Large Language Models Assisted Contract Review

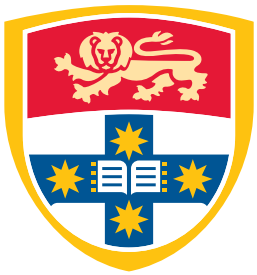
YUXIANG HUANG
SID: 490174780

Supervisor: Dr. Ying Zhou

This thesis is submitted in partial fulfillment of
the requirements for the degree of
Bachelor of Advanced Computing (Honours)

School of Computer Science
The University of Sydney
Australia

18 June 2023



THE UNIVERSITY OF
SYDNEY

Student Plagiarism: Compliance Statement

I certify that:

I have read and understood the University of Sydney Student Plagiarism: Coursework Policy and Procedure;

I understand that failure to comply with the Student Plagiarism: Coursework Policy and Procedure can lead to the University commencing proceedings against me for potential student misconduct under Chapter 8 of the University of Sydney By-Law 1999 (as amended);

This Work is substantially my own, and to the extent that any part of this Work is not my own I have indicated that it is not my own by Acknowledging the Source of that part or those parts of the Work.

Name: Yuxiang Huang

Signature: *Yuxiang Huang* **Date**: 18 June 2023

Abstract

Contract review, defined as the systematic examination of a contract's terms, clauses, and stipulations, is a crucial component of any legal process which ensures all parties involved in a transaction understand their rights, obligations, and the consequences of non-compliance. Nonetheless, owing to the specialized knowledge required to comprehend and interpret legal documents, traditional contract review processes frequently prove to be costly and time-intensive. This has resulted in considerable expenditure of financial resources and time for both corporations and individuals over the years.

In our research, we explored machine learning methodologies to assist real-life legal contract review tasks. Acknowledging the limitations of current AI-based review methods with BERT-based models, we turned to Large Language Models (LLMs) for better accuracy and efficiency. Specifically, we chose GPT-3.5-Turbo model from OpenAI for information extraction and question answering tasks, both fundamental components of AI-driven contract review. We designed multiple prompts tailored to our tasks, enabling the model to better understand the context and generate more desirable responses.

Our methods were validated on the Contract Understanding Atticus Dataset (CUAD), an Expert-Annotated NLP Dataset specifically designed for Legal Contract Review. The results demonstrated that contract review tasks processed by GPT-based model exhibited greater precision and efficiency compared to baseline methods, including prevalent BERT-based models such as ALBERT and DeBERTa. These findings highlight the immense potential of LLMs in contract review and similar applications.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Dr. Ying Zhou, for offering me the opportunity to undertake this research project and providing meticulous and patient guidance throughtout the research period. Her professional knowledge and critical insights also have greatly facilitated my research journey.

I would also like to acknowledge my parents. Their support throughout this year's honor study has helped me overcome challenges and remain focused on my goals.

Thank you all for playing such a significant role in my academic journey.

CONTENTS

Student Plagiarism: Compliance Statement ii

Abstract iii

Acknowledgements iv

List of Figures vii

List of Tables ix

Chapter 1 Introduction 1

1.1 Background and Motivation 1

1.2 Current State and Challenges 3

1.3 Innovative Technologies 3

1.4 Contributions 4

1.5 Thesis Outline 5

Chapter 2 Literature Review 6

2.1 AI Technology in Legal Contracting 6

2.2 Transformer and BERT 7

2.3 Large Language Models and GPT 10

2.4 Prompt Engineering 13

2.5 Prompt Tuning 16

2.6 Legal Document Datasets 18

Chapter 3 Methods 20

3.1 Task Definition 20

3.2 Implementation of LLMs 21

3.2.1 Model Selection 21

3.2.2 Building GPT-based Application 21

3.3 Prompting Methods 22

3.3.1 Zero-Shot Prompting . . . 22
3.3.2 One-Shot Prompting . . . 23
3.3.3 Few-Shot Prompting . . . 23
3.4 Input Processing . . . 24
3.4.1 Dataset Structure . . . 24
3.4.2 Text Segmentation . . . 25
3.5 Output Validation . . . 26
3.5.1 Answer Extraction . . . 27
3.5.2 Textual Similarity . . . 27

Chapter 4 Experiments and Results **28**
4.1 Experiment Setup . . . 28
4.1.1 Implementation Details . . . 28
4.1.2 Evaluation Metrics . . . 29
4.1.3 Datasets . . . 30
4.1.4 Initial Prompt Design . . . 30
4.2 Prompt Tuning . . . 31
4.2.1 Output Format . . . 31
4.2.2 General Task Instructions . . . 32
4.2.3 Label-specific Rules . . . 32
4.2.4 Examples in Prompts . . . 34
4.3 Overall Performance . . . 35
4.4 Performance by Category . . . 37
4.5 Error Analysis . . . 42

Chapter 5 Discussion **46**
5.1 Limitations . . . 46
5.2 Future Work . . . 47

Chapter 6 Conclusion **48**

Bibliography **49**

List of Figures

1.1 A bar chart displaying the hourly rates of attorneys, categorized by different document types they handle. [14] 2

1.2 The parameter size of GPT 3 and some BERT-based models. [27] 4

2.1 The Transformer architecture 8

2.2 The pre-training and fine-tuning process in BERT [7] 9

2.3 A summary of pre-trained language models [20] 10

2.4 Original GPT model Architecture [22] 11

2.5 The parameter size of GPT 3 and some BERT-based models. [27] 12

2.6 GPT-3 Performance on TriviaQA [11]. One-shot and few-shot performance make significant gains over zero-shot behavior[2] 16

2.7 An example of Chain-of-Thought prompting. [29] 16

2.8 Performance of BERT-based models on CUAD. Each bar is an average of the performance of all models in each model class. [9] 19

3.1 An example of the automated annotation process from CUAD paper: The model takes a contract as the input and outputs relevant clauses for each label [9] 21

3.2 Our work flow with LangChain, explaining the process from query submission to receiving model output 22

3.3 Distribution of Document Lengths in the CUAD Dataset: Each contract has been tokenized using the GPT tokenizer. The histogram illustrates that while most contracts contain fewer than 5,000 tokens, many of them extend to 20,000 tokens or more. 25

4.1 Performance on labels with poor performance by different prompting methods 41

4.2 Performance of precision and recall by label. The last three labels have a precision and recall of 0 because there's no relevant clause to them in the test dataset according to the ground truth. We do not consider them here. 45

List of Tables

4.1 Overall performance of GPT-3.5 model based on different prompting techniques 35

4.2 Overall performance of BERT-based models on CUAD datasets. All models has been trained and fine-tuned by the authors of CUAD. 36

4.3 Precision across different label categories under zero-shot, one-shot and few-shot prompting. The highest value for each label is bold, indicating the best performance achieved 38

4.4 Recall across different label categories under zero-shot, one-shot and few-shot prompting. 39

4.5 F1-score across different label categories under zero-shot, one-shot and few-shot prompting. 40

4.6 Results for the label category 'Most Favored Nation' 42

4.7 Results for the label category 'Joint IP Ownership' 43

4.8 Results for the label category 'Warranty Duration' 43

4.9 Results for the label category 'Affliate IP License-Licensor' 44

CHAPTER 1

Introduction

In this chapter, we present a detailed introduction to our research background and motivations. We begin with the concept of contract review and the shortcomings of the traditional contract review process, followed by the introduction of prevalent AI-based contract review methods and their constraints. Subsequently, we introduce the latest Large Language Models, highlighting their exceptional performance on various NLP tasks and their significant potential in the area of AI-based contract review.

Following this, we provide a statement of our research's main contributions. To conclude the chapter, we offer an overview of the structure of this thesis to facilitate a better understanding of the ensuing content.

1.1 Background and Motivation

Contract review plays a pivotal role in the realm of legal processes: It typically involves a multi-step process, including a comprehensive reading of the entire contract, a close examination of relevant clauses and terms for clarity and unambiguity, and a meticulous confirmation of the contract's compliance with applicable laws, regulations and industry standards. As a preventive measure, contract review effectively identifies and mitigates potential legal risks. In essence, contract review is the cornerstone of fair and effective legal transactions, safeguarding the interests of all involved parties.

Due to the complex structure and diverse contextual backgrounds of legal contracts, the traditional contract review process is a labor-intensive and time-consuming task requiring expert knowledge from legal professionals. As of 2023, the average hourly rate for retaining an attorney for such tasks ranges from $200 to $500 (as shown in Figure 1.1), and it can be even higher for experienced attorneys. Meanwhile, a short contract might take only an hour or two to review, but more complex contracts such as large business agreements or contracts in highly regulated industries could demand many hours or even days to

review thoroughly. These factors together contribute to the substantial expense associated with contract review.

Consequently, contract review not only costs large corporations considerable amounts every year, but also prevents small companies and individuals from thorough and reliable contract review and thereby considerably increases the legal risks they face. Therefore, there's a critical need for more accessible and efficient contract review processes.

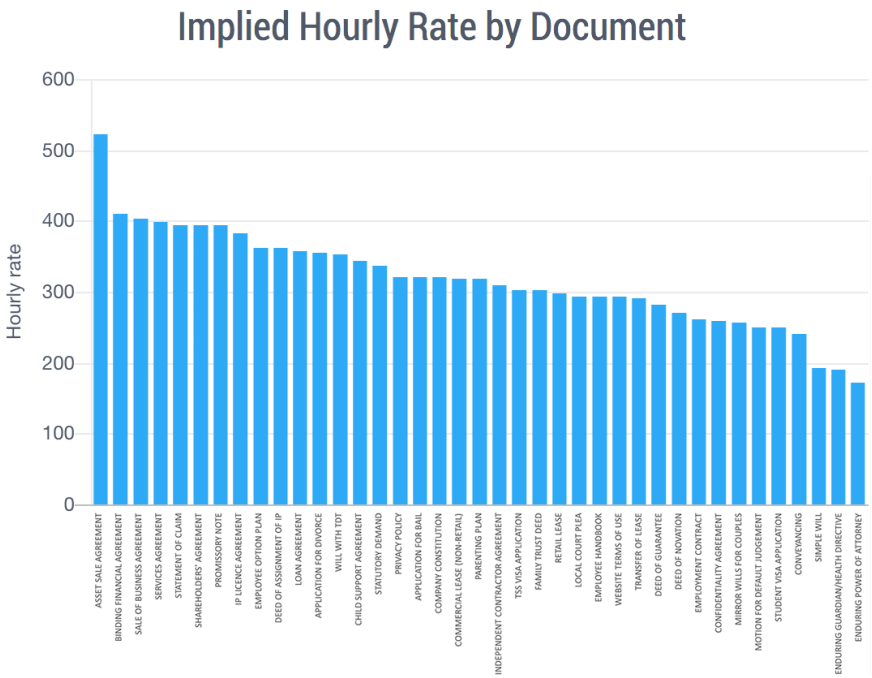


FIGURE 1.1: A bar chart displaying the hourly rates of attorneys, categorized by different document types they handle. [14]

1.2 Current State and Challenges

In recent years, AI-based contract review solutions have became an innovative alternative to the traditional manual process. By leveraging natural language processing (NLP) and machine learning technologies, these AI tools automate the review process by interpreting contract content, identifying key clauses and flagging potential issues. The rise of AI-based contract review has significantly increased efficiency, reduced costs, and minimized the risk of human error, marking a considerable improvement over traditional labor-intensive approaches.

At present, BERT-based models have been widely used in AI-based contract review solutions. As a pre-trained model developed by Google, BERT can understand the context of words in a sentence by analyzing the sentence in both directions, making it particularly powerful for understanding complex legal language.

Despite their effectiveness, BERT-based models have shown certain limitations in performing contract review tasks. They require extensive computational resources, and their maximum sequence length of 512 tokens falls short for long legal contracts. In addition, BERT requires fine-tuning with domain-specific data to achieve high performance, which sets up obstacles for developers.

To summarize, the limitations of BERT-based models have driven our research for a more effective and efficient alternative for AI-based contract review.

1.3 Innovative Technologies

Large Language Models (LLMs) have emerged as a new frontier in the field of Natural Language Processing (NLP). Owing to their significantly larger model size, LLMs are capable of superior performance compared to earlier BERT-based models. The latest LLMs, such as the GPT (Generative Pretrained Transformer) series developed by OpenAI and the PaLM (Pathways Language Model) developed by Google, have achieved state-of-the-art performance on a range of benchmark tasks, including Text Classification and Question Answering. Among them, as of March 2023, GPT-3 models are the top performers.

