ABSTRACT

Leveraging Large Language Models for Legal Document Understanding and Software System Analysis: Addressing Key Challenges

Ernesto Quevedo Caballero, Ph.D.

Co-Mentor: Pablo Rivas, Ph.D.,

Co-Mentor: Tomas Cerny, Ph.D.

In the rapidly advancing field of software development, ensuring compliance with legal regulations and policies has become increasingly critical. The intricate separation between legal expertise and software engineering creates challenges that demand robust, automatic compliance and auditing methods. Thesis focuses on leveraging Large Language Models (LLMs) to bridge this gap, particularly in two key areas: legal document question answering and classification and understanding complex software systems based on microservices architectures. By evaluating the capabilities of LLMs in these domains, this dissertation contributes essential insights into their potential role as auditors of legal compliance in software systems. Although this work covers only a portion of the broader vision of LLMs in full-spectrum compliance auditing, it lays the groundwork for a comprehensive approach by addressing the applications of LLMs in understanding legal documents and software systems.

However, a crucial challenge in deploying LLMs for legal compliance lies in their tendency to hallucinate. This limitation affects their application in legal document analysis, software understanding, and the broader goal of ensuring automatic compliance with legal regulations, where accuracy and reliability are essential, and errors can have significant consequences. Given that hallucinations in LLMs are a significant barrier to achieving automatic legal compliance in software, this thesis also addresses the hallucination problem in LLMs, ultimately contributing to the development of more reliable LLM-based tools not only for the main goal of this thesis but also for any domain where LLMs are utilized.

Hold for signature page

Leveraging Large Language Models for Legal Document Understanding and Software System Analysis: Addressing Key Challenges

by

Ernesto Quevedo Caballero, B.ENG, M.S.

A Thesis

Approved by the Department of Computer Science

Greg Hamerly, Ph.D., Chairperson

Submitted to the Graduate Faculty of
Baylor University in Partial Fulfillment of the
Requirements for the Degree
of
Doctor of Philosophy

Approved by the Thesis Committee

Pablo Rivas, Ph.D., Chairperson

Tomas Cerny, Ph.D., Co-chair

Greg Hamerly, Ph.D.

Eunjee Song, Ph.D.

Andrei Martinez-Finkelshtein, Ph.D.

Accepted by the Graduate School
January 14, 2025

J. Larry Lyon, Ph.D., Dean

Page bearing signatures is kept on file in the Graduate School.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I wish to express my deepest and most sincere gratitude to my advisors, Dr. Tomas Cerny and Dr. Pablo Rivas, whose unwavering guidance, relentless support, and mentorship have been indispensable throughout this dissertation and my academic life at Baylor University. Both are amazing professors, advisors, and researchers who helped me grow enormously in a short amount of time. Their wisdom and profound insights shaped the direction of this research and were instrumental in achieving success. Their commitment to academic excellence and rigorous scholarly inquiry has inspired me to aim higher, think more critically, and pursue knowledge with great dedication. I wouldn't be the researcher I am today if it were not for them.

I extend my heartfelt appreciation to the members of my dissertation committee. In particular, I would like to recognize Dr. Tomas Cerny, Dr. Pablo Rivas, Dr. Greg Hamerly, Dr. Eunjee Song, and Dr. Andrei Martinez for invaluable feedback, prompt engagement, and flexibility greatly enriched this research. Their expertise, thoughtful critiques, and guidance since the proposal were essential in helping me refine and elevate my work. Their collective support contributed to this dissertation's academic rigor and significantly influenced my growth as a researcher.

I owe an enormous gratitude to my labmates, including Amr Elsayed Abdelfattah, Alejandro Rodriguez, Jorge Yero, Korn Sooksatra, Bikram Khanal, Maisha Binte Rashid, and many others. The intellectual exchange and shared curiosity we cultivated within our lab profoundly impacted my research experience. The opportunity to collaborate with such talented individuals facilitated the success of numerous experiments and made the long hours spent in the lab far more enjoyable and fulfilling. Their insights, encouragement, and friendship transformed the lab into a dynamic and inspiring environment that fostered both personal and professional growth.

I also want to acknowledge the exceptional administrative support from Sharon Humphrey, Candace Ditsch, and Dr. Eunjee Song, whose guidance through the often complex procedural aspects of graduation was indispensable. Navigating the intricate requirements of the graduation process can be overwhelming. Still, their support ensured everything was completed smoothly and efficiently, allowing me to focus on my research and its final presentation without undue stress.

This thesis is dedicated to my grandmother, Daisy Rivero, whose dream was for me to become a PhD, just as she did. Sadly, she passed away a few months ago, but even though she won't be here to see it, I dedicate all of this to her, as she inspired me to pursue this path.

I would like to extend my deepest gratitude to my mother, who has supported me throughout my life, being present in every moment, and is why I am who I am today. I also want to thank Maribel, whom I love as a mother, for her unwavering support, love, and guidance.

Of course, I cannot express enough thanks to my partner in life, Maca, who embraced me during moments of immense stress, providing me with the strength to keep going and the love to find happiness, even in difficult circumstances, for her love, help, and support during this period I will be eternally grateful. I am deeply grateful to my closest friends in Waco, such as Marchena and Rafa, who have been exceptional friends, offering me unconditional support and help throughout these four years, which, thanks to them, were also very fun. I also want to thank other friends who shared this journey with me: Alejandro, Yero, Quintero, Benni, Betty, Valeria, Ana, Giselle, Diana, Laura, and more. The great moments we spent together over these four years made this time feel much closer to home.

To each of you, whether through mentorship, collaboration, family, friendship, or support, you have made an indelible impact on my academic and personal life. Your

contributions have been invaluable, and I am eternally grateful. Thank you for being part of this transformative and pivotal chapter in my life.

CHAPTER ONE

Introduction

The growing body of legal texts, including privacy policies, state regulations, contracts, and judicial rulings, contributes to an increasing workload for legal professionals, making many tasks repetitive and time-consuming. The complex and often ambiguous nature of legal language demands thorough analysis and understanding, which can be challenging even for experienced practitioners. For individuals without legal expertise, these documents are often incomprehensible. For instance, they routinely agree to or ignore terms in organizational security and privacy policies without fully understanding them. The specialized vocabulary in legal documents further complicates interpretation, making it challenging to address basic questions or understand the content Chalkidis and Kampas (2019); Rosili et al. (2021); Sansone and Sperli (2022).

On the other hand, modern society's reliance on software systems, from daily tasks to complex services, requires architects and developers to continuously update their knowledge to avoid technical debt and maintain system quality. The shift toward cloud-native designs, with decentralized microservices managed by various teams, has created complex environments that are difficult to manage holistically Parker et al. (2023); Abdelfattah (2022); Taibi et al. (2017); Abdelfattah and Cerny (2023). As these systems evolve independently, they risk degradation and pose challenges in maintaining cybersecurity, privacy, and legal compliance. Compliance with rules and regulations often becomes an afterthought, leading to costly post-development fixes and potential legal penalties, especially as systems and legislation evolve separately. Addressing legal compliance issues in software systems requires a thorough understanding of the associated business processes, structures, data flows, and policies. Designing systems

that are both aware of and adaptable to legal changes helps ensure their long-term viability and provides clear insights into the legal implications.

Recent developments in Natural Language Processing (NLP) have shown promise in improving and automating various tasks within the legal sector. In this context, the term "Legal NLP" has emerged, highlighting the specialized application of NLP techniques to legal documents and processes. Legal NLP researchers have explored diverse applications, including text classification, which helps organize legal documents but requires expert oversight to prevent misclassification. Summarization makes lengthy legal texts more accessible, though it needs careful review to avoid inaccuracies. Information extraction identifies key details in complex texts, benefiting from the structured nature of legal documents but still requiring validation. Question-answering tools offer initial guidance for legal inquiries; though they may struggle with complex questions, they are best used as supplementary resources. Information retrieval automates the search for legal documents but may overlook important contextual nuances. In most cases, the state-of-the-art is determined by applying Large Language Models (LLMs) Chalkidis and Kampas (2019); Chalkidis et al. (2021); Quevedo et al. (2023).

Additionally, investigators explored various aspects of privacy and software design, including consulting and legal counseling, developing privacy policies for vendors, and researching the implications of privacy measures on system security. Creating a cross-platform distributed system static code analysis to determine internal system structures, endpoints, and data flow holistically Cerny et al. (2020) through software architecture reconstruction Walker et al. (2021); Bushong et al. (2021). Their advancements were applied to assess systems for incompatible security policies Das et al. (2021) or poor design practices in code Walker et al. (2020).

Integrating Legal NLP techniques into the analysis and management of software regulations faces several critical challenges. While advancements in Legal NLP have

improved the ability to process and interpret legal texts, there is a pressing need to enhance these techniques for evaluating compliance with complex software regulations, including cybersecurity policies. Current Legal NLP tools and LLMs struggle with several issues in evaluating compliance in a software system: their effectiveness in addressing the nuanced requirements of software development regulations remains underexplored, their capability to handle intricate software projects through source code analysis is limited, and their reliability is compromised by the phenomenon of hallucinations where generated outputs may be misleading or incorrect Hadi et al. (2023); Ji et al. (2023); Jin et al. (2024).

Despite their impressive Natural Language Generation (NLG) capabilities, state-of-the-art LLMs exhibit significant limitations in comprehending and responding accurately to complex inquiries related to software projects. Their tendency to generate hallucinated content further exacerbates the problem, leading to potential inaccuracies in compliance assessments. These challenges highlight the need for a more robust methodology to enhance the effectiveness of Legal NLP tools, improve the capability of agents to assess software from multiple perspectives, and develop reliable techniques for detecting and mitigating hallucinations in LLM-generated content. Addressing these gaps is crucial for advancing the reliability and accuracy of automated compliance systems in the context of evolving software regulations.

1.1 Objectives

This dissertation aims to advance the application of cutting-edge Legal NLP techniques to legal regulations of software development, including creating new datasets for underexplored areas such as cybersecurity policies. It will also investigate how state-of-the-art LLMs can analyze software projects. Furthermore, the thesis addresses the critical challenge of hallucination detection in LLMs, which impacts various domains of artificial intelligence, including creating an assistant for automated compliance

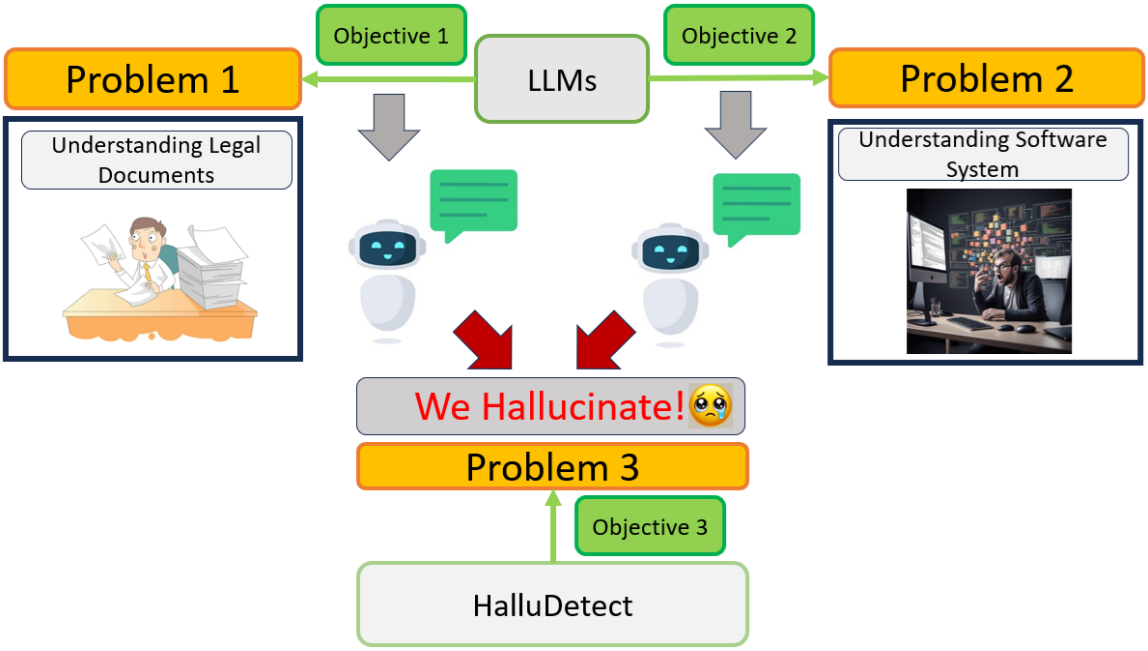

Figure 1.1. Problems and Objectives of the Thesis.

analysis of legal regulations in software systems. Figure 1.1 illustrates the problems and objectives that this thesis will address. The specific objectives are as follows:

(1) **Study the performance of state-of-the-art Legal NLP in legal regulations associated with software systems.**

Understanding legal documents associated with software is crucial for achieving automatic legal compliance. Question Answering is one key task for an agent to interpret legal documents and communicate their understanding. However, as discussed in Chapter 2 of this thesis, a significant challenge in Legal NLP is the limited availability of resources for Machine Learning approaches, particularly in software related legal documents such as cybersecurity policies Sansone and Sperli (2022); Rosili et al. (2021); Chalkidis and Kampas (2019); Quevedo et al. (2023).

To address these challenges, the thesis:

- Studies the performance of LLMs in question answering legal documents related to software.
- Introduces a new dataset focused on cybersecurity policies.

(2) **Assessing state of the art LLMs capability to comprehend and respond to complex software projects inquiries through source code analysis.**

A critical aspect of achieving automatic legal compliance in software is automatically understanding a software system, regardless of its complexity. Dr. Abdelfattah from Baylor University addressed this problem in his thesis from the Software Architecture Reconstruction (SAR) perspective Abdelfattah (2024). However, as discussed in Chapter 4, current state-of-the-art research has limitations in code summarization, automatic documentation, and software understanding for enterprise systems. Applying LLMs to understand complex software systems could produce promising results. Despite these advancements, the application of LLMs in this area still needs to be explored, particularly in extending their use beyond source code to include other intermediate system representations. Furthermore, there is a need for a question-answering tool that enables various users (without technical expertise or development knowledge) to ask questions directly about a software project and receive clear, natural language answers. Addressing these issues is essential for achieving the second objective of this thesis, which involves exploring software understanding through the application of LLMs, thereby supporting the main goal of automatic legal compliance in software Jin et al. (2024).

To address this problem, the thesis:

- Builds a question-answering tool based on LLMs over a complex software project.

- Evaluates such tools in well designed and challenging questions about the software system.
- Includes in the context of the LLMs the use of an intermediate representation of the source code, instead of the source code alone.

(3) **Detect hallucinations from LLMs generation using a token probability based approach.**

Both Objective 1 and Objective 2 address critical aspects of the overarching goal of achieving automatic legal compliance in software systems using LLMs. Improving LLMs is essential for understanding legal documents and software systems. As discussed before and deeply in chapters 6, 7 and 8 of this thesis, one of the major challenges with current LLMs is their tendency to hallucinate, which significantly hinders the success of an LLM agent being able to answer questions about its understanding of a legal document and software system. This issue also poses a substantial barrier to developing an effective tool for automatic legal compliance, should it rely on LLMs, as suggested by the findings of this thesis and existing literature Ji et al. (2023); Saxena and Bhattacharyya (2024).

To address the problem of hallucinations in LLMs, it is crucial to focus on their detection. Therefore, this objective of the thesis:

- Develops a supervised learning approach for detecting hallucinations in LLMs.
- Evaluate the performance of this approach, comparing it with state-of-the-art methods and highlighting its strengths and weaknesses.

The dissertation's objectives represent a significant advancement at the intersection of Legal NLP and software development, focusing on enhancing the

understanding and application of legal regulations in this domain. The first objective, which involves applying cutting-edge Legal NLP techniques to software development regulations, is crucial for bridging the gap between complex legal requirements and their practical implementation in software systems, as understanding the specific legal regulations that need to be assessed is a necessary precursor to evaluating compliance. Additionally, by creating new datasets, such as those focusing on cybersecurity policies, the research addresses an underexplored area in Legal NLP Quevedo et al. (2023); Chalkidis and Kampas (2019). This expansion of resources will facilitate further research into state-of-the-art Legal NLP methods, particularly in cybersecurity. Such advancements are essential for developing an automated legal compliance agent capable of detecting cybersecurity challenges in software based on relevant policies.

The second objective, investigating how state-of-the-art LLMs can understand and answer questions about software projects, is significant for advancing our ability to interpret and manage complex software systems. By leveraging advanced LLMs and intermediate representations of software projects, the objective will enhance the capacity of these LLMs to provide meaningful insights and answers from a software system. This objective is pivotal for developing tools that can automate and improve the quality of software maintenance and design validation, particularly in the scope of this thesis, the legal compliance assessment.

Furthermore, addressing the challenge of hallucination detection in LLMs is vital for ensuring the reliability and accuracy of LLM-driven solutions. Hallucinations, where models generate incorrect or nonsensical outputs, pose a significant risk in various LLM applications, including legal compliance analysis of software. By focusing on detecting hallucinations, the objective improves the robustness of LLMs-based agents, making them more reliable for critical tasks such as automated compliance analysis. This focus is essential for creating a dependable assistant to provide accurate and actionable insights into legal regulations governing software projects.

1.2 Contributions

This dissertation makes several significant contributions to Legal NLP, Software Development, and the domain of LLMs.

1.2.1 Study state-of-the-art approaches of Legal NLP in legal documents oriented to software development.

- **Study on the performance of state of the art LLMs in legal documents oriented to software development:** A study is conducted on the performance of several models, including BERT, LEGAL-BERT, ALBERT, DistilBERT, and RoBERTa, on the SQuAD and PolicyQA datasets. After comparing and analyzing the results, the best model for achieving optimal performance on the PolicyQA dataset using pre-trained models was identified. The findings reveal that these models perform significantly worse on the PolicyQA dataset than SQuAD V2.0. Surprisingly, general domain models like ALBERT and BERT outperformed the domain-specific LEGAL-BERT in this task.

- **Release the CSIAC-DoDIN V1.0 dataset, focused on cybersecurity policies, responsibilities, and procedures of the organizations involved:**

  Due to the lack of resources for Legal NLP, especially in software-based policies like cybersecurity, a new curated dataset is introduced, focused on cybersecurity-related policies and issuances developed by the DoD Deputy CIO for Cybersecurity. This thesis established a baseline using four classic transformer-based language models, BERT, RoBERTa, Legal-BERT, and PrivBERT, applied to multiclass classification and text co-occurrence tasks derived from the dataset. Additionally, the dataset and the code for training

and evaluating these baselines are openly accessible, facilitating further research and testing new or pre-trained models.

This dataset helps to address the resource gap in Legal NLP, particularly in the context of cybersecurity policies. Providing a structured and detailed collection of cybersecurity-related documents supports the development of more robust and accurate NLP models for policy analysis. Researchers and practitioners can leverage this dataset to advance their work in automated legal compliance, improve policy enforcement mechanisms, and explore new methodologies in NLP for cybersecurity.

1.2.2 Assessing state-of-the-art LLMs like ChatGPT capabilities to comprehend and respond to complex software project inquiries through source code analysis.

- **Create a set of practical system design questions to evaluate the service and interaction view perspectives:** This contribution involves developing a targeted set of questions designed to assess how effectively ChatGPT can understand and respond to queries about microservice systems. Using these questions, ChatGPT's ability to leverage source code to answer questions related to service and interaction views of microservices is evaluated. This evaluation helps determine whether ChatGPT's performance improves using source code versus intermediate representations and whether integrating both approaches enhances efficiency.

  This contribution is crucial for advancing automatic legal compliance in software systems, as it provides insights into how LLMs can be effectively used to understand and analyze complex software architectures.

1.2.3 A Token Probability Method for Detecting Hallucinations in Large Language Model Outputs.

This objective contributes to the goals of the thesis by addressing a critical challenge in deploying LLMs for legal compliance: the tendency of these models to generate hallucinations. By developing a method that leverages token probability to detect hallucinations, this objective enhances the accuracy and reliability of LLM outputs in general and, therefore, is also applicable in the context of legal document analysis and understanding complex software systems. This contribution is essential for ensuring that LLMs can be effectively utilized as auditors of legal compliance, where precision is vital. By mitigating the risks associated with hallucinations, the proposed method supports the broader vision of using LLMs for automatic legal compliance in software systems and other domains requiring high-stakes decision-making.

- **Propose a supervised learning approach using four features to detect hallucinations:** This thesis proposes a supervised learning approach using four features to detect hallucinations in conditional text generated by LLMs, achieving success with two classifiers: Logistic Regression and Simple Neural Network.

- **Evaluation and Validation:** Evaluate the performance of this approach across three datasets, comparing it with state-of-the-art methods and highlighting its strengths and weaknesses.

- **Model and Features analysis:** Investigate the impact of using the same versus different LLMs as evaluators for detecting hallucinations. Compare the performance of smaller LLM evaluators against larger ones, like LLaMa-Chat-7b. Perform a feature importance study through ablation.

These contributions advance the understanding of Legal NLP by evaluating the performance of state-of-the-art LLMs on software-oriented legal documents

and introducing a novel cybersecurity-focused dataset. Additionally, they enhance knowledge in software development by assessing ChatGPT's capability to handle microservice architecture queries using source code insights, providing a roadmap for integrating agents in the question-answering process related to software, and highlighting the challenges. Finally, they contribute to general NLP and the domain of LLMs by proposing and validating a token probability method for detecting hallucinations in LLM-generated text, considered one of the most significant challenges currently Ji et al. (2023); Saxena and Bhattacharyya (2024); Xu et al. (2024).

1.3 Thesis Organization

Chapter 2 provides a literature review to analyze Legal NLP research. Focusing on research problems, limitations, and areas for improvement. The chapter then elaborates on the taxonomy of Legal NLP tasks and the approaches taken in each one. Furthermore, it explores these techniques and the datasets that have been predominantly utilized over the years. It provides a comprehensive understanding of the historical and current state of research in Legal NLP.

Chapter 3 discusses the evaluation of state-of-the-art LLMs in the context of legal documents relevant to software development. This includes a comparative study of models such as BERT, LEGAL-BERT, ALBERT, DistilBERT, and RoBERTa using the SQuAD and PolicyQA datasets. It highlights the performance variations observed, with general domain models like ALBERT and BERT outperforming the domain-specific LEGAL-BERT on the PolicyQA dataset. Additionally, this chapter introduces the CSIAC-DoDIN V1.0 dataset, which focuses on cybersecurity policies. It provides a baseline for multiclass classification and text co-occurrence tasks, with open access to the dataset and code to support further research.

Chapter 4 provides a literature review on applying LLMs to improve automatic software understanding.

Chapter 5 evaluates ChatGPT's capability to understand and respond to questions about microservice architecture using source code insights. It details the creation of practical system design questions to assess ChatGPT's effectiveness in addressing service and interaction view perspectives. The chapter investigates whether ChatGPT performs more efficiently with source code than PO-CCG intermediate representation and explores the potential benefits of combining both approaches. Additionally, it includes a statistical analysis to determine whether ChatGPT's performance varies across different question categories when using source code versus PO-CCG.

Chapter 6 provides a literature review on techniques for hallucination detection in LLMs.

Chapter 7 explores a token probability method for detecting hallucinations in outputs from LLMs. It introduces a supervised learning approach that employs four features to identify hallucinations in generated text, demonstrating its effectiveness with classifiers such as Logistic Regression and Simple Neural Networks. The chapter evaluates and validates this method across three datasets, comparing its performance with state-of-the-art techniques and assessing its strengths and weaknesses. Additionally, it analyzes the impact of using different LLMs as evaluators for hallucinations, including a comparison between smaller and larger models. Finally, it shows a feature importance study through ablation.

Chapter 8 provides a literature review on methods to mitigate hallucinations in LLMs and a deep discussion of the problem, using all the knowledge presented in Chapters 6 and 7.

Chapter 9 concludes the dissertation by summarizing the key findings and contributions of the research. It reviews the thesis and suggests potential future studies in Legal NLP, Question-Answering systems for software development, and detecting and mitigating hallucinations in LLM outputs. The chapter aims to stimulate

further research in these areas, mainly focusing on Legal NLP applications in software development and addressing the challenges of hallucinations in LLMs.

CHAPTER TWO

Literature Review on Legal NLP

The work detailed in Chapter Two has been written mainly from the publication: Quevedo, Ernesto, Tomas Cerny, Alejandro Rodriguez, Pablo Rivas, Jorge Yero, Korn Sooksatra, Alibek Zhakubayev, and Davide Taibi "Legal Natural Language Processing from 2015-2022: A Comprehensive Systematic Mapping Study of Advances and Applications" IEEE Access (2023).

This chapter performs a literature review to provide descriptive statistics of Legal Natural Language Processing (Legal NLP) research. It categorizes and sub-categorizes key publications based on research problems, identifies limitations and areas for improvement, and highlights diverse methods used for tasks like Language Modeling, Multiclass Classification, Summarization, Information Retrieval, and Question Answering.

This literature review examines the primary methods employed in Legal NLP for various tasks and identifies the top-performing approaches. This analysis will provide the necessary insights to choose the most suitable methodology for conducting a state-of-the-art Legal NLP evaluation of software-related legal documents.

2.1 Taxonomy of Legal NLP tasks

To make the readability and clarity of the discussion easier, this section presents a category per Legal NLP task. The categories to be covered include:

Language Modeling (LM): predicts upcoming words from prior word context.

Multiclass Classification (Mult. Class.): in Machine Learning consists of classifying data instances into two or more selected classes.

Summarization (Sum.): in NLP is the task of producing a shorter version of one or several documents that preserves most of the input's semantics.

Information Extraction (IE): is the NLP task of extracting limited semantic content from text.

Question Answering and Information Retrieval (QA/IR): Question Answering is the NLP task where a given question is answered by using a set of documents as a knowledge base. Information Retrieval under NLP encompasses the retrieval of all media based on the user needs related to a topic.

2.1.1 Language Modeling

The language modeling task involves forecasting the next words based on the context of preceding words. Formally, given A sequence of words $w_1, w_2, ..., w_n$ drawn from a vocabulary V, where n is the length of the sequence, and V is the set of all possible words. Then, the objective is to estimate the joint probability distribution $P(w_1, w_2, ..., w_n)$ of the sequence. This joint probability can be decomposed using the chain rule of probability:

$$P(w_1, w_2, ..., w_n) = \prod_{i=1}^{n} P(w_i | w_1, w_2, ..., w_{i-1}) \tag{2.1}$$

Where $P(w_i | w_1, w_2, ..., w_{i-1})$ is the conditional probability of word w_i given the previous $i - 1$ words in the sequence.

Recent advancements in language modeling have been primarily driven by Neural Language Models Song et al. (2022), where embeddings of previous words capture the context. Notable high-performance models include transformer-based architectures such as Bidirectional Encoder Representations from Transformers (BERT) Devlin et al. (2018), Generalized Autoregressive Pretraining Model based on Transformer-XL (XLNet) Yang et al. (2019), and Generative Pre-trained Transformer (GPT) Radford et al. (2018).

In the context of language modeling for legal texts, Chalkidis et al. (2020) explored how to enhance BERT's performance within the legal domain. They introduced a specialized model, Legal Bidirectional Encoder Representations from Transformers (Legal-BERT), by fine-tuning the original BERT model on legal datasets. The data sources used for this fine-tuning included the official European Union Law database (EURLEX) [1], the UK's legislation database (LEGISLATION.GOV.UK) [2], the European Court of Human Rights case-law database (HUDOC) [3], the CASE LAW ACCESS PROJECT [4], and the SEC's EDGAR system [5].

Huang et al. (2021) developed a language model for legal documents utilizing GPT Radford et al. (2018). They replaced uncertain tokens with slots during training to address uncertainties, such as variable article numbers. Legal documents containing these slots were used to train the model, which was then enhanced with Key-Value Memory Networks and Transformer encoders to fill in the slots, treating the task as a question-answering problem. The model's performance surpassed existing models, assessed through F-score and perplexity metrics. Additionally, their slot-filling approach achieved greater accuracy compared to Memory Networks (MemNN) Weston et al. (2014) and Whoosh[6].

Zheng et al. (2021) introduced a new dataset called Case Holdings On Legal Decisions (CaseHOLD), which addresses a significant task for legal professionals and presents notable NLP challenges. They evaluated the performance of models on CaseHOLD and other legal NLP datasets. The findings suggested that domain-specific data pretraining could be advantageous if the task aligns well with the pretraining

[1] `http://eur-lex.europa.eu` Accessed on 02/04/2023

[2] `http://www.legislation.gov.uk`

[3] `http://hudoc.echr.coe.int`

[4] `https://case.law`

[5] `https://www.sec.gov/edgar.shtml`

[6] `https://pypi.org/project/Whoosh/`

corpus. The study found that task domain specificity improved performance on three legal tasks. Additionally, they compared SVM and BERT for an overruling task and found similar performance, though differences were more pronounced for more complex tasks.

In the research by Xiao et al. (2021), the authors introduced Lawformer, a pre-trained model designed for analyzing lengthy legal documents. Lawformer is based on the Transformer architecture adapted to manage long sequences, incorporating sliding window attention, dilated sliding window attention, and global attention mechanisms to achieve linear complexity for extended data. The study proposed the CAIL-Long dataset, an extension of CAIL2018 with average case lengths reflecting real world scenarios, and used additional datasets like LeCaRD, CJRC, and JEC-QA.

Qin et al. (2022), the performance of four pre-trained models was assessed on both general and legal domain corpora, focusing on classification accuracy across three Chinese legal document datasets: CAIL2018, CAIL-Long, and other datasets with varying text lengths. The study identified the leading language model based on the best performance in these evaluations. The leading language model showcasing the best results was:

Lawformer: which utilizes Longformer as a basic encoder and collects tens of millions of case documents published by the Chinese government for pretraining.

Legal-RoBERTa: same dataset as Lawformer, but with the Robustly optimized BERT approach (RoBERTa) architecture. The main limitations mentioned are long text, quadratic complexity of self-attention, and position embeddings not generalizable (like BERT, max 512).

The authors also noted a limitation where the classification effectiveness of Pretrained Language Models (LLMs) diminishes as the semantic complexity and composition of the documents increase. To address these issues, they propose

using transformer variants with attention mechanisms that have lower computational complexity.

Cui et al. (2023) introduce Chatlaw, an innovative legal assistant designed to improve the reliability and accuracy of AI-driven legal services. Chatlaw employs a Mixture-of-Experts (MoE) model with a multi-agent system. To ensure high-quality training, the authors have integrated knowledge graphs with artificial screening, resulting in a refined legal dataset tailored to the MoE model. This model leverages specialized experts to tackle various legal issues, enhancing the precision of legal responses. Additionally, the implementation of Standardized Operating Procedures (SOP), inspired by the workflows of real law firms, plays a crucial role in minimizing errors and reducing hallucinations in legal services. The authors report that their MoE model outperforms GPT-4, achieving a 7.73% increase in accuracy on the Lawbench and scoring 11 points higher on the Unified Qualification Exam for Legal Professionals. Furthermore, the model surpasses other models across multiple dimensions in real case consultations, demonstrating its strong capability for legal consultation.

2.1.2 Multiclass Classification

The multiclass classification task in Machine Learning consists of classifying data instances into two or more selected classes. Formally, given an input sequence of words $w_1, w_2, ..., w_n$ from the vocabulary V and a set of classes $C = \{c_1, c_2, ..., c_k\}$ where k is the number of classes. Then, it is assigned the input sequence $w_1, w_2, ..., w_n$ to a given class in C based on the estimated probabilities $P(c_i|w_1, w_2, ..., w_n)$ for $i = 1, 2, ..., k$. Usually, the optimization problem to solve is:

$$c^* = arg\ max_{c_i \in C} P(c_i|w_1, w_2, ..., w_n) \tag{2.2}$$

Where c^* is the predicted class for the given input sequence.

Similar to other NLP domains, several types of classifications that are desirable to automate for legal text are actively researched. The following key works in the Legal NLP field are presented.

In the study by Luo et al. (2017), the prediction of judgment outcomes for Chinese court cases was approached as a multiclass classification problem. Data was obtained from online court judgments available through China Judgement Online (CJO)[7]. The research utilized a two-stack attention mechanism for fact and dynamically generated article embeddings guided by fact-side clues. Word embeddings were generated using "Words to Vectors" (Word2Vec) Mikolov et al. (2013) on various legal texts. A noted limitation was the model's restriction to single-defendant cases, as multiple defendants complicate mapping facts to defendants.

The study by Shulayeva et al. (2017) focused on automatically identifying legal principles and facts in common law citations. The Naive Bayesian Multinomial Classifier was employed to classify features such as part of speech tags, unigrams, dependency pairs, sentence length, text position, and citation presence. A new corpus was introduced, derived from 50 British and Irish Legal Institute common law reports, with annotations for citation names and labels indicating whether sentences contained facts, principles, or neither. The corpus is available upon request.

Additionally, alongside Word2Vec embeddings Mikolov et al. (2013), the study investigated other pre-trained word embeddings such as FastText vectors [8] from Facebook AI Research and GloVe vectors [9]. The study also examined publicly available pre-trained word vectors derived from approximately 100 billion words in the Google News dataset [10].

---

[7] `https://wenshu.court.gov.cn` Accessed on 02/04/2023

[8] `https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md`

[9] `https://nlp.stanford.edu/projects/glove/`

[10] `https://code.google.com/archive/p/word2vec/`

From the comparisons, the most effective approach for automating legal document classification in this study was the combination of Convolutional Neural Networks (CNN) with word embeddings from a general domain (Google News), achieving 72.4% accuracy for 15 general categories and 31.9% accuracy for 279 more specific categories.

In the study by Zhong et al. (2018), the TOPJUDGE model was introduced for predicting judgments based on Chinese legal documents. The study proposed a unique multitasking approach using Directed Acyclic Graph (DAG)-based architectures for legal judgment prediction. This method was compared with baseline models, including CNNs and LSTMs with a softmax activation function. The authors also introduced new datasets: China Judgement Online (CJO)[11], Peking University Law Online (PKU)[12], and the Chinese AI and Law Challenge (CAIL)[13].

Consistent with previous approaches, Harkous et al. (2018) conducted a study to evaluate whether privacy policies effectively address users' general privacy concerns. They used the 115 Online Privacy Policies (OPP-115) dataset Wilson et al. (2016) and employed a neural architecture featuring neural embeddings, a Convolutional Neural Network (CNN), and dense layers with a classification head. Custom word embeddings specific to the privacy-policy domain, known as "Policies Embeddings," were trained on a corpus of 130,000 privacy policies from Google Play Store apps, reflecting the data practices of app companies. Additionally, Bag-of-Words techniques were used to represent judicial documents and extract features for further learning.

In the study by Fang and Zhao (2018), manifold learning-based dimensionality reduction methods were evaluated for classifying judicial documents. Their dataset,

[11] http://wenshu.court.gov.cn/

[12] http://www.pkulaw.com/

[13] http://cail.cipsc.org.cn/index.html

drawn from Aletras et al. (2016), included features from European Court of Human Rights (ECHR) case texts and utilized N-gram and topic models [14].

Various dimensionality reduction techniques were assessed, including autoencoders, factor analysis, Gaussian Processes Latent Variable Model (GPLVM), Isomap, Principal Component Analysis (PCA), its kernel and probabilistic variants, Landmark Isomap, Locally Linear Embedding (LLE), Multidimensional Scaling (MDS), Sammon Mapping, Stochastic Neighbor Embedding (SNE), Symmetric SNE, and t-Distributed Stochastic Neighbor Embedding (t-SNE). Methods such as Bagging, K-Nearest Neighbors (KNN), Logistic Regression, Random Forest, and Support Vector Machine (SVM) were examined for classification.

The authors identified a limitation in the typically smaller number of labeled judicial documents compared to their feature dimensionality, which can hinder prediction performance when using text features directly. The Bag of Words model, in particular, can generate many features, potentially impacting the performance of NLP algorithms. While linear dimensionality reduction techniques are anticipated to produce improved vectors with fewer dimensions, non-linear techniques were suggested to better preserve the distances between points in reduced dimensions.

In the study by Chalkidis et al. (2019), various neural models were evaluated using a newly introduced English legal judgment prediction dataset from the European Court of Human Rights (ECHR)[15]. The study presented the dataset, which achieved significant multiclassification results and highlighted the advantages of pre-trained language models (LLMs) over simpler approaches. The models assessed included Bidirectional Gated Recurrent Unit with Attention (BiGRU-Att) Xu et al. (2015), Hierarchical Attention Network (HAN) Yang et al. (2016), Label-Wise Attention Network (LWAN) Mullenbach et al. (2018), and BERT. A hierarchical variant of

[14] `https://figshare.com/s/6f7d9e7c375ff0822564`

[15] `https://archive.org/details/ECHR-ACL2019`

BERT, named HIER-BERT, was introduced to address BERT's limitations with long texts. HAN and HIER-BERT showed solid performance.

A separate study by Chalkidis et al. (2019) focused on Large-Scale Multi-label Text Classification (LMTC) within the legal domain. This research introduced a dataset comprising 57,000 legislative documents from EUR-LEX[16]. Techniques such as BERT, BIGRU-ATT, HAN, CNN-LWAN, BIGRU-LWAN, and their Zero-Shot variants like ZERO-CNN-LWAN and ZERO-BIGRU-LWAN were evaluated. The findings indicated that pre-trained language models (LLMs), particularly BERT, outperformed several other methods and achieved the highest results across most evaluation metrics.

The study by Noguti et al. (2020) focused on automating the categorization of petitions into their appropriate legal areas. The dataset used was from the "Public Prosecutor's Office of the Ministério Público" (PRO-MP), which includes public petition records from 2016 to 2019. It was carefully labeled by prosecutors from the "Ministério Público do Estado do Paraná" (MPPR). The study utilized standard text preprocessing techniques such as lowercasing, lemmatization, and punctuation removal. Texts were represented using Term Frequency-Inverse Document Frequency (TF-IDF) or word embeddings like Word2Vec, FastText, and Glove.

The authors evaluated various classification models, including Logistic Regression, SVM, Gradient Boosting, and different neural networks. Among these, Recurrent Neural Networks, especially LSTM, achieved up to 90% accuracy, exceeding human performance. However, the study needed a clear reference for the dataset used.

In the research by Jayasinghe et al. (2021), sentence embeddings were applied to multiclass classification to identify key sentences in legal cases. They used an adapted dataset based on the SigmaLaw ABSA Dataset from Mudalige et al. (2020).

[16] `http://nlp.cs.aueb.gr/software_and_datasets/EURLEX57K` Accessed on 02/04/2023

The BERT-cased model[17] was fine-tuned, with hidden states averaged and predictions made through a fully connected layer. A custom loss function was created to address the specific requirements of their classification task.

In the research conducted by Akça et al. (2022), the prediction of crime labels in Turkish court decisions was investigated. The authors developed supervised and unsupervised datasets and tested various models, ranging from traditional machine learning to transformers. Hyperparameters were optimized using grid search. The models compared included Naïve Bayes, Logistic Regression, SVM, Bidirectional Long Short-Term Memory (BiLSTM), Distilled BERT (DistilBERT), and BERT. Different word embeddings, such as Bag of Words + TF-IDF and FastText, were examined. Two datasets were utilized: an unlabeled collection for transformer pretraining and a labeled set of court cases, both sourced from Turkish legal documents.

Shankar et al. (2023) introduced PrivacyGLUE, a benchmark dataset for general language understanding in privacy policies. This dataset aggregates several existing privacy policy datasets and includes a comprehensive analysis and comparison of the performance of various transformer-based language models, such as BERT, RoBERTa Liu et al. (2019), Legal-BERT Chalkidis et al. (2020), and PrivBERT Srinath et al. (2020).

Song et al. (2022) introduced a novel legal extreme multi-label classification dataset, POSTURE50K, containing 50,000 legal opinions and their corresponding legal procedural postures. A deep learning architecture was proposed, featuring domain-specific pre-training and a label attention mechanism for multi-label document classification. The model was evaluated on both the POSTURE50K dataset and the EUROLEX57K dataset, achieving state-of-the-art results. The methodology employed was based on a RoBERTa-driven deep learning architecture, incorporating label embeddings and multi-task learning strategies.

[17] `https://huggingface.co/bert-base-cased`

Yang et al. (2016) mapped fact description sentences into latent spaces using a hierarchical BiLSTM encode. Key criminal elements, such as criminals and targets, were identified with the help of a reinforced criminal extractor. An element discriminator was also employed to differentiate law articles with similar TF–IDF representations. The proposed method's effectiveness for Legal Judgment Prediction (LJP) was validated through comprehensive experiments on the benchmark datasets CAIL-small and CAIL-big.

The empirical study by Song et al. (2022) demonstrated that the top-performing Pretrained Language Models (LLMs) were evaluated across four datasets for binary and multilabel classification tasks. The metrics used were F1, m-F1 (micro F1), M-F1 (Macro F1), W-F1 (weighted F1). Table 2.1 shows the models that get the best results in each dataset.

Table 2.1. Results from empirical study Song et al. (2022) with metrics: F1, m-F1 (micro F1), M-F1 (Macro F1), W-F1 (weighted F1).

Dataset	PLM	F1	m-F1	M-F1	W-F1
Overruling Zheng et al. (2021)	Custom LegalBERT Zheng et al. (2021)	0.973			
Terms of Service Lippi et al. (2019)	Custom LegalBERT	0.812			
POSTURE50K	BigBird Zaheer et al. (2020)				0.809
POSTURE50K	LightXML Jiang et al. (2021) + Custom LegalBERT		0.820		
POSTURE50K	LAMT_MLC Song et al. (2022)			0.263	
EUROLEX57K	LightXML + Custom LegalBERT		0.727		0.700
EUROLEX57K	LAMT_MLC			0.326	

The research of Liga and Robaldo (2023) proposes a Legal Rule Classification (LRC) task utilizing GPT-3. They train and test the LRC task on the General

Data Protection Regulation (GDPR) encoded in LegalDocML and LegalRuleML, both widely recognized XML standards within the legal domain. The authors use annotations to fine-tune GPT-3. Their findings demonstrate that LLMs can effectively learn to classify legal and deontic rules, even with limited data. Furthermore, GPT-3 significantly outperforms previous experiments on the same task. The study focuses on a multiclass classification, revealing GPT-3's capability to distinguish between obligation, permission, and constitutive rules, achieving superior performance compared to prior LRC efforts.

2.1.3 Summarization

Summarization in NLP is the task of producing a shorter version of one or several documents that preserves most of the input's semantics. Formally, given an input document D which consists of sentences $s_1, s_2, ..., s_m$ where each sentence s_i is a sequence of words $w_{i_1}, w_{i_2}, ..., w_{i_n}$ from the vocabulary V. The objective is to produce a concise summary S that retains the essential semantic information from D.

In the research by Polsley et al. (2016), TF-IDF and Part of Speech Tagging (POS-tag) were used to assign importance to different parts of a legal document. These weights were then used to create summaries of the documents. The dataset used in this study was obtained from the Federal Court of Australia[18].

In the research by Merchant and Pande (2018), legal text summarization was explored using latent semantic analysis. Singular Value Decomposition (SVD) was employed to pinpoint key sentences from singular vectors, selecting them based on their importance. The Bag of Word embeddings Le and Mikolov (2014) served as the vector representation of sentences. For the summarization tasks, criminal judgments were used in multi-document scenarios, while civil judgments were used for single-document tasks. The authors considered moving away from the Recall-Oriented Understudy

[18] `https://archive.ics.uci.edu/ml/datasets/Legal+Case+Reports`

for Gisting Evaluation (ROUGE) Lin (2004) method for evaluating summaries and discussed the potential application of their approach on mobile devices.

Tran et al. (2019) investigated retrieving legal cases as part of the Competition on Legal Information Extraction/Entailment 2019 (COLIEE 2019). They introduced a method called decided summarization, which combines lexical and latent features. The approach involves comparing a query case with potential candidates using different perspectives. Each query is represented by its summary and paragraphs, while each candidate is represented by its summary, the leading sentence of each paragraph, and the following paragraphs.

Anand and Wagh (2019) redefined legal document summarization as a binary classification task, where sentences are categorized as either essential or non-essential. Noting that many judgments come with a preliminary summary called a headnote, they developed a novel dataset generation method using these reference summaries. This technique eliminates the need for domain experts. Their approach enables the creation of legal document summaries without the need for complex feature engineering or specialized domain knowledge.

In their empirical study, Song et al. (2022) tackled the summarization task using the JRC-Acquis and BillSum datasets Kornilova and Eidelman (2019). The PLM that performed best across all metrics and test sets for the JRC-Acquis dataset was DYPLOC Hua et al. (2021). For the BillSum dataset, Global Aware Ma et al. (2021) achieved the top results in most metrics and test sets, although TextRank Mihalcea and Tarau (2004) excelled in the California test set.

2.1.4 Information Extraction

Information Extraction (IE) is the automated process of identifying and extracting structured information from unstructured text. This involves recognizing and retrieving specific data points, such as names, dates, or relationships, from a text corpus. IE aims to convert unstructured textual content into a structured format that

can be more easily analyzed and utilized. This process typically involves techniques from natural language processing and machine learning to extract relevant information systematically based on predefined criteria or patterns.

In the research by Dragoni et al. (2016), NLP techniques were used to extract rules from legal documents. These rules are represented as logical statements in $A \implies B$. The approach relied on the Stanford Parser[19] for parsing. WordNet was employed to handle variations in language, and logical dependencies were identified using the Boxer framework Curran et al. (2007).

Chalkidis et al. (2017) defined and automated extracting elements from contracts. They introduced a new dataset to support the creation of models for this extraction process. Two linear classifiers, Logistic Regression (LR) and Support Vector Machine (SVM), were evaluated using hand-crafted features, pre-trained word embeddings, and pre-trained POS tag embeddings. The best performance was achieved by a hybrid approach that combined machine learning (LR or SVM with the features and embeddings) with manually crafted post-processing rules. They released two datasets covering 11 contract element types: a labeled dataset with 3,500 English contracts and an unlabeled set with 750,000 contracts, both encoded to ensure privacy. These datasets are available online[20].

García-Constantino et al. (2017) introduced the Commercial Law Information Extraction based on Layout (CLIEL) system, which extracts information from legal documents regardless of format, structure, or layout. The system focuses on context and begins with a Rule-based Layout Detection (RLD) phase. This is followed by integrating a proposed Rule-based Layout Detection Tree (RLDT) data structure. In the RLD phase, document sections are annotated, extracted, and organized into the RLDT, allowing for structured storage of identified parts and entities for further

[19] `http://nlp.stanford.edu/software/lex-parser.shtml`

[20] `http://nlp.cs.aueb.gr/software_and_datasets/CONTRACTS_ICAIL2017/index.html`

processing. The study specifically targeted five types of data points: (i) "Date of document," (ii) "Name of party," (iii) "Name of counterparty," (iv) "Governing law," and (v) "Jurisdiction."

Ji et al. (2020) tackled information extraction from court records by framing it as a combined task of paragraph classification and sequence labeling, which is typical for Named Entity Recognition (NER). They employed a BiLSTM + Attention-based architecture with a shared core, which was trained jointly. This architecture featured two separate heads for the classification and labeling tasks and a final Conditional Random Field (CRF) layer. This approach achieved a 72% success rate in extracting legal evidence information, surpassing previous methods. A noted limitation was the extended length of legal documents, and to address this, the study suggested incorporating paragraph classification as part of future joint training efforts.

Yoshioka et al. (2021) achieved a significant breakthrough in the field by surpassing previous benchmarks in the statute law legal textual entailment task of the Competition on Legal Information Extraction/Entailment (COLIEE)[21]. They introduced an ensemble method based on BERT combined with data augmentation techniques. This approach assessed whether a given legal article supports a specified question statement. The authors constructed multiple fine-tuned BERT models and carefully selected the best model ensemble, accounting for the inherent variability in BERT fine-tuning and the nature of the questions.

Using their proposed method, Yoshioka et al. (2021) achieved an accuracy of 0.7037 on the statute law legal textual entailment task. Their implementation involved an ensemble of BERT models. For each question and legal article pair, they used a sentence-separator token ([SEP]) to concatenate the two texts before inputting them into the BERT model to determine if the article entailed the question (positive:1) or not (negative:0). In addition to their primary approach, they tested ten more

[21] `https://sites.ualberta.ca/~rabelo/COLIEE2021/`

models without data augmentation, calculating the average probability of positive and negative outcomes from these models. The dataset used was from task number 4 in COLIEE, known as the COLIEE Statute Law Task, and is available in both Japanese and English on the official COLIEE website.

2.1.5 Question Answering and Information Retrieval

Question Answering (QA) is a natural language processing (NLP) task designed to automatically generate precise and contextually relevant answers to questions posed in natural language. The process begins with question analysis, where the system interprets the intent and context of the input question. Next, document retrieval is conducted to identify and retrieve relevant documents or passages that might contain the answer. Finally, the system performs answer extraction or generation to pinpoint or construct the specific answer from the retrieved information. QA systems can handle a range of question types, from fact-based queries to more complex interrogatives, and typically operate on both structured and unstructured data sources.

Information Retrieval (IR) refers to locating and obtaining relevant information from a vast repository of data in response to a user query. This process involves several key stages: indexing, where a structured index of the content is created to facilitate efficient searching; query processing, which involves analyzing and interpreting user queries to understand their information needs; retrieval, where algorithms and models are used to search the indexed data and retrieve documents that match the query; and ranking, where the retrieved documents are ordered based on their relevance to the query using relevance scoring or ranking algorithms. IR systems are foundational to search engines, digital libraries, and other applications that require effective information retrieval from large datasets.

In QA and IR tasks, employing methods that can accurately capture text similarity is essential. This is illustrated in the work by Landthaler et al. (2016), where a word embedding technique was introduced. The goal was to measure the

similarity between two vectors: one representing the entire search by summing the word embeddings and another of the same size, which maintains the original word order through the summation process.

The authors employed Word2Vec for word embedding and used cosine similarity to assess the similarity between vectors. To compute the vector for a search, they summed the vectors of each word in the query. For the documents, they iterated over all the words and created a comparison vector using a window size of n/2, where n represents the length of the search query. These results were concatenated after identifying the top X results most similar to the query using cosine similarity. Subsequently, the selected words were shifted sequentially, and the similarity was recalculated to see if a better result could be found.

John et al. (2017) introduced a system designed to tackle Bar Examination questions written in Natural Language. They utilized a BiLSTM architecture enhanced with an Attention mechanism and Glove word embeddings for this task. The dataset for their study was sourced from the MultiState Bar Examination (MBE). However, the corpus used in their research is only accessible upon request.

Do et al. (2017) proposed methods for information retrieval and answering legal questions using the dataset from the Competition on Legal Information Extraction/Entailment (COLIEE) 2016[22]. They utilized six features: TF-IDF, Euclidean, Manhattan, Jaccard, LSI, and LDA for information retrieval. To rank the articles relevant to a given query, a Ranking SVM was trained on pairs of queries and articles using selected features. The trained SVM generated scores for each article relative to the query during inference. Experimental results demonstrated that a combination of LSI, Manhattan, and Jaccard produced the best outcomes.

Question Answering is treated as a form of textual entailment, approached as a binary classification task. Word embeddings are generated using Word2Vec

[22] `https://webdocs.cs.ualberta.ca/~miyoung2/COLIEE2016/` Accessed on 02/04/2023

with the Continuous Bag of Words (CBOW) technique. Sentence embeddings are then created by summing all word embeddings and normalizing by the number of words in the sentence. The embeddings for the question and article are combined and passed through a Convolutional Neural Network (CNN). The CNN's output is further enhanced by integrating Latent Semantic Indexing (LSI) and Term Frequency-Inverse Document Frequency (TF-IDF) features. These combined features are then processed through two perceptron layers for final output prediction. An ablation study revealed that incorporating LSI and TF-IDF with CNN's output results in the most effective performance.

The Legal Case Retrieval Task of COLIEE 2019 was tackled by Shao et al. (2020) by introducing a new BERT-based model, incorporating Paragraph-Level Interactions (BERT-PLI). This model focused on capturing interactions between paragraphs within case documents using BERT and then aggregating these interactions through sequential modeling to determine the overall relevance of the document. The experimental results demonstrated that this approach outperformed existing solutions at the time.

The methodology began by narrowing down the candidate set using BM25 rankings. To enhance the model's ability to understand the semantic connections between legal paragraphs, BERT was fine-tuned on an available entailment dataset specific to the legal domain before being integrated with BERT-PLI. This fine-tuning enabled BERT to infer better relationships between supportive paragraphs, which proved advantageous for the legal case retrieval task.

Kien et al. (2020) introduced a retrieval-based legal Question-answering model designed to learn attentive neural representations of both the input question and relevant legal articles. The authors validated the model's effectiveness by providing an annotated corpus and conducting experiments that compared their approach to other leading methods in the field. The model features two distinct encoders: a Sentence Encoder and a Paragraph Encoder. The Sentence Encoder employs word embeddings

and a CNN framework, while the Paragraph Encoder calculates a sentence's attention weight by averaging the attention weights of its words. Recognizing that not every sentence contributes equally to the paragraph's overall meaning, the authors replaced the traditional softmax function with sparsemax Martins and Astudillo (2016) to better handle this variation.

Chalkidis et al. (2021) tackled the Document to Document (Doc2Doc) problem, which is focused on Information Retrieval within the context of European Union (EU) and United Kingdom (UK) legislation. The task involves identifying relevant documents from one legislative body when a document from the other is used as a query. This problem is typically approached in two steps. The first step, document prefetching, involves retrieving the top k most relevant documents to maximize recall. Various algorithms are employed to generate document embeddings, including Best Match 25 (BM25), Words to Vector Centroids (W2VCent) Brokos et al. (2016), BERT, Sentence-BERT (S-BERT) Reimers and Gurevych (2019), Legal-BERT, and an ensemble method. Additionally, the study mentions C-BERT, a BERT variant pre-trained through a classification task based on the multilingual thesaurus maintained by the Publications Office of the European Union, