu/9781479833641.001.0001.

[51] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. “Improving language understanding by generative pre-training”. In: (2018).

[52] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8 (2019), page 9.

[53] Jack W Rae et al. “Scaling language models: Methods, analysis & insights from training gopher”. In: *arXiv preprint arXiv:2112.11446* (2021). DOI: https://doi.org/10.48550/arXiv.2112.11446.

[54] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *Nature* 323.6088 (1986), pages 533–536. DOI: https://doi.org/10.1038/323533a0.

[55] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. *WINOGRANDE: An Adversarial Winograd Schema Challenge at Scale*. 2019. arXiv: 1907.10641 [cs.CL].

[56] Jürgen Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural Networks* 61 (2015), pages 85–117. DOI: https://doi.org/10.1016/j.neunet.2014.09.003.

[57] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. “Green ai”. In: *Communications of the ACM* 63.12 (2020), pages 54–63. DOI: https://doi.org/10.1145/3381831.

[58] Claude Elwood Shannon and Warren Weaver. *The mathematical theory of communication, by Claude E. Shannon and Warren Weaver*. University of illinois Press, 1949.

[59] Julia Siderska. “The Adoption of Robotic Process Automation Technology to Ensure Business Processes during the COVID-19 Pandemic”. In: *Sustainability* 13.14 (2021). ISSN: 2071-1050. DOI: 10.3390/su13148020. URL: https://www.mdpi.com/2071-1050/13/14/8020.

[60] Shaden Smith et al. “Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model”. In: *arXiv preprint arXiv:2201.11990* (2022).

[61] Emma Strubell, Ananya Ganesh, and Andrew McCallum. “Energy and policy considerations for deep learning in NLP”. In: *arXiv preprint arXiv:1906.02243* (2019). DOI: https://doi.org/10.48550/arXiv.1906.02243.

[62] Ilya Sutskever, James Martens, and Geoffrey E Hinton. “Generating text with recurrent neural networks”. In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011, pages 1017–1024.

[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017). DOI: `https://doi.org/10.48550/arXiv.1706.03762`.

[64] Paul Voigt and Axel Von dem Bussche. “The eu general data protection regulation (gdpr)”. In: *A Practical Guide, 1st Ed., Cham: Springer International Publishing* 10.3152676 (2017), pages 10–5555.

[65] Carole-Jean Wu et al. *Sustainable AI: Environmental Implications, Challenges and Opportunities*. 2022. arXiv: `2111.00364 [cs.LG]`. URL: `https://arxiv.org/abs/2111.00364`.

[66] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. “Bloomberggpt: A large language model for finance”. In: *arXiv preprint arXiv:2303.17564* (2023). DOI: `https://doi.org/10.48550/arXiv.2303.17564`.

[67] Wonjin Yoon, Jinhyuk Lee, Donghyeon Kim, Minbyul Jeong, and Jaewoo Kang. “Pre-trained language model for biomedical question answering”. In: *Joint European conference on machine learning and knowledge discovery in databases*. Springer. 2019, pages 727–740.

[68] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. *HellaSwag: Can a Machine Really Finish Your Sentence?* 2019. arXiv: `1905.07830 [cs.CL]`.

[69] Lucia Zheng, Neel Guha, Brandon R. Anderson, Peter Henderson, and Daniel E. Ho. *When Does Pretraining Help? Assessing Self-Supervised Learning for Law and the CaseHOLD Dataset*. 2021. DOI: `https://doi.org/10.48550/arXiv.2104.08671`. arXiv: `2104.08671 [cs.CL]`.

[70] Shoshana Zuboff. “The age of surveillance capitalism”. In: *Social theory re-wired*. Routledge, 2023, pages 203–213. DOI: `https://doi.org/10.48550/arXiv.2005.14050`.

Appendices

Table 6: Performance metrics of the contracts (Part 1). False positives are entries incorrectly created by the LLM. Wrong entries are those that matched the template codes but not the values. Missing entries are those present in the ground truth but not predicted by the LLM. Accuracy is the percentage of correctly predicted entries by the LLM divided by the total number of entries.

Document	**False Positives**	**Wrong Entries**	**Missing Entries**	**Accuracy (%)**
Document 0	0	0	3	91.43
Document 1	0	0	0	100.00
Document 2	0	0	1	97.14
Document 3	1	0	1	97.14
Document 4	0	0	0	100.00
Document 5	0	0	1	97.14
Document 6	1	0	1	97.14
Document 7	0	0	0	100.00
Document 8	0	0	0	100.00
Document 9	0	0	0	100.00
Document 10	1	0	1	97.14
Document 11	0	1	3	88.57
Document 12	0	0	0	100.00
Document 13	0	0	0	100.00
Document 14	0	0	0	100.00
Document 15	0	0	1	97.14
Document 16	0	0	0	100.00
Document 17	0	0	0	100.00
Document 18	0	0	1	97.14
Document 19	0	0	2	94.29
Document 20	0	0	2	94.29
Document 21	0	0	0	100.00
Document 22	0	0	1	97.14
Document 23	0	0	0	100.00
Document 24	0	0	0	100.00
Document 25	0	0	1	97.14
Document 26	0	0	0	100.00
Document 27	0	0	1	97.14
Document 28	0	0	0	100.00
Document 29	2	0	2	94.29

Table 7: Performance metrics of the contracts (Part 2). False positives are entries incorrectly created by the LLM. Wrong entries are those that matched the template codes but not the values. Missing entries are those present in the ground truth but not predicted by the LLM. Accuracy is the percentage of correctly predicted entries by the LLM divided by the total number of entries.

Document	**False Positives**	**Wrong Entries**	**Missing Entries**	**Accuracy (%)**
Document 30	0	0	1	97.14
Document 31	0	1	1	94.29
Document 32	0	0	0	100.00
Document 33	0	0	1	97.14
Document 34	0	0	4	88.57
Document 35	0	0	0	100.00
Document 36	0	0	0	100.00
Document 37	0	0	5	85.71
Document 38	0	0	0	100.00
Document 39	0	0	1	97.14
Document 40	0	0	1	97.14
Document 41	0	0	0	100.00
Document 42	0	0	0	100.00
Document 43	0	0	0	100.00
Document 44	0	0	1	97.14
Document 45	0	0	0	100.00
Document 46	0	0	1	97.14
Document 47	0	0	0	100.00
Document 48	0	0	0	100.00
Document 49	0	0	0	100.00
Average	0.1	0.04	0.76	97.71
Standard Deviation	0.36	0.20	1.10	3.32
95% Confidence Interval	0.00 - 0.20	0.00 - 0.10	0.45 - 1.07	96.77 - 98.66

Table 8: Detailed Performance Metrics of Documents

Document	**False Positives**	**No-Match**	**Missing**
Document 0		Truth: No — Output: Commercial Bank	Type of the entity maintaining the register of information
Document 1		Truth: 75 days — Output: 150 days, Truth: 150 days — Output: 75 days	
Document 2			Currency, Nature of the financial entity
Document 3			Notice period for the financial entity, Type of the entity maintaining the register of information, Notice period for the ICT third-party service provider
Document 8			Notice period for the financial entity, Notice period for the ICT third-party service provider
Document 10			Notice period for the financial entity, Type of the entity maintaining the register of information, Notice period for the ICT third-party service provider
Document 14			Type of the entity maintaining the register of information
Document 15			Type of the entity maintaining the register of information
Document 16			Currency, Type of the entity maintaining the register of information
Document 17			Currency, Type of the entity maintaining the register of information, Contractual arrangement reference number
Document 19			Currency
Document 20			Type of the entity maintaining the register of information
Document 22			Country of issuance of the other code to identify the ICT third-party service provider, Recovery time objective of the function, Other code to identify the ICT third-party service provider, Type of the entity maintaining the register of information
Document 23			Notice period for the financial entity, Notice period for the ICT third-party service provider
Document 25			Type of the entity maintaining the register of information
Document 26	Country of issuance of the other code to identify the ICT third-party service provider, Other code to identify the ICT third-party service provider		Currency, Country of issuance of the other code to identify the ICT third-party service provider, Other code to identify the ICT third-party service provider, Contractual arrangement reference number