Unlike BERT-based models, which require fine-tuning on a large labelled dataset, GPT models can perform tasks in a zero-shot, one-shot, or few-shot setting. By using prompts that guide the model's

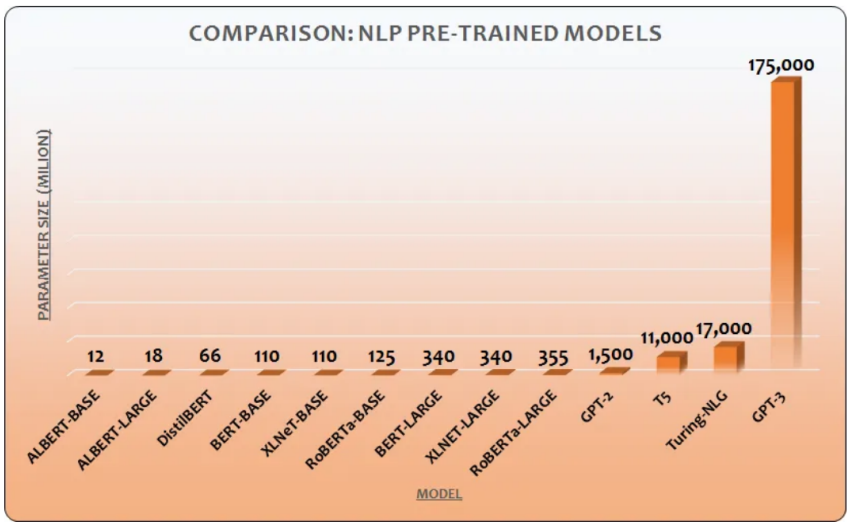


FIGURE 1.2: The parameter size of GPT 3 and some BERT-based models. [27]

predictions, they reduce the need for task-specific fine-tuning and large labeled datasets. Trained on a diverse range of internet text, GPT models are able to understand and generate text based on a wider range of contexts without needing task-specific training data. Furthermore, GPT models can handle longer sequences of text, which is critical for tasks like contract review where longer documents are input.

In conclusion, GPT models have shown remarkable potential in the area of contract review. Given their advantages over BERT, GPT models may eventually replace BERT-based models in the field of AI-based contract review. However, as GPT models function based on prompt-based learning, as opposed to BERT-based models which rely on training and fine-tuning, crafting effective prompts that enable GPT models to accurately comprehend task content and adapt to complex legal text remains a critical challenge.

1.4 Contributions

The key contributions from our study include:

(1) Implementation of zero-shot, one-shot, and few-shot prompting techniques with GPT-3.5-Turbo. Our results indicate that our approach outperforms prevalent BERT-based models on contract review tasks.

(2) Iterative refinement and design of few-shot prompts for the GPT-3.5 model. This prompt tuning method achieved superior performance compared to DeBERTa-xlarge, especially in identifying and extracting certain labels that proved challenging for BERT models.

1.5 Thesis Outline

In Chapter 2, we present the literature review regarding the current progress on topics related to our research, including Transformer and BERT, Large Language Models and GPT, prompt engineering and legal document datasets.

In Chapter 3, we go through the methods we used in the research. including prompting methods, input processing, answer extracting from GPT output and textual similarity used when compared with ground truth from dataset.

In chapter 4, we state our experimental designs: implementation details; evaluation benchmarks; prompt tuning; We then analyze the result from experiments, validating our method with the benchmark (BERT-based models). We analyze the error regarding the identification of several label with low recall.

In chapter 5, we point out the limitation of our method and discuss future research direction.

In chapter 6, we present the conclusion of this thesis.

CHAPTER 2

Literature Review

This chapter provides a comprehensive review of the relevant literature related to our research. We start with a review of AI technology in legal contracting, followed by an exploration of foundational technologies shaping our research: BERT-based models and Large Language Models, including their applications in legal document analysis. We then focus on prompt engineering, a critical technique for effectively utilizing Large Language Models. Finally, we explore legal document dataset for conducting our research, which is a critical component in the training and fine-tuning of AI models. In conclusion, this review forms the basis for our study, providing a contextual background for our methodology, experiments and evaluation.

2.1 AI Technology in Legal Contracting

Artificial Intelligence (AI) has induced a significant shift in the legal field, especially in the domain of contracting. Contract AI employs machine learning methodologies for managing, extracting, and reviewing legal contracts. With its ability to identify patterns, flag specific behaviors posing potential risks, and convert unstructured data into structured ones, Contract AI has proven effective in assisting various contracting tasks such as drafting, reviewing, and extracting critical data from contracts. By automating and simplifying these processes, Contract AI significantly improves efficiency, accuracy, and speed while minimizing potential human error. [12]

Moreover, AI technology also aids in data collection for practitioners. Possessing the capacity to analyze and learn from vast amounts of contract data, Contract AI provides predictive insights to practitioners. Additionally, its ability to process a substantial number of contracts allows for the establishment of a comprehensive database of contract information that facilitates life cycle management, risk analysis, and strategic decision-making, making Contract AI an invaluable asset in legal contracting. [5]

In terms of contract review, there has been a various AI contract review tools, such as LawGeex, Leah and Kira. Current AI-based contract review solutions tend to leverage a mix of traditional machine learning techniques and more advanced methodologies like transformer-based models, such as BERT. Seonghyeon et al. (2022) [18] introduced a model for detecting contractual risk categories in each clause using BERT. With an accuracy of 0.889 for validation and a 0.934 F1 score on testing, the BERT-based clause classification model outperforms other machine learning methods like the support vector machine and a simple deep neural network in every risk category. The ability of the model to classify clauses based on risk categories can simplify the review process and creates opportunities for automation in the review process. The reviewers can focus on risky clauses and therefore significantly mitigate risks during construction projects.

However, despite their strengths, current AI-based contract review methods exhibit certain limitations. Due to the architecture of BERT, it requires a lot of computational resources for training and predicting, making current AI contract review tools unaffordable for many startups and scaleups [12]. Beside, as the grow of emerging new language models, there's plenty of room for improving efficiency and accuracy in utilizing models for contract review tasks.

In conclusion, the limitation of BERT-based models have driven our research for a more effective and efficient alternative for AI-based contract review. Detailed exploration of these issues and potential solutions will be covered in the following sections.

2.2 Transformer and BERT

In the realm of natural language processing (NLP), Recurrent Neural Networks (RNN) held a position of prominence for years. They have been widely used in various supervised NLP tasks such as classification and regression. This success was largely attributed to architectures like Long Short-Term Memory (LSTM) [10] and Gated Recurrent Unit (GRU) [6]. Despite their successes, these models encountered several limitations, most notably their inherent difficulty in parallelization due to their recurrent structure, and struggles with handling long sequences owing to the vanishing gradient problem.

In an attempt to overcome these hurdles, Vaswani et al. (2017) introduced the Transformer [28]. This architecture steered clear of recurrent patterns and instead relied exclusively on attention mechanisms for sequence processing. As a result, Transformers don't suffer from the vanishing gradient problem and are readily parallelizable, thus facilitating the training of broader networks and significantly accelerating

computational performance (Anthony et al., 2020). This breakthrough paves the way for the advent of large language models and indicates a new era in NLP.

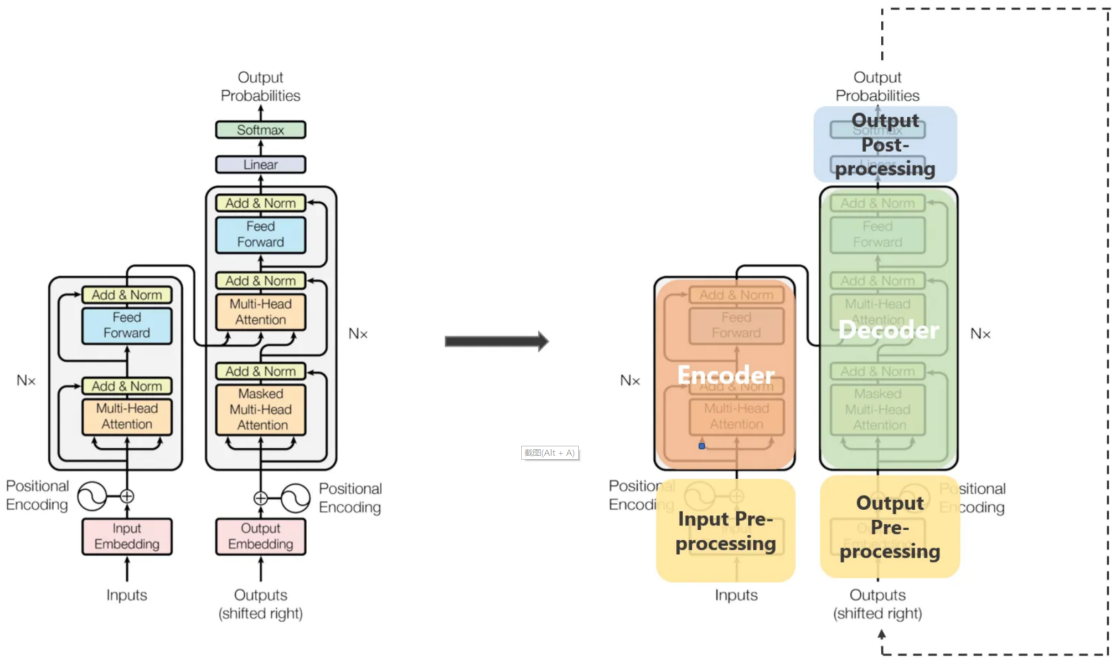


FIGURE 2.1: The Transformer architecture

Bidirectional Encoder Representations from Transformers (BERT) [7], developed by researchers at Google AI Language and open-sourced in 2018, is a state-of-the-art machine learning model for natural language processing tasks.

Based on the Transformer architecture, BERT extends the Transformer's capabilities by processing language bidirectionally. While previous models read the text input sequentially (left-to-right or right-to-left), BERT is bidirectional and processes the whole sentence at once. This bidirectional approach allows BERT to better understand the meaning and role of each word in its context. In addition, BERT leverages a two-step process: pre-training and fine-tuning. BERT is pre-trained on a large corpus of text data (Wikipedia and BooksCorpus) using two unsupervised learning tasks: Masked Language Modeling and Next Sentence Prediction. This provides BERT with a good understanding of language semantics. During fine-tuning phase, BERT is further trained on a specific task using labelled data, allowing it to adapt to a variety of NLP tasks with relatively minimal changes to the model architecture.

Furthermore, BERT's open-source nature allows the development of numerous variants and adaptations for specific use-cases and languages: One notable derivative of BERT is RoBERTa [16], introduced

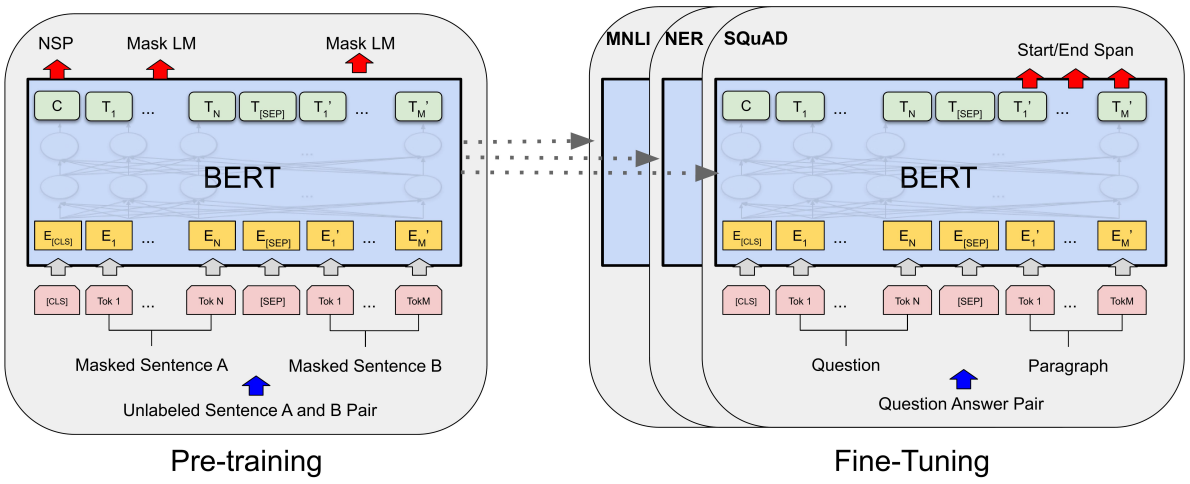


FIGURE 2.2: The pre-training and fine-tuning process in BERT [7]

by Liu et al. in 2019. On the basis of BERT, RoBERTa features on training the model longer with bigger batches and removing the next-sentence pre-training objective. These changes make RoBERTa outperforms BERT on several benchmark datasets.

ALBERT [30] (A Lite BERT) is another variant, proposed by Lan et al. in 2019, which makes significant modifications to reduce model size while retaining the performance. ALBERT employs two strategies: factorized embedding parameterization to separate the size of the hidden layers from the size of vocabulary embedding, and cross-layer parameter sharing to prevent the growth of parameters with depth.

In 2021, researchers from Microsoft published DeBERTa (Decoding-enhanced BERT with disentangled attention) [8], a significant advancement over the original BERT model. In contrast to BERT, DeBERTa utilises a disentangled attention mechanism, in which word content and position are separated. In addition, DeBERTa uses an enhanced mask decoder by taking both relative and absolute position of words into account. Given the similar model size, DeBERTa outperforms former models (e.g. RoBERTa, XL-Net, ALBERT, ELECTRA) in many tasks with only half of the training data, achieving state-of-the-art performance. [20]

Due to its remarkable performance and flexibility, BERT-based models have been a popular choice for various NLP tasks including text classification, named entity recognition, sentiment analysis, and question answering. However, despite their groundbreaking features, BERT and its variants does come with limitations. Their large size and extensive parameters can lead to overfitting when trained on limited

Name	Novel ideas	Size	Data	Training	Hardware	Speed	Performance
BERT	The first building block of all Pretrained LMs	Base: 110M Large: 340M	16GB	- Masked Language Modeling (MLM) - Next Sentence Prediction	Base: 16 TPU chips Large: 64 TPU chips	Both versions: 4 days	SOTA on GLUE and SquAD
Transformer-XL	- Introduce recurrence in attention-based models - Relave Positional Encoding						SOTA on WikiText-102, enwiki8, One Billion Word, Penn Treebank
XLNet	Combine Autoregressive and Bi-directional styles.	Comparable to BERT	158GB	Permutation Language Modeling	Large version: 512 TPU v3 chips	Large version: 5.5 days	Outperform BERT, SOTA on 20 tasks
RoBERTa	Better hyper-parameter tuning for BERT	The same as BERT	160GB	MLM	1024 32GB V100 GPUs	1 day	Outperform BERT, comparable to XLNet
DistilBERT	Distill from BERT	66M	Same as BERT	MLM with Distillation	8 16GB V100 GPUs	90 hours	97% of BERT BASE
ALBERT	- Factorized embedding parameterization - Cross-layer parameter sharing - Sentence Order Prediction	Base: 12M Large: 18M XLarge: 60M XXLarge: 235M	Union of data used for XLNet and RoBERTa	MLM and Sentence Order Prediction	64 to 512 TPU V3		Outperform BERT, RoBERTa, XLNet
BART	- Use the whole Transformer architecture - Reconstruct corrupted texts		Same as RoBERTa	Reconstruct corrupted texts			Comparable to RoBERTa, SOTA on some NLG tasks
MobileBERT	- Inverted-Bottleneck BERT - Careful optimizations for distillation	25.1M	Same as BERT	MLM and NSP with distillation	256 TPU v3 chips		Comparable to BERT
ELECTRA	Replaced Token Detection		Same as XLNet	Replaced Token Detection (RTD)	V100 GPUs	Match RoBERTa and XLNet performance with ¼ time	Outperform, RoBERTa, XLNet, ALBERT
ConvBERT	- Mixed attention and convolution - Span-based dynamic convolution - Grouped linear operator	Small: 14M Base: 106M	32GB	Replaced Token Detection		Outperform ELECTRA with ¼ time	Better performance and speed than ELECTRA
DeBERTa	- Disentangled attention - Enhanced mask decoder	Base: 134M	78GB	MLM (optionally RTD)	Base: 64 V100 Large: 96 V100	Base: 10 days Large: 20 days	Outperform ELECTRA, ALBERT
BigBird	Sparse attention		123GB	MLM	8 x 8 TPU v3		SOTA on long-text datasets

FIGURE 2.3: A summary of pre-trained language models [20]

data. Besides, BERT can only accommodate input sequences up to 512 tokens long, posing a significant constraint. Furthermore, their impressive performance heavily depends on substantial amounts of labeled data for training and fine-tuning, which might be a significant hurdle for specific tasks. [19] These limitations imply that there might be considerable room for improvement, especially in the context of document-level tasks such as legal document review. Therefore, the search continues for models that can deliver comparable or superior performance while overcoming the limitations.

2.3 Large Language Models and GPT

Large Language Models (LLMs) have emerged as a novel frontier in the field of Natural Language Processing (NLP). With their substantially larger model sizes, LLMs exhibit superior performance when compared to their BERT-based predecessors. Latest LLMs such as the GPT (Generative Pretrained Transformer) [22] series developed by OpenAI and the PaLM (Pathways Language Model) [4] developed by Google, have achieved state-of-the-art performance on a variety of benchmark tasks, including Text Classification and Question Answering.

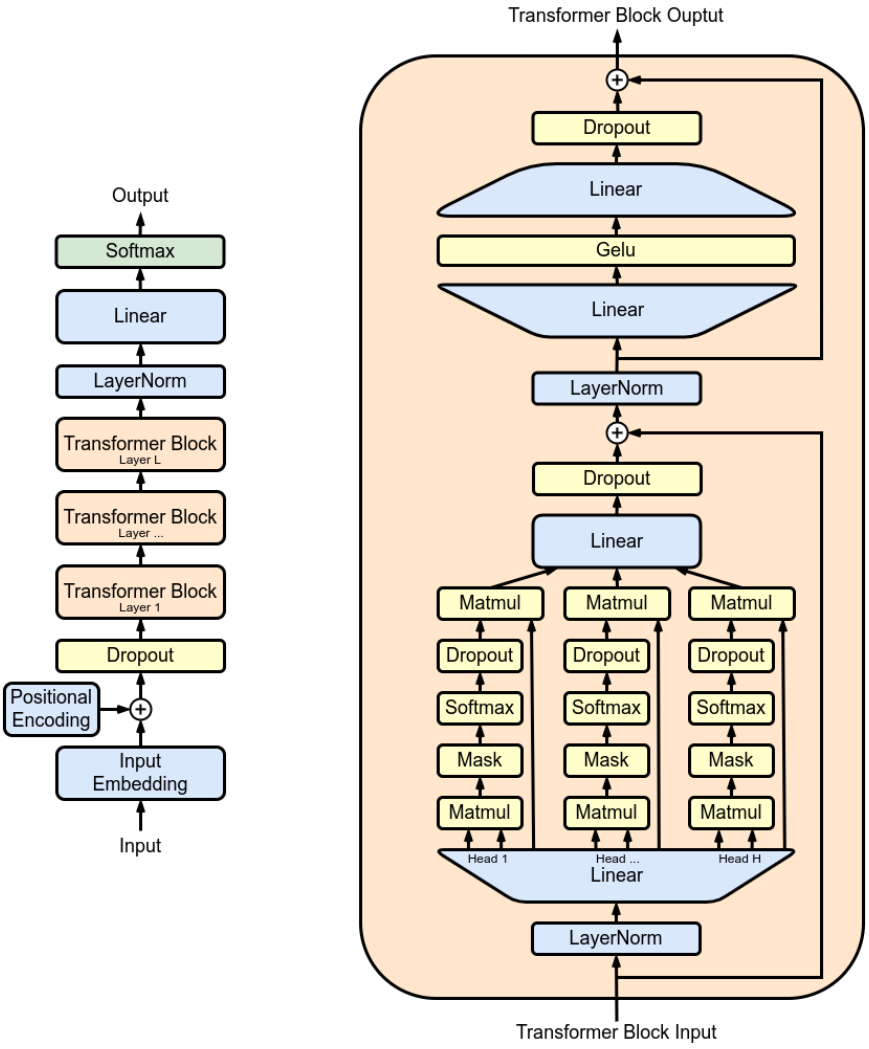


FIGURE 2.4: Original GPT model Architecture [22]

In June 2020, OpenAI launched the third iteration of the Generative Pretrained Transformer (GPT-3) [2]. This model stands out from its predecessors due to its scale and capability. With 175 billion parameters, GPT-3 became the largest language model upon its release (see Figure 2.5). The immense scale of GPT-3 allows it to capture a wider range of language patterns, nuances and contexts, making it highly effective at generating coherent and contextually relevant language outputs. Moreover, as a pre-trained model, GPT-3 benefited from training on a diverse range of internet text across vast array of topics. With its expansive pre-training data, GPT-3 models are capable of detecting subtle language nuances and make

more accurate predictions about the next word or sentence in a text. This ability is crucial in tasks that require understanding lengthy and complex texts.

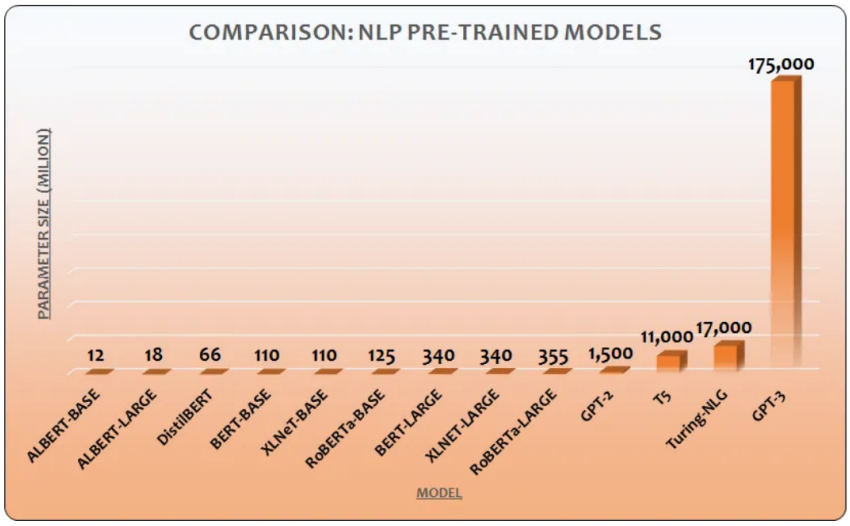


FIGURE 2.5: The parameter size of GPT 3 and some BERT-based models. [27]

Unlike the BERT-based models which require fine-tuning on a large labelled dataset, GPT models can perform tasks in a zero-shot, one-shot, or few-shot setting by using prompts that guide the model's predictions, reduces the need for task-specific fine-tuning and large labeled datasets. In addition, the vast and varied pre-training corpus allows GPT-3 models to comprehend and generate text based on a broader range of contexts without the need for task-specific training data. Furthermore, GPT-3 models can handle longer sequences of text, which is critical for tasks like contract review that involve longer documents as input. For instance, the Text-Davinci-003 model from GPT-3.5 series allows a maximum input length of 4096 tokens. This capacity is already sufficient for a variety of small and medium-sized contracts.

Although GPT-3 models achieve state-of-the-art performance on a variety of NLP tasks, we are more interested in their performance on specific tasks in legal domain. A research on Evaluation of GPT for Zero-Shot Semantic Annotation of Legal Texts [26] evaluates Text-Davinci-003's ability to semantically annotate short text snippets from various legal documents in zero-shot learning settings. The researchers focused on three tasks: contract review, statutory/regulatory provisions investigation, and case-law analysis. They compared the performance of GPT-3.5 with that of a traditional statistical machine learning model (random forest) and RoBERTa. The results showed that GPT-3.5 performs impressively well in a

zero-shot learning setting, achieving a micro F1 score of .73 for adjudicatory decisions, .86 for contractual clauses, and .54 for statutory and regulatory provisions. These results are significant for researchers and legal professionals that aim to reduce the cost of high-volume workloads involving semantic annotation of legal documents.

In conclusion, GPT models have demonstrated exceptional potential in the field of contract review. The key advantage of GPT models lies in their ability to function without the necessity of task-specific fine-tuning and large labelled data as required in BERT-based models. This advantage potentially positions GPT models as a future alternative to BERT-based models in AI-based contract review. However, in contrast to the BERT-based models, GPT models function on the basis of prompt-based learning. Consequently, crafting effective prompts that enable the GPT model to accurately understand the task and adapt to the complex legal text remains a critical challenge.

2.4 Prompt Engineering

Prompts serve as inputs or queries that users provide to elicit specific responses from a Large Language Model (LLM). Fundamentally, a well-crafted prompt steers the model's focus and leads to more desired and accurate outputs. Therefore, we introduce the concept of prompt engineering.

Prompt Engineering is a relatively new discipline focusing on the development and optimization of prompts to assist users in applying Large Language Models to various scenarios and research fields. Mastering skills related to prompt engineering can help us better understand the capabilities and limitations of large language models. Through prompt engineering, users can enhance the capabilities of large language models by using professional domain knowledge and external tools. Researchers can utilize prompt engineering to enhance the ability of large language models to handle complex task scenarios, while developers can design and develop powerful engineering techniques through prompt engineering and efficiently integrate them with large language models. [25]

The quality of results depends on the structure of the prompt and the information it conveys. A prompt typically contains the following elements:

- **Instruction** - a specific task or instruction you want the model to perform
- **Context** - external information or additional context that can steer the model to better responses
- **Input Data** - the input or question that we are interested to find a response for

– **Output Indicator** - the type or format of the output

Crafting effective prompts is an iterative process that requires consistent experimentation to achieve optimal results. The process usually begins with a simple and basic prompt, then is gradually enhanced by adding more elements and context for better results. Always keep the instruction clear and specific, and include contextual information in the prompt. In addition, control the output length and format for a desired output.

A critical component in prompt engineering is the examples we use to guide the model's behaviour. Depending on the number of examples provided in the prompt, we can generally categorize our approach into two main strategies:

Zero-shot Prompting: In zero-shot prompting, no specific example is given in the prompt. GPT models rely solely on the pre-existing knowledge from their pre-training process to complete the task. An example of zero-shot prompting is like:

Classify the text into neutral, negative or positive.
Text: Study makes me happy.
Sentiment:

The model produces the output of "Positive". This is because the LLM already understands "sentiment", and that's the capability of zero-shot.

Zero-shot prompting has simplicity and efficiency, but without any additional context or direction about the task for the model, the result can be unpredictable.

Few-shot Prompting: While LLMs demonstrate remarkable zero-shot capabilities, they still fall short on more complex tasks. That's when we need few-shot prompting. Few-shot prompting involves providing several examples in the prompt. By providing a clearer pattern, the model often produces more accurate and contextually appropriate responses, making few-shot prompting especially effective for complex tasks. Furthermore, providing multiple examples can help the model adapt to specific tasks that pattern is less common in the training data.

An example of few-shot prompting is like:

Classify the text into neutral, negative or positive.
Text: Today is a nice day.
Sentiment: positive
Text: What a horrible show!
Sentiment: negative
Text: The food is okay.
Sentiment:

In this example, we provide the model with multiple examples of input-output pairs to prime it for a text sentiment classification task. Each input-output pair consists of a text (input) and its corresponding sentiment (output). The model can draw upon these examples to understand the task it needs to perform.

In addition, there are several tips for effective few-shot prompting [24] [17]:

(1) Make sure that the label is well-defined and relevant to the task at hand. This significantly affects the performance of the model.
(2) Balance the distribution of input text in the demonstrations. The model's ability to generalize to unseen examples heavily relies on this aspect.
(3) Maintain a consistent format in the demonstrations. Consistency helps the model in comprehending the task and in generating better responses.

In general, few-shot prompting is powerful in getting the model to produce accurate and properly formatted output, especially for complex tasks. However, while few-shot prompting can often lead to better performance than zero-shot prompting, irrelevant or misleading examples may not lead to better performance in few-shot prompts.

Another emerging strategy in prompt engineering is **Chain-of-Thought** [29], a method that enhances the complex reasoning abilities of large language models.

By providing a series of intermediate reasoning steps or 'chains of thought' as part of the prompts, the models have demonstrated marked improvement in a variety of tasks. This expands the range of tasks that large language models can perform. As a powerful tool that significantly improves the reasoning abilities of large language models, the Chain-of-Thought method might be significant for tasks like question answering.

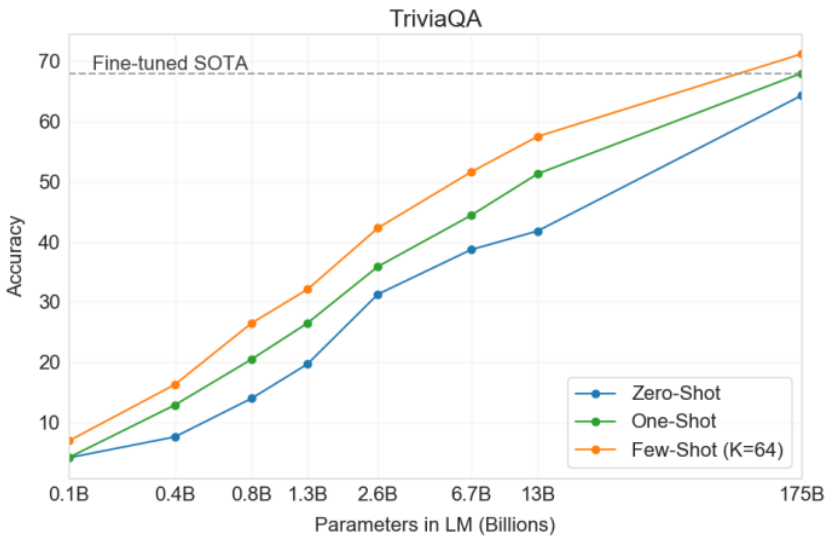


FIGURE 2.6: GPT-3 Performance on TriviaQA [11]. One-shot and few-shot performance make significant gains over zero-shot behavior[2]

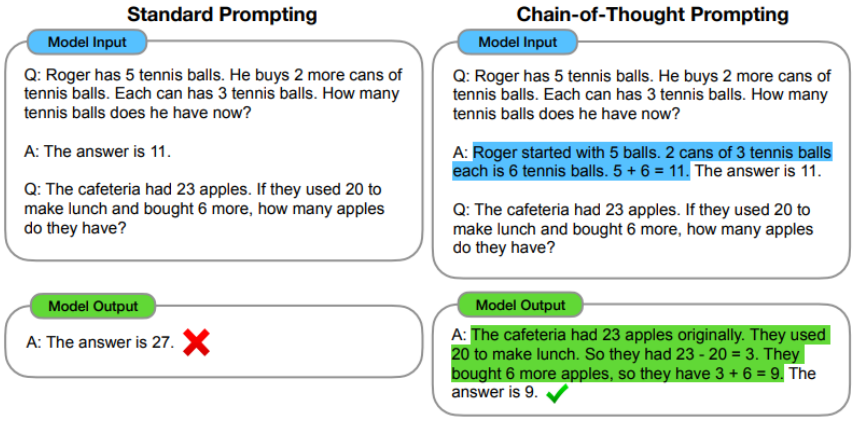


FIGURE 2.7: An example of Chain-of-Thought prompting. [29]

2.5 Prompt Tuning

Prompt tuning is a more intensive approach compared to prompt engineering, which primarily refines the prompts or inputs given to the model. Rather than merely tweaking the input, prompt tuning adjusts the AI model's parameters, allowing for more specialized modifications to the model's behavior. This results in outputs that are more precise, pertinent, and dependable. Through prompt tuning, the model learns nuanced alterations to its internal representations, which enhances its performance on specific tasks or prompts. This is achieved without the need for extensive retraining or alterations to the base

model. Prompt tuning is particularly useful when using limited data to tailor a massive model to a narrow task. [21]

For instance, consider a user query such as "What is prompt tuning?". You could refine this query to improve specificity, such as "Explain the concept of prompt tuning in AI and its purpose." Following this, evaluate the AI's response and if needed, modify the prompt once more to extract a more precise or succinct response. For example, "In two sentences, describe prompt tuning and its benefits in AI."

Prompt tuning effectively addresses some of the inherent challenges of AI models, such as the tendency to be overly verbose or literal, by expecting more concentrated and concise outputs. As AI technology evolves, a comprehensive understanding of prompt tuning becomes vital in enhancing the efficiency of AI models. This optimization paves the way for productive human-AI synergy. By cultivating this collaborative energy, we can maximize the utility of AI technology, thereby augmenting our efficiency and promoting informed decision-making across a broad spectrum of fields.

Pre-trained models, such as GPT-3, are initially trained on massive datasets, during which they grasp language structure, grammar, and a breadth of general knowledge. Nonetheless, these models may encounter difficulties when tasked with specialized operations or generating responses tailored to specific domains. Fine-tuning is the act of further specializing a model that has typically been trained on a broader data distribution by updating the model's parameters in some way. As a traditional method that has been around as long as the idea of pre-training models, fine-tuning can ensure that the model generates more accurate, relevant, and reliable results in their specific context, which in turn leads to better decision-making. Therefore, we are particularly interested in comparing the performance of prompt tuning with fine-tuning in optimizing the performance of similar tasks.

Srijan et al. (2023) [1] provided an exhaustive analysis of two tuning paradigms - model tuning and prompt tuning - for unified question-answering (QA) tasks under low-resource settings. It employs 16 QA datasets to showcase the benefits and drawbacks of each approach in various scenarios. The paper finds that prompt tuning can perform as well as model tuning in few-shot settings, given a good initialization. In terms of parameter sharing, it leads to superior performance in the few-shot setting but falls behind model tuning in full-shot settings. In conclusion, the paper demonstrates the potential of prompt tuning as a solution for unified QA, especially in low-resource, few-shot scenarios.

In conclusion, prompt tuning offers substantial improvements in the model's performance by enhancing its ability to make accurate predictions and generate desired outputs. The effective and efficient feature of

prompt tuning has made it a viable alternative for model customization and optimization when contrasted with fine-tuning.

2.6 Legal Document Datasets

As the need for automated contract review grows, numerous legal document datasets have emerged in recent years. Leivaditi et al. (2020) [15] introduced their datasets of 179 lease agreement documents manually annotated for the extraction of specific contract terms, aiding users in understanding a contract by offering a consistent legend for the terms discussed within. However, their work primarily focuses on term identification, without providing clarifications about what these terms stipulate, which is a crucial aspect of contract review.

To address this issue, Yuta et al. (2021) introduced the ContractNLI [13] Dataset consisting of 607 annotated contracts and showed that linguistic characteristics of contracts, particularly negations by exceptions, make the problem difficult. They performed tasks including document-level three-class classification and multi-label binary classification. However, their focus was limited to a single type of contract. While this approach enables the inclusion of less common and more nuanced examples, it also restricts the method's generality and adaptability due to the complex structures across different contract types.

The CUAD (Contract Understanding Atticus Dataset) paper [9] introduces a high-quality dataset designed for contract review, aiming to gauge the performance of NLP models in highly specialized domains. The dataset comprises 510 contracts, which belong to 25 different types, with a total of 13,101 labeled clauses. These contracts vary in length from a few to over a hundred pages. The authors designed labels around clauses that would typically require a lawyer's review or analysis. The labeling process was manually conducted by law students who were trained by experienced lawyers and followed detailed labeling instructions for consistency. For the prediction, they trained the models to extract relevant clauses corresponding to label categories by identifying the start and end tokens that mark the relevant text.

The authors tested multiple BERT-based models on the CUAD dataset (as shown in Figure 2.8), concluding that while the performance was promising, there was significant room for improvement. They also suggested that the CUAD dataset serves not only as a tool for accelerating contract review-focused research, but also as a benchmark for evaluating NLP models in legal domains.

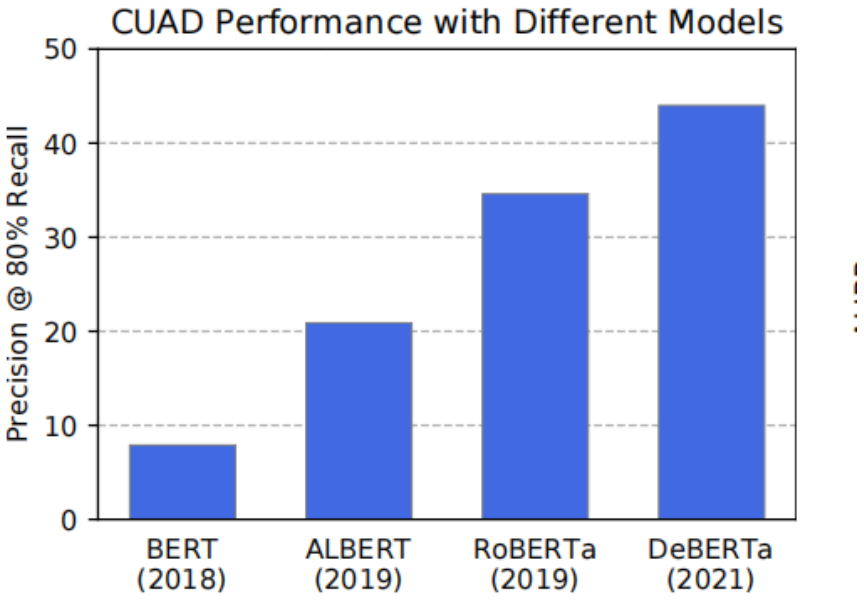


FIGURE 2.8: Performance of BERT-based models on CUAD. Each bar is an average of the performance of all models in each model class. [9]

The CUAD dataset is critical for our research as its careful and detailed labeling process, coupled with the diverse range of its content, provides a solid foundation for training and evaluating our models in contract review tasks. Furthermore, the potential for performance improvement in BERT-based models, as indicated by CUAD's evaluations, underscores the relevance of our investigation into the use of GPT models for this task. In essence, CUAD serves as an invaluable resource and benchmark for our research to enhance AI-based contract review.

CHAPTER 3

Methods

In this study, we aim to replicate the process outlined in the Contract Understanding Atticus Dataset (CUAD) [9] paper, but with a key innovation: we substitute the BERT-based model used in the CUAD paper with the latest GPT model, specifically GPT-3.5 Turbo. Our objective is to verify the hypothesis that the advanced language processing capabilities of GPT-3.5 Turbo may outperform the prevalent BERT-based models in assisting legal contract review task. To achieve this, we will design a similar experiment on the CUAD dataset, but conduct specific model training, validation, and testing exclusively conducted with GPT-3.5 Turbo.

This chapter focuses on the details of the methods employed in our work. We start with the definition of our task, followed by the specifics of the experiment design, including input processing, model implementation, and the selection of evaluation metrics.

3.1 Task Definition

In legal contracts, typically there's only a small number of key clauses in the contract require a lawyer's review, and our goal is to automate the identification and extraction of such clauses.

Specifically, we aim to use Large Language Models to highlight the text associated with 41 labels categories. Given an input document, We expect the model to identify if it contains any text relevant to each of the labels, and then extract and output the most relevant text. The annotation process is depicted in the following Figure 3.1, adapted from the CUAD paper.

The key lies in effectively adapting the LLM to our specific task. Detailed procedures and methods for this adaptation will be discussed in the following sections.

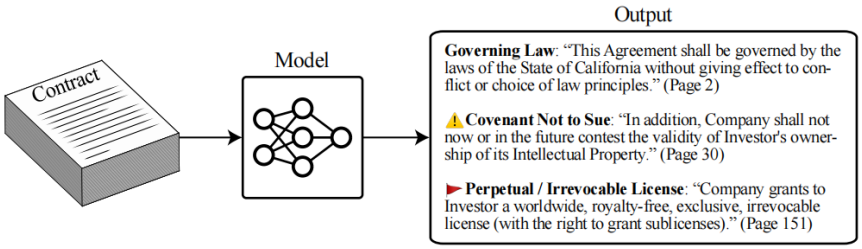


FIGURE 3.1: An example of the automated annotation process from CUAD paper: The model takes a contract as the input and outputs relevant clauses for each label [9]

3.2 Implementation of LLMs

3.2.1 Model Selection

We chose GPT-3.5 Turbo for our experiment. As the latest GPT model from OpenAI that has been made accessible for API use, GPT-3.5 Turbo has enhanced its speed and accuracy compared to its predecessor, GPT-3. Trained on a diverse range of internet text, GPT-3.5 Turbo is highly suitable for tasks that require context understanding. Moreover, GPT-3.5 Turbo performs at a similar capability to the earlier GPT-3 model but at only 10% of the price per token, making it a more cost-effective choice for large-scale experiments.

3.2.2 Building GPT-based Application

We use LangChain [3] to build our own application based on GPT models. LangChain is a powerful framework designed to simplify the development of LLM-powered applications for researchers. It provides a number of components, including a generic interface to a variety of models and a framework to manage the prompts. In addition, it allows users to assemble these components for their specific tasks and integrate additional resources.

After loading LangChain as a Python library, we start to build our GPT-based application. After setting up the API key from OpenAI in the environment, we're able to initiate our LLM by loading the GPT-3.5 Turbo model. We can now send queries to GPT model and receive the output. This Figure 3.2 here shows the work flow of our application based on LangChain:

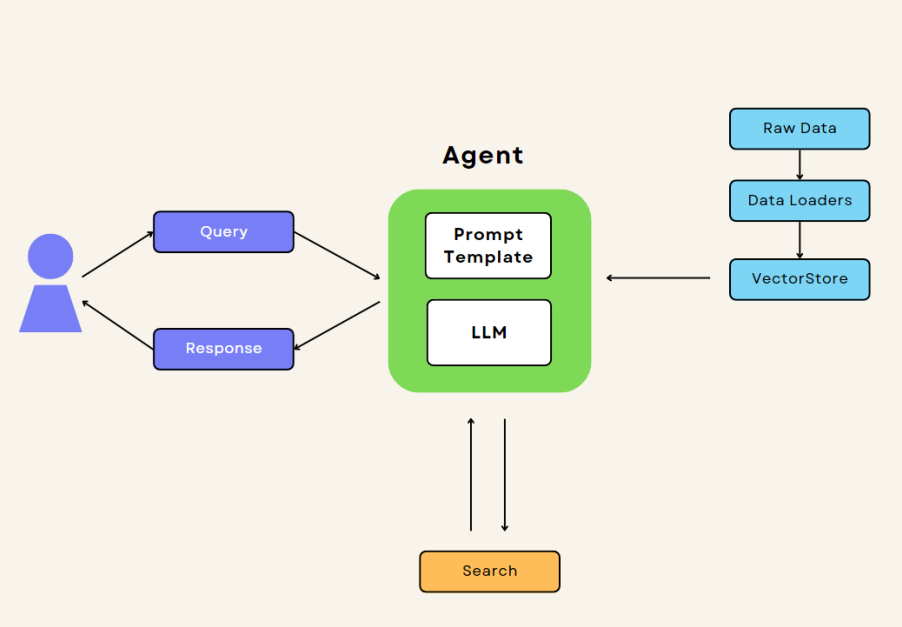


FIGURE 3.2: Our work flow with LangChain, explaining the process from query submission to receiving model output

3.3 Prompting Methods

A fundamental difference between BERT-based models and Language Learning Models such as GPT models is that LLMs rely on prompts provided by users to communicate and direct their behavior. Prompts serve as inputs or queries that users can provide to elicit specific responses from an LLM.

3.3.1 Zero-Shot Prompting

In zero-shot prompting, no examples or labeled data are provided to the model. We designed a basic prompt, outlining our task and supplying label categories and document text as parameters. In this case, the model generates responses solely based on its pre-existing knowledge of the labels and relevant clauses.

The zero-shot approach allows us to assess the baseline performance of the GPT-3.5 model on our task without any task-specific guidance or examples. It serves as a benchmark, providing valuable insights into the raw capability of the model in handling our contract review task. This foundational understanding helps us with our exploration of one-shot and few-shot approaches. However, while GPT-3.5 has impressive zero-shot capabilities, it's often insufficient in complex tasks. In that case, we'll need a more powerful prompting method.

3.3.2 One-Shot Prompting

We start with the one-shot approach, a step forward from our zero-shot method. In the one-shot approach, we augment our base prompt with a single example of our task input-output pair before the model makes its predictions. The example serves as a guide to show the model the task process and expected output.

The process of example selection is a critical aspect of our methodology for one-shot and few-shot prompting in the GPT-3.5-Turbo model. In this work, we utilize examples from the Contract Understanding Atticus Dataset (CUAD) to maintain the consistency on label definition and identification between the our implementation and the implementation of the authors of CUAD.

In our implementation, **the selection of examples is performed randomly from the CUAD training dataset** to ensure an unbiased representation of the dataset and to avoid overfitting to specific instances. This random selection process aids in encapsulating the variety and complexity present in the CUAD dataset, thus providing the model with diverse contexts to learn from.

In our one-shot prompts, one single randomly-selected example is included to guide the model's response. The incorporation of the example alongside task descriptions aids in providing an extensive comprehension of the tasks to the model. The selection of examples greatly influences the model's performance, particularly in identifying and extracting certain labels that proved challenging for BERT models.

The goal of our one-shot prompting is not to provide label-specific information to the model, but rather to clarify the desired type and format of the output. Based on this, our experiments should focus on understanding the impact of this single example on the model's output, specifically regarding the output's type and format.

3.3.3 Few-Shot Prompting

Few-shot prompting involves providing the model with a small set of examples, typically two to five, to guide it in adapting to various question-answering tasks and producing the desired output. We extend our one-shot approach to few-shot prompting by providing more label-specific examples and instructions. As the number of examples increases, we could enhance the model's understanding of our task.

However, it's worth noting that the specific number of examples can have a significantly impact on the language model's performance.

The selection of the optimal number of examples is contingent on various factors. These include the specificity of our task, the diversity of responses we expect from the model, and practical considerations such as the costs of preparing and executing those examples and the time spent on prediction. In our context where the model identifies and extracts clauses relevant to a label in a legal contract, a few examples are needed to adequately "teach" the model our task specifics. In addition, considering the complex structure of contracts and the variable format of related clauses, diverse examples are required to cover enough scenarios.

However, although generally the performance of model improves with the more examples provided, the selection of examples is a trade-off between the benefits of added performance and the costs of preparing those examples. Balancing these considerations, **we opt for three examples in each few-shot prompt**, providing a manageable yet representative range of scenarios.

Our experimental goal is to evaluate the influence of the number of examples on the model's performance. Following the evaluation of initial experiments, we perform iterative prompt tuning, refining our approach based on the observed results.

In conclusion, through zero-shot, one-shot, and few-shot prompting, we aim to understand how different levels of guidance impact GPT model performance on our specific task. By comparing these strategies, we seek to identify the ideal balance between prompt simplicity and complexity for contract review tasks. The results of this comparative analysis will be further discussed in Chapter 4.

3.4 Input Processing

3.4.1 Dataset Structure

As previously discussed, the CUAD dataset is the ideal choice for our research into LLM-assisted contract review. The authors of CUAD has structured CUAD in the same format as SQuAD 2.0 [23], a reading comprehension dataset with questions that have spans of the passage as answers.

CUAD is organized as a JSON file, which comprises a nested structure of lists and dictionaries containing over 500 contracts. Each contract in the dataset is represented as a dictionary included "qas"

and "context" fields. The "context" filed contains the contract content, while the "qas" field is a list of dictionaries, each corresponding to a question-answer pair for each of the 41 Atticus Labels.

Each dictionary within the "qas" field contains "question", "answer" , "answer_start" and "is_impossible" fields. They respectively represents the question body, the related text in the document, the index of the answer in the document and whether there's an answer to this question (whether there are related text in this document). This structured format allows us to retrieve and process the relevant information for our experiments.

3.4.2 Text Segmentation

This Figure 3.3 here shows the distribution of the document lengths across the CUAD dataset. In LLM like GPT-3.5 Turbo, a document's length is typically measured in tokens. Given that GPT-3.5 Turbo allows a maximum length of 4096 tokens and there are a number of contracts in CUAD exceeding this limit, it's crucial that we segment long documents into smaller pieces for GPT models.

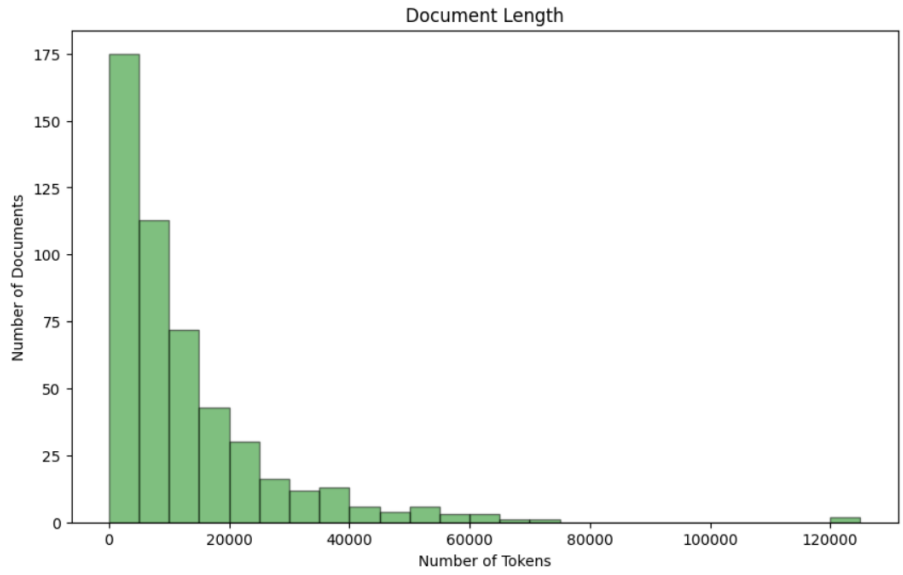


FIGURE 3.3: Distribution of Document Lengths in the CUAD Dataset: Each contract has been tokenized using the GPT tokenizer. The histogram illustrates that while most contracts contain fewer than 5,000 tokens, many of them extend to 20,000 tokens or more.

Consequently, we adopt a sliding window approach, mirroring the method used by the authors of CUAD. A sliding window segments long documents into smaller subsets based on a given window size, and then slides through the document to capture the entire content. The overlap between windows ensures that no

important information is lost at the boundaries. Furthermore, the sliding window can preserve the local context in each segment, which is crucial for a question answering task. A sliding window can be used as a consistent and efficient method across all documents.

The sliding window approach has the flexibility of allowing us to adjust the window size and stride size based on specific tasks and performance. When setting our window size, we had to consider several factors:

The model needs enough context for accurate predictions, especially when handling extensive and complex contract clauses. Moreover, a smaller window size would result in more segments per document and thus more API calls, increasing the computational time and cost. However, considering maximum input length of 4096 tokens for GPT-3.5 Turbo includes both input and output tokens, We must ensure sufficient room for the prompt and output. A too large window size will risk breaching the token limit and affecting the quality of predictions.

Given these considerations, we decide to set the window size to 3000 tokens. This window size allows us to process most of the documents in CUAD in a few segments, thus maintaining efficiency while also giving the model a reasonable amount of context for prediction. As for the stride size, we set it to 2800 tokens. Since the answers to questions are typically one or two sentences long, an overlap of 200 tokens should be sufficient for the model to capture the relevant context.

3.5 Output Validation

In our prompt, we have structured the desired output for each question as a combination of two fields, "Highlighted Text" and "Is_impossible". Correspondingly, we collect the actually answer by retrieving the corresponding field in the CUAD dataset. The "Highlighted Text" serves as the answer string while the "Is_impossible" returns a Boolean value reflecting whether this question is answerable or not. In this section, we introduce the methods we developed to validate the model's predictions against the ground truth.

3.5.1 Answer Extraction

Given the implementation of the sliding window in our approach, the question-answering process produce multiple potential answers across different windows for each question. To handle this, we implement a merging strategy to merge these answers.

Our merging strategy depends on two factors: overlap and length. If an overlapping exists among the answers obtained from different windows, we combine the answer strings to form a comprehensive answer to each respective question. In cases where no overlap exists between the answers, we opt for the longest answer. This methodology aims to optimize the likelihood of acquiring the most accurate answer and generates a final answer for each question from the given document.

3.5.2 Textual Similarity

In our task, we determine "correct identification" by checking for a match between the "is_impossible" fields in the model's output and the ground truth, while "correct answer" is verified by the textual Similarity between "Highlighted Text" field.

To determine a match between the predicted output and the answer from labelled dataset, we implement the Jaccard Similarity Coefficient strategy. This metric quantifies the degree of similarity between two text strings, in our case, the predicted answer and the answer from labelled dataset. Our implementation of the Jaccard similarity follows the following steps:

- **Normalization:** Both the model output and the ground truth are normalized. This involves removing punctuation, replacing the "/" character with a space, and converting all text to lowercase. Normalization step ensures that variations in case or punctuation do not affect the similarity score.
- **Intersection and Union Calculation:** The Jaccard similarity is computed as the size of the intersection (the common words in both strings) divided by the size of the union (the total number of unique words across both strings).

The calculation result should be a value between 0 and 1, where 1 indicates that the two strings are identical (disregarding the order), and 0 indicates that they have no words in common. We set a threshold of 0.5 for the Jaccard similarity based on the methodology followed by the CUAD authors, marking a match if the similarity is greater than this threshold between the model prediction and ground truth.

CHAPTER 4

Experiments and Results

This chapter focuses on the design and analysis of our experimental process. We start with an overview of our experiment setup, including implementation details and metrics we plan to use, followed by a detailed description of our iterative prompt tuning process. We then evaluate our model's overall performance across different prompting methods and delve into a breakdown of its performance by category. Finally, we conduct an error analysis on edge cases displaying poor performance.

4.1 Experiment Setup

4.1.1 Implementation Details

To evaluate the performance of the GPT-3.5 Turbo model in identifying and categorizing clauses in legal contracts, we conducted experiments using the Contract Understanding Atticus Dataset (CUAD) [9].

The experiments were implemented using Python 3.8. We crafted the prompt template under LangChain framework [3], and GPT-3.5 Turbo was accessed through the OpenAI API.

Our GPT models were prompted using different techniques: zero-shot, one-shot, and few-shot prompting. For each technique, we crafted specific prompts aimed at helping the model to correctly identify and categorize the clauses in the contracts.

After crafting the prompts, we preprocessed the input contracts as parameters passed into the prompt. To handle contracts that exceeded the maximum token limit, we utilized a sliding window approach. It allowed us to segment the contracts into smaller portions, and therefore fit in with the input length limit of the GPT models.

The output from the GPT models was then post-processed to extract the predicted answer. The extraction process involved parsing the model's responses to relevant clauses and a Boolean value indicating the

existence of answer. The results were then matched and compared with the ground truth in the CUAD dataset.

4.1.2 Evaluation Metrics

Confusion matrix, also known as error matrix, is commonly employed in classification and question-answering problems. Given our task is a hybrid of question answering and information extraction, it is suitable for our task due to its binary nature.

We defined the elements in confusion matrix as follows:

- **True Positives (TP):** The model correctly identified and answered the question.
- **False Positives (FP):** The model answered the question, but there should not be an answer according to the ground truth.
- **False Negatives (FN):** The model did not answer the question, but there should be an answer according to the ground truth.
- **True Negatives (TN):** The model correctly identified that there should not be an answer to the question.

With the these definitions established, we can compute a set of metrics to assess the performance of our model: Precision, Recall, Accuracy, and F1-Score.

- **Precision** measures the proportion of correctly identified answers out of all answers the model identified:

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

- **Recall** measures the proportion of correctly identified answers out of all answers that should have been identified according to the ground truth:

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

- **Accuracy** measures the proportion of correct identifications out of all identifications:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.3}$$

. **F1-Score** is the harmonic mean of Precision and Recall, providing a single measure that balances both values:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4.4}$$

Each of these measures offers a different perspective on the model's performance, and together they provide us with a comprehensive understanding of our model's effectiveness in our question answering task.

4.1.3 Datasets

Our experiments were conducted on the Contract Understanding Atticus Dataset (CUAD), a large labelled dataset that includes more than 500 contracts and over 13,000 expert annotations spanning 41 label categories. Manually labelled by experts, CUAD has proven to be an ideal resource for training and evaluating models. In our work, we leveraged CUAD as a benchmark to validate our methods and evaluate performance against other BERT-based models.

The prediction of GPT-3.5 Turbo was performed on CUAD test dataset. The methods employed for processing the CUAD dataset and verifying the predictions of our model on labeled data have been thoroughly discussed in Chapter 3.

4.1.4 Initial Prompt Design

Our task involves identifying relevant contract clauses associated with multiple labels. Given the quantity of labels and the extensive length of the legal contracts, it is impractical to encapsulate all labels in a single prompt, as one might in a text classification task. Furthermore, most clauses in a contract do not correspond to any of the labels, and it's common for a contract to contain no clause related to a particular label.

Therefore, formulating our task as a question related to a single label for each document and then iterating this process through all documents proves to be a more efficient and manageable approach. Our general prompt structure was designed as follows:

Highlight the parts (if any) of this document related to "<Label Category>" that should be reviewed by a lawyer. Details: <Label Category Description>.
Document: <Document Text>

Highlighted Text:
Is_impossible:

Using LangChain, we were able to create a prompt template that dynamically incorporates user-specified parameters and then formats the prompt with that information. For each document input, we iterated through 41 labels, passing the corresponding label category and label category description into the prompt, and then anticipate the output from the model in a desired format. The field "Highlighted Text" in our prompts serves as the output indicator for model output, while "Is_impossible" aligns with the CUAD datasets, indicating the presence or absence of an answer within the contract context. This structure forms the basis of our zero-shot, one-shot, and few-shot prompts.

4.2 Prompt Tuning

Prompt tuning was a dynamic and iterative process throughout our experiments, which involved continuous updates and refinement to our prompts based on experimental results and performance feedback. In this section, we focus on introducing the complete life cycle of our prompt tuning efforts.

4.2.1 Output Format

Our initial experiment started from the basic structure of the prompt as described above. We executed the prompts by providing the parameters such as label category and document text, intending to evaluate the model's unaided output. However, we noticed inconsistencies in the model output. Many responses were unformatted and therefore hindered the extraction of answers. This could be due to the unclear expression of our task, given that we only included a single question in the prompt.

To address this issue, we incorporated explicit instructions in the prompt for producing a consistent output in a predefined format:

Desired Format:

Highlighted Text: Output the span of text in the given document that should be highlighted as relevant to the label category in the question, output N/A if nothing should be highlighted
Is Impossible: Output True if there's no text related to the label, otherwise False

With the revised zero-shot prompt, we can now reliably extract data from the model's output, then compare model predictions with ground truth as planned.

4.2.2 General Task Instructions

While the model now produce desired output format, we encountered a new issue: the model often generated and included its own text in the "Highlighted Text" field. The generated text was not extracted from the document according to our fundamental task requirement. This suggested that while the model had learned to conform to our prescribed format, it still lacked sufficient task comprehension to extract the correct type of content.

To rectify this, we augmented our prompt with general task instructions:

Instructions:

1. Label the text by highlighting the span of text in the given document text that is relevant to the label category in the question. Refer to the description to determine if the text is relevant.
2. Try to be accurate and do not consider the text relevant unless it's closely related to the label.
3. Be consistent in the output with desired format and only output the relevant text itself in Highlighted Text, no other output needed

The result indicated that with the augmented prompt, the GPT models demonstrated a considerable improvement in understanding the task by restricting their "Highlighted Text" output to relevant extracts from the provided document.

4.2.3 Label-specific Rules

While the results achieved from one-shot prompting were indeed satisfactory, we identified substantial opportunities for improvement for several labels, such as "Covenant not to sue", "Post-Termination

Services", "Change Of Control", etc. These labels had a precision of less than 0.2, suggesting that the model frequently made incorrect predictions.

Upon a more detailed examination of the performance metrics, we realized that a low precision was generally attributed to two main issues: the model often make predictions under a non-answerable scenario (no answer according to the ground truth), and it also provided incorrect predictions. This indicates that the model faced difficulties in accurately identifying some labels, especially the less common ones.

In response to these findings, we made a significant modification: we decided to craft prompt for each label separately. This decision came from the inconsistent performance of the model across all labels. For those label with poor performance, we noticed that the model often misunderstood the label category and the conditions under which a clause should be marked as relevant.

To assist with crafting our prompt, we referred the Contract Understanding Atticus Dataset (CUAD) (Commercial Contracts) Labeling Handbook, released by the researchers who introduced the CUAD dataset. This handbook contains selected guidelines developed by the Atticus Project that used to guide their volunteer annotators in curating CUAD. In addition, the handbook provides the interpretation of selected Atticus Labels and contract clauses, and specific labelling instructions for them. With this, we could ensure that the model's understanding aligns with the CUAD dataset by feeding it label-specific information.

Utilizing the guidelines from the handbook, we managed to summarize specific rules for identifying a single label. For instance, for the label "Non-Compete", we incorporated the following detailed rules:

Specific rules for this label <Competitive Restriction Exception>:

1. Label clauses stating exceptions to Exclusivity, Non-Compete, or No-Solicit of Customers.
2. Include clauses within the restrictive covenant or separate sentences that provide exceptions.
3. Label clauses indicating termination events for Exclusivity or Non-Compete.
4. Do not label clauses about other party's approval or consent.
5. Do not label NDA exceptions as Competitive Restriction Exception.

This approach distinguishes from the general rules and instructions that were applied uniformly to all labels. These label-specific rules are exclusively included in the prompts addressing the identification of the corresponding label.

Following this methodology, we proceeded to design label-specific prompts for each of the 41 labels. These customized prompts are then employed align with the label category during our interaction with the model.

4.2.4 Examples in Prompts

Following a typical strategy of prompt engineering, we divided our prompts into zero-shot prompt, one-shot prompt and few-shot prompt by different number of examples included in the prompt.

Selection of examples is performed randomly from the CUAD training dataset to ensure an unbiased representation of the dataset and to avoid overfitting to specific instances. Based on the randomly selected example from dataset, we incorporated the following content to our prompts:

> *Examples:*
>
> *If the document text is:*
> *"""*
> *Exhibit 10.16 SUPPLY CONTRACT Contract No: Date: The buyer/End-User: Shenzhen LOHAS Supply Chain Management Co., Ltd.*
> *"""*
>
> *Then the output should be:*
> *highlighted_text: SUPPLY CONTRACT*
> *is_impossible: False*

Our initial experiment relied on a basic zero-shot prompt. In this context, no additional task-specific example or instruction was given, intends to measure the model's performance solely based on its understanding of question body and its existing knowledge.

Based on this, we evolved our prompts into one-shot and few-shot prompts by adding one or more examples of input-output pairs, which demonstrates the process of extracting relevant clause for a given

label from the document text. The inclusion of these examples offered the model further context and a richer understanding of the task at hand, with the ultimate goal of enhancing its performance.

The same experimental protocol was applied across all three types of prompts. The corresponding results were then evaluated to discern the impact of varying the number of examples on the model's output. The results and evaluation of these experiments are detailed in the following sections of this chapter.

4.3 Overall Performance

To compare performance across zero-shot prompting, one-shot prompting and few-shot prompting, we compute the average precision recall and f1-score among all label classes for each prompting strategy. The result is shown in Table 4.1.

Prompting	Average Precision (%)	Average Recall (%)	Average F1 Score (%)
Zero-Shot	29.2	79.9	42.7
One-Shot	41.5	81.6	54.8
Few-Shot	52.1	82.1	63.4

TABLE 4.1: Overall performance of GPT-3.5 model based on different prompting techniques

Compare between prompting methods: The results of our experiments showed a consistent pattern of improvement across the different prompting methods: zero-shot, one-shot, and few-shot. The recall scores, indicating the proportion of relevant labels our model identified, improved slightly from 79.9% in the zero-shot scenario to 82.1% in the few-shot scenario. This high recall rate is particularly important in the context of legal document annotation, where missing a relevant clause could have significant consequences. The gradual increase in recall demonstrates the model's ability to capture a majority of the relevant labels, even with minimal training examples, thus reducing the chances of false negatives.

Precision started at 29.2% for zero-shot, jumping to 41.5% in the one-shot scenario, and further increasing to 52.1% in the few-shot scenario, indicating the positive effect of including examples and instructions in the prompt. The F1 score, representing the harmonic mean of precision and recall, also demonstrated consistent improvements across the three methodologies. The score rose from 42.7% in zero-shot, to 54.8% in one-shot, and reached 63.4% in few-shot, effectively reflecting the comprehensive enhancement in the model's performance with the few-shot prompting.

Model	Precision at 80% Recall
BERT-base	8.2
BERT-large	7.6
ALBERT-base	11.1
ALBERT-large	20.9
ALBERT-xlarge	20.5
ALBERT-xxlarge	31.1
RoBERTa-base	31.0
RoBERTa-base + Contracts Pretraining	34.1
RoBERTa-large	38.1
DeBERTa-xlarge	44.0

TABLE 4.2: Overall performance of BERT-based models on CUAD datasets. All models has been trained and fine-tuned by the authors of CUAD.

This consistent improvement in F1-scores across the three prompting methods underscores the comprehensive enhancement in our model's performance with few-shot prompting, reflecting an effective balance between precision and recall.

Compare with BERT: We also present the results of various BERT-based models on CUAD datasets displayed in the CUAD paper [9] as shown in Table 4.2. The performance of BERT-based models on CUAD was defined by "precision at 80% recall" by the authors. They set a threshold on the model's confidence such that the model had 80% recall, and then analyzed the model precision at that specific threshold.

Given the design and limitations of the GPT 3.5 model, it does not directly output confidence scores for predictions like some BERT-based models. This poses a challenge when attempting to directly compare it to other models using metrics like "Precision @ X% Recall". However, by optimizing our prompt structures and experiment setups, we ensure that the GPT model achieves at least 80% recall for all predictions, mitigating this issue. This might not be a perfect solution and there remains differences on how each model handles uncertainty, but it's a reasonable approximation under the constraints.

The Tables 4.1 and 4.2 reveals the varied performance of our GPT-3.5 model based on the prompting strategy used. It is noteworthy that with few-shot prompting, the model achieves an average precision of 52.1%, recall of 82.1%, and an F1 score of 63.4%. This suggests that the GPT-3.5 model can achieve competitive, or even superior performance to the BERT-based models, depending on the examples provided in the prompt.

4.4 Performance by Category

In practice, models should not only have strong overall performance but should also exhibit effectiveness in handling individual label categories. Given the vast number of labels we're dealing with, it is essential to investigate the performance of our methods across these diverse categories.

We analyze the precision, recall, and F1-score across different label categories under zero-shot, one-shot, and few-shot prompting respectively. Tables 4.3, 4.4, and 4.5 display these metrics, respectively.

Overview: The table highlights a significant variation in performance across different label categories. Precision, recall, and F1-scores vary, with some labels achieving high precision, recall, and F1-scores, while others lag behind. While the precision tells us how accurate the model's annotations are, recall reveals if the model overlooks on any relevant labels while annotating.

While high precision in identifying certain labels is observed, there's substantial room for improvement for others. Labels that frequently appear across various legal contracts exhibit high performance, possibly due to the GPT-3.5 model's pre-existing knowledge aligning with the definition and labeling rules for these common labels, examples of which are "Document Name", "Parties" and "Governing Laws".

As we move from zero-shot to one-shot and few-shot prompting, most labels show a general enhancement in precision. This improvement indicates that the model is becoming more accurate in its annotations, reducing the instances of false positives. For labels initially scoring low on precision, increased F1 scores suggest that the model, with the help of additional examples in the prompts, is learning to better recognize these labels, resulting in a more accurate extraction of relevant contract clauses.

Comparing precision and recall: By further analyzing the tables, we observe a trade-off between precision and recall for certain categories. For instance, in labels such as "Anti-Assignment" and "IP Ownership Assignment", an increase in precision has led to a decrease in recall and vice versa. The observed phenomenon may arise from a challenge facing the model: reaching a delicate balance between inclusiveness (with the aim of capturing all relevant cases and increasing recall) and exclusiveness (aiming to limit incorrect labels, thereby raising precision).

The F1-score, being the harmonic mean of precision and recall, offers a balanced view of this trade-off. It becomes particularly useful in cases where one of precision or recall significantly outperforms the other. For instance, the "Most Favored Nation" label, despite having high precision, suffers from low recall, resulting in a lower F1-score. This suggests that while the model can accurately label instances

Label ID	Label Name	Zero shot	One shot	Few shot
1	Document Name	0.728	0.835	**0.880**
2	Parties	0.660	0.767	**0.894**
3	Agreement Date	0.256	0.308	**0.818**
4	Effective Date	0.111	0.117	**0.493**
5	Expiration Date	0.392	0.474	**0.632**
6	Renewal Term	0.362	**0.444**	0.392
7	Notice to Terminate Renewal	0.185	0.191	**0.405**
8	Governing Law	0.653	**0.760**	0.754
9	Most Favored Nation	0.333	0.415	**1.000**
10	Non-Compete	0.225	0.277	**0.500**
11	Exclusivity	0.350	0.432	**0.583**
12	No-Solicit of Customers	0.136	0.142	**0.571**
13	Competitive Restriction Exception	0.057	0.063	**0.398**
14	No-Solicit of Employees	0.261	0.313	**0.571**
15	Non-Disparagement	0.154	0.160	**0.330**
16	Termination for Convenience	0.097	0.103	**0.391**
17	ROFR/ROFO/ROFN	0.071	0.077	**0.387**
18	Change of Control	0.250	**0.302**	0.294
19	Anti-Assignment	0.587	0.694	**0.740**
20	Revenue/Profit Sharing	0.311	0.393	**0.542**
21	Price Restriction	0.000	0.000	0.000
22	Minimum Commitment	0.269	**0.321**	0.243
23	Volume Restriction	0.000	0.000	0.000
24	IP Ownership Assignment	0.140	0.146	**0.500**
25	Joint IP Ownership	0.130	0.136	**0.363**
26	License Grant	0.258	0.310	**0.537**
27	Non-Transferable License	0.089	0.095	**0.465**
28	Affliate IP License-Licensor	0.000	0.000	0.029
29	Affiliate IP License-Licensee	0.043	0.049	0.083
30	Unlimited/All-You-Can-Eat License	0.143	0.149	**0.363**
31	Irrevocable or Perpetua License	0.083	0.089	**0.459**
32	Source Code Escrow	0.000	0.000	0.000
33	Post-Termination Services	0.064	0.070	**0.208**
34	Audit Rights	0.289	0.341	**0.720**
35	Uncapped Liability	0.107	0.113	**0.387**
36	Cap on Liability	0.372	**0.454**	0.443
37	Liquidated Damages	0.146	0.152	**0.430**
38	Warranty Duration	0.160	**0.166**	0.100
39	Insurance	0.525	**0.632**	0.560
40	Covenant Not to Sue	0.069	0.075	**0.338**
41	Third Party Beneficiary	0.107	0.113	**0.200**

TABLE 4.3: Precision across different label categories under zero-shot, one-shot and few-shot prompting. The highest value for each label is bold, indicating the best performance achieved

of "Most Favored Nation" when it does recognize them, it struggles to identify all instances, thereby missing out on several potential correct labels. Conversely, for labels such as "Unlimited/All-You-Can-EAT", the model exhibits high recall but lower precision, still leading to a reasonably high F1-score.

Label ID	Label Name	Zero shot	One shot	Few shot
1	Document Name	0.797	0.904	0.973
2	Parties	0.969	0.969	1.000
3	Agreement Date	0.793	0.900	0.926
4	Effective Date	0.714	0.821	0.949
5	Expiration Date	0.861	0.968	0.915
6	Renewal Term	0.944	0.944	1.000
7	Notice to Terminate Renewal	0.833	0.940	0.900
8	Governing Law	0.907	0.907	0.961
9	Most Favored Nation	0.333	0.415	0.363
10	Non-Compete	0.750	0.857	0.909
11	Exclusivity	0.875	0.982	0.895
12	No-Solicit of Customers	1.000	1.000	0.820
13	Competitive Restriction Exception	0.333	0.415	0.895
14	No-Solicit of Employees	1.000	1.000	1.000
15	Non-Disparagement	0.667	0.774	1.000
16	Termination for Convenience	0.600	0.707	0.833
17	ROFR/ROFO/ROFN	0.143	0.149	1.000
18	Change of Control	0.583	0.690	0.853
19	Anti-Assignment	0.841	0.948	0.824
20	Revenue/Profit Sharing	0.700	0.807	0.887
21	Price Restriction	0.000	0.000	0.000
22	Minimum Commitment	0.333	0.415	0.838
23	Volume Restriction	0.000	0.000	0.000
24	IP Ownership Assignment	1.000	1.000	0.917
25	Joint IP Ownership	1.000	1.000	0.687
26	License Grant	1.000	1.000	0.866
27	Non-Transferable License	1.000	1.000	0.909
28	Affliate IP License-Licensor	0.000	0.000	0.520
29	Afiliate IP License-Licensee	0.500	0.607	0.620
30	Unlimited/All-You-Can-Eat License	1.000	1.000	1.000
31	Irrevocable or Perpetua License	0.400	0.482	0.877
32	Source Code Escrow	0.000	0.000	0.000
33	Post-Termination Services	0.300	0.382	0.909
34	Audit Rights	0.812	0.919	0.877
35	Uncapped Liability	0.429	0.511	1.000
36	Cap on Liability	0.667	0.774	0.846
37	Liquidated Damages	0.667	0.774	0.820
38	Warranty Duration	0.500	0.607	0.770
39	Insurance	1.000	1.000	0.933
40	Covenant Not to Sue	0.222	0.274	0.820
41	Third Party Beneficiary	0.750	0.857	1.000

TABLE 4.4: Recall across different label categories under zero-shot, one-shot and few-shot prompting.

In this case, the model effectively identifies most instances of this label, but its accuracy in labeling is compromised, resulting in a few false positives.

Improvement on specific labels: A core aspect of our prompt-tuning methodology involved crafting unique prompts for each label, especially those that were consistently misunderstood or misidentified by

Label ID	Label Name	Zero shot	One shot	Few shot
1	Document Name	0.761	0.868	0.924
2	Parties	0.785	0.856	0.944
3	Agreement Date	0.387	0.458	0.869
4	Effective Date	0.192	0.205	0.648
5	Expiration Date	0.539	0.637	0.748
6	Renewal Term	0.523	0.604	0.563
7	Notice to Terminate Renewal	0.303	0.318	0.559
8	Governing Law	0.760	0.827	0.845
9	Most Favored Nation	0.333	0.415	0.533
10	Non-Compete	0.346	0.419	0.645
11	Exclusivity	0.500	0.600	0.706
12	No-Solicit of Customers	0.240	0.249	0.674
13	Competitive Restriction Exception	0.098	0.110	0.551
14	No-Solicit of Employees	0.414	0.477	0.727
15	Non-Disparagement	0.250	0.265	0.496
16	Termination for Convenience	0.167	0.179	0.532
17	ROFR/ROFO/ROFN	0.095	0.102	0.558
18	Change of Control	0.350	0.420	0.437
19	Anti-Assignment	0.692	0.802	0.780
20	Revenue/Profit Sharing	0.431	0.529	0.673
21	Price Restriction	0.000	0.000	0.000
22	Minimum Commitment	0.298	0.362	0.377
23	Volume Restriction	0.000	0.000	0.000
24	IP Ownership Assignment	0.245	0.254	0.647
25	Joint IP Ownership	0.231	0.240	0.475
26	License Grant	0.410	0.473	0.663
27	Non-Transferable License	0.163	0.173	0.615
28	Affliate IP License-Licensor	0.000	0.000	0.054
29	Afiliate IP License-Licensee	0.080	0.091	0.147
30	Unlimited/All-You-Can-Eat License	0.250	0.259	0.533
31	Irrevocable or Perpetua License	0.138	0.151	0.602
32	Source Code Escrow	0.000	0.000	0.000
33	Post-Termination Services	0.105	0.118	0.339
34	Audit Rights	0.426	0.497	0.791
35	Uncapped Liability	0.171	0.185	0.558
36	Cap on Liability	0.478	0.572	0.582
37	Liquidated Damages	0.240	0.255	0.564
38	Warranty Duration	0.242	0.261	0.177
39	Insurance	0.689	0.775	0.700
40	Covenant Not to Sue	0.105	0.118	0.478
41	Third Party Beneficiary	0.188	0.200	0.333

TABLE 4.5: F1-score across different label categories under zero-shot, one-shot and few-shot prompting.

the model. This approach is a step beyond standard methods as it tailors the model's learning process to specific labels.

To investigate the performance of our label-specific prompts, we select five label categories that initially performed poorly as examples. Figure 4.1 shows the changes in precision for these labels under the

different prompting methods, as well as their performance when using the DeBERTa-xlarge model from the CUAD authors

From this figure, a clear improvement in precision across these label categories is apparent as we progressed from zero-shot to few-shot prompting. This improvement signifies that few-shot prompting, particularly when using label-specific prompts, substantially enhances the model's ability to identify and extract clauses related to uncommon and complex labels.

Furthermore, a comparison with the DeBERTa-xlarge model reveals that our approach led to a marked improvement in precision for label categories that were proved challenging for the DeBERTa-xlarge model according to the experiments of the CUAD authors. This is a significant breakthrough, as it demonstrates successful progress towards one of our research objectives: improving precision for label categories that the prevalent BERT-based models found problematic, thus offering more accurate contract review outcomes.

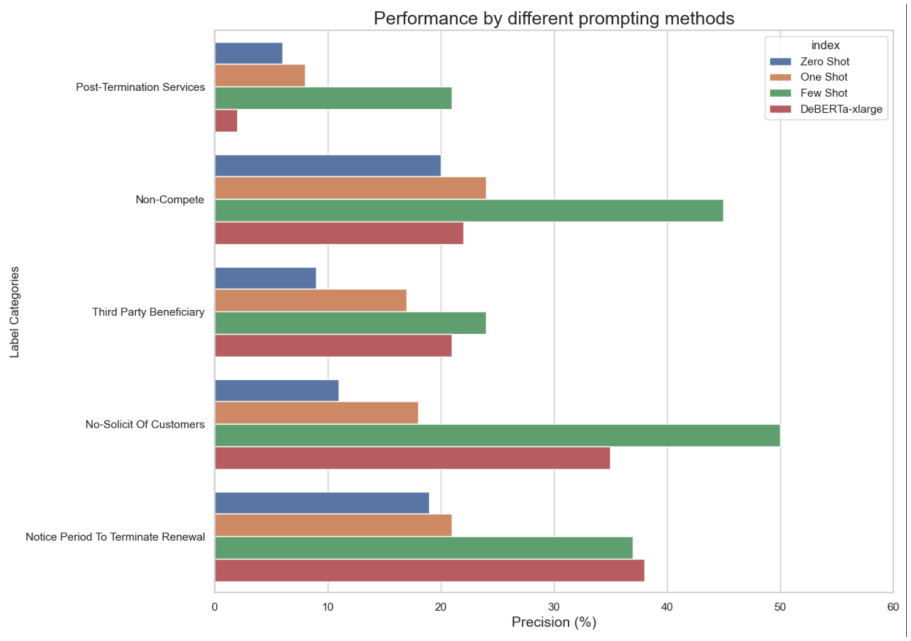


FIGURE 4.1: Performance on labels with poor performance by different prompting methods

In conclusion, our analysis demonstrates the impressive effectiveness of our label-specific prompting methodology when combined with the GPT-3.5 model. It highlights the benefits of incorporating more examples and specific instructions into prompts, especially for complex or unique contract clauses. However, given that several labels still have relatively low precision, there remains room for further optimization in these label categories.

4.5 Error Analysis

As proposed by the CUAD authors in the handbook, it is important that an AI tool developed using CUAD does not miss a clause that is responsive, which means it should have 80%+ in recall. Consequently, We achieve this by constantly refined our prompts for each label. However, as shown in Figure 4.2, While most labels reach 80%+ in recall,a few labels exhibit lower recall. We then conduct an error analysis on these outliers. By delving into the validation results for each label in detail, we aim to figure out the reasons behind these lower recall rates.

Table 4.6 and 4.7 displays the details of the task result upon label with low recall such as "Most Favored Nation" and "Joint IP Ownership". "Number of Answerable Questions" indicates the number of documents that include relevant clauses to the label.

The low recall for these labels might be due to several reasons. Given the fact that only three documents contained relevant clauses for them, it becomes challenging for the model to correctly identify them, especially if those clauses differ significantly from the examples. Besides, these uncommon labels have complex definition and labelling rules compared with others, influencing the model's ability to identify them correctly. Another possible reason might be the low number of instances of these labels in the test dataset makes their recall very sensitive to small changes in the number of true positives or false negatives. As a result, the recall here may not reflect the performance of our prompts, indicating that future improvement of the model's performance on these labels might involve collecting more examples.

TABLE 4.6: Results for the label category 'Most Favored Nation'

Metric	**Value**
True Positives	1
False Positives	0
False Negatives	2
True Negatives	82
Number of Answerable Questions	3
Correctly Answered	1
Incorrectly Answered	0
Missed	2
Accuracy	0.9765
Precision	1.0
Recall	0.3333
F1 Score	0.5

TABLE 4.7: Results for the label category 'Joint IP Ownership'

Metric	**Value**
True Positives	2
False Positives	4
False Negatives	1
True Negatives	78
Number of Answerable Questions	3
Correctly Answered	2
Incorrectly Answered	0
Missed	1
Accuracy	0.9412
Precision	0.3333
Recall	0.6667
F1 Score	0.4444

TABLE 4.8: Results for the label category 'Warranty Duration'

Metric	**Value**
True Positives	3
False Positives	28
False Negatives	1
True Negatives	53
Number of Answerable Questions	8
Correctly Answered	3
Incorrectly Answered	4
Missed	1
Accuracy	0.659
Precision	0.097
Recall	0.750
F1 Score	0.171

TABLE 4.9: Results for the label category 'Affliate IP License-Licensor'

Metric	**Value**
True Positives	1
False Positives	34
False Negatives	1
True Negatives	49
Number of Answerable Questions	3
Correctly Answered	1
Incorrectly Answered	1
Missed	1
Accuracy	0.589
Precision	0.029
Recall	0.500
F1 Score	0.054

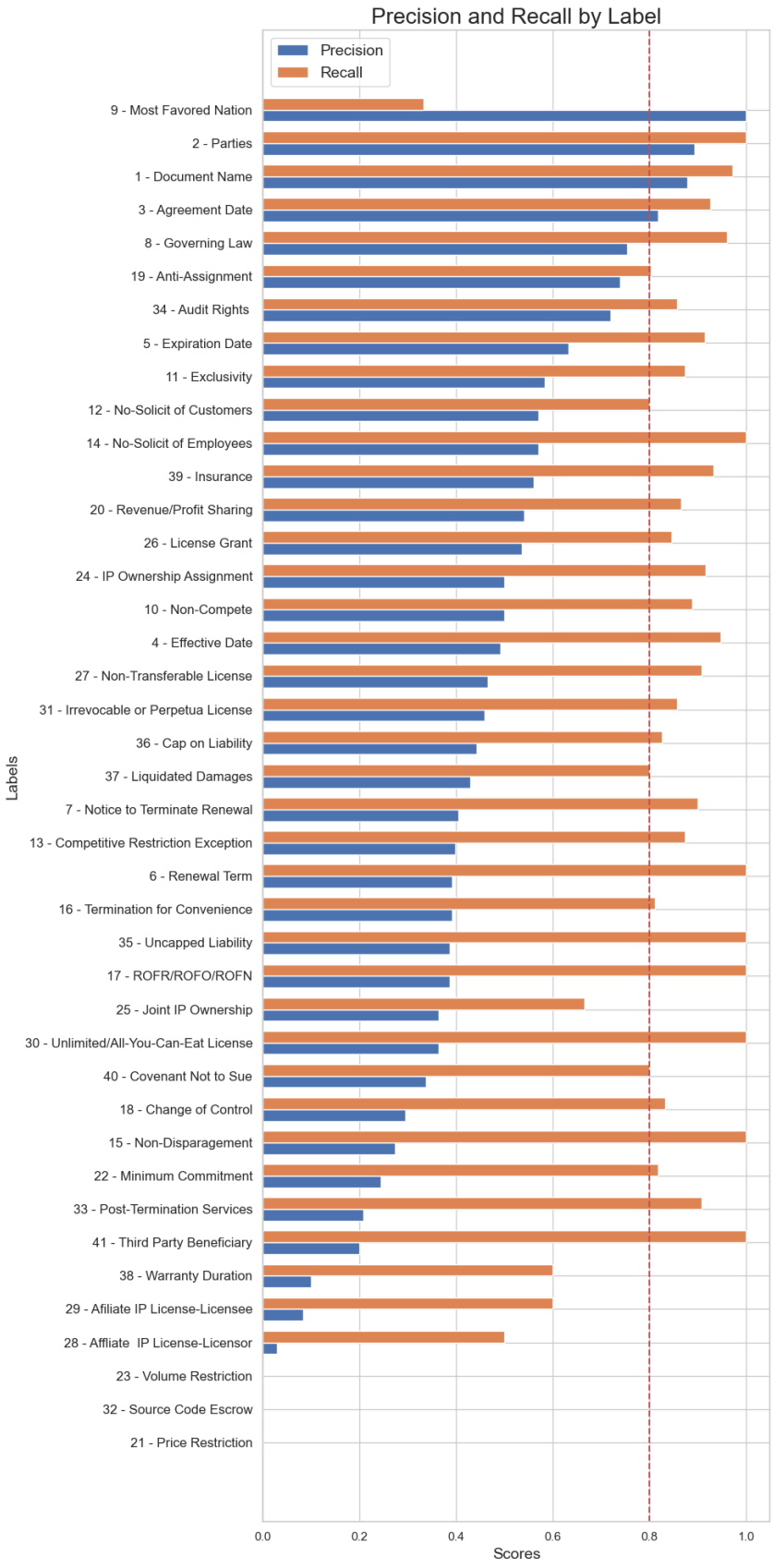


FIGURE 4.2: Performance of precision and recall by label. The last three labels have a precision and recall of 0 because there's no relevant clause to them in the test dataset according to the ground truth. We do not consider them here.

CHAPTER 5

Discussion

In this chapter, we conduct a critical examination of our study, recognizing its limitations and outlining potential directions for future research.

5.1 Limitations

Although the GPT 3.5-turbo model achieved impressive results on the CUAD dataset using zero-shot, one-shot, and few-shot prompting, it's important to acknowledge some limitations that arose during our research. These can provide valuable insights for potential refinement and improvement of our methodology.

Prompt Engineering Bias: The effectiveness of our approach using GPT models is highly dependent on the quality of the prompts, which could introduce a degree of bias. Creating effective prompts requires a comprehensive understanding of both the task and the model, often necessitating iterative tweaking and experimentation to find the optimal prompt for a specific task. Our prompts, designed and tuned based on specific labels and labeling rules, might not perform as well in different scenarios.

Input Length Limit: The current GPT-3.5 model has a maximum token limit of 4096 tokens, posing a challenge when dealing with large legal contracts that exceed this limit. Although we employed a sliding window approach to circumvent this limitation, there's still a risk of losing relevant contextual information due to the segmentation, potentially leading to less accurate identification and extraction of clauses.

Lack of Transparency: As previously mentioned, the GPT 3.5-turbo model does not provide confidence scores for each prediction. This complicates the task of establishing a performance baseline or implementing features like thresholding for model predictions. Furthermore, GPT and other deep learning models, including BERT and its variants, suffer from issues of interpretability. These "black box"

models do not offer clear insights into their decision-making processes. This lack of transparency could be particularly problematic in fields like law, healthcare, or finance, where understanding the rationale behind a decision and the certainty of a prediction is critical.

5.2 Future Work

This study provides a foundation for leveraging large language models for contract review tasks with promising results, leading to several potential future research.

Advanced Models: More powerful large language models, such as GPT-4, have been developed. These latest models may offer improvements in document understanding and lead to a better performance on contract review tasks. Furthermore, GPT-4 accepts a maximum input length of 32000 tokens. By breaking the input limit constraints, it could mitigate the risk of losing important contextual information and potentially lead to more accurate clause extraction.

Prompt Optimization: The performance of GPT models is heavily dependent on the quality of the prompts. Hence, future work could focus on optimizing the existing prompts. Many factors have not been explored during our current prompt design, such as the type and length of the contract, the format of relevant clauses, and the contextual information surrounding them. Further analysis of these factors could provide a more profound understanding of how to craft effective prompts for contract review tasks.

Fine-tuning GPT models: Similar to BERT-based models, GPT models also offer the potential for fine-tuning. This could enable the GPT models to better understand legal language and complex clauses, leads to better accuracy and performance. However, the fine-tuning process can be both time-consuming and computationally expensive. It remains a challenge for researchers to balance the model's pre-trained capabilities while adapting to the specific tasks.

CHAPTER 6

Conclusion

In this work, we present a comprehensive exploration into the application of large language models, specifically the GPT-3.5-Turbo, in assisting the contract review process. Our method with GPT models was evaluated against existing benchmarks set by BERT-based models on the CUAD dataset, demonstrating its superior performance on performing contract review tasks.

Our method exhibits a consistent pattern of improvement in performance across the zero-shot, one-shot, and few-shot prompting, highlighting the flexibility and potential of GPT models with effective prompts. Furthermore, through iterative prompt tuning, our refined few-shot prompting method not only outperformed DeBERTa-xlarge in overall precision, but also showed significant improvement on the identification and extraction of certain labels that proved challenging for BERT models.

Our findings emphasize the importance of prompt tuning and iterative improvement when employing large language models for complex tasks. This has implications for future research in the realm of AI and Law, especially in the development of more sophisticated methods of prompt design.

In conclusion, our study substantiates the competitive performance of GPT models and the potential of leveraging large language models for complex tasks such as contract review. While we acknowledge the limitations inherent to the utilization of GPT models, there are a lot of opportunities for future research to improve the model's accuracy and efficiency in the context of contract review and similar tasks.

Bibliography

[1] Srijan Bansal, Semih Yavuz, Bo Pang, Meghana Bhat, and Yingbo Zhou. 2023. Few-shot unified question answering: Tuning models or prompts?

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

[3] Harrison Chase. 2023. Langchain: Python library for language chain models. `https://python.langchain.com/en/latest/`. Accessed: 2023-06-14.

[4] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311v5*.

[5] IT Chronicles. 2023. Ai contracts: The future of contract law in the legal field. Accessed: 2023-05-29.

[6] Junyoung Chung, Caglar Gulcehre, and KyungHyun Cho. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[8] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*. ArXiv:2006.03654v6 [cs.CL].

[9] Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. 2021. Cuad: An expert-annotated nlp dataset for legal contract review.

[10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[11] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611. Association for Computational Linguistics, Vancouver, Canada.

[12] Juro. 2023. Contract ai: What is it and how does it work? Accessed: 2023-05-29.

[13] Yuta Koreeda and Christopher Manning. 2021. ContractNLI: A dataset for document-level natural language inference for contracts. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1907–1919. Association for Computational Linguistics, Punta Cana, Dominican Republic.

[14] Artificial Lawyer. 2020. Doc automation + fixed fees can drive law firm profits. `https://www.artificiallawyer.com/2020/05/04/doc-automation-fixed-fees-can-drive-law-firm-profits/`. Accessed: 2023-06-04.

[15] Spyretta Leivaditi, Julien Rossi, and Evangelos Kanoulas. 2020. A benchmark for lease contract review.

[16] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

[17] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work?

[18] Seonghyeon Moon, Seokho Chi, and Seok-Been Im. 2022. Automated detection of contractual risk clauses from construction specifications using bidirectional encoder representations from transformers (bert). *Automation in Construction*, 142:104465.

[19] James Mutinda, Waweru Mwangi, and George Okeyo. 2023. Sentiment analysis of text reviews using lexicon-enhanced bert embedding (lebert) model with convolutional neural network. *Appl. Sci.*, 13(3):1445.

[20] Tung M. Phung. 2021. A review of pre-trained language models: from bert, roberta, to electra, deberta, bigbird, and more.

[21] Renaissance Rachel. 2023. Prompting: Getting ai models to do what you want. `https://renaissancerachel.com/prompting/`.

[22] Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

[23] Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789. Association for Computational

Linguistics, Melbourne, Australia.

[24] Sunil Ramlochan. 2023. Master prompting concepts: Zero-shot and few-shot prompting. https://www.promptengineering.org/master-prompting-concepts-zero-shot-and-few-shot-prompting/. Accessed: 15-06-2023.

[25] Elvis Saravia. 2022. Prompt Engineering Guide. *https://github.com/dair-ai/Prompt-Engineering-Guide*.

[26] Jaromir Savelka. 2023. Unlocking practical applications in legal domain: Evaluation of gpt for zero-shot semantic annotation of legal texts. *arXiv preprint arXiv:2305.04417*.

[27] Soheil Tehranipour. 2020. Image in "openai gpt-3: Language models are few-shot learners". https://medium.com/analytics-vidhya/openai-gpt-3-language-models-are-few-shot-learners-82531b3d3122. Accessed: 2023-06-10.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

[29] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

[30] Sebastian Goodman Kevin Gimpel Piyush Sharma Radu Soricut Zhenzhong Lan, Mingda Chen. 2019. Albert: A lite bert for self-supervised learning of language representations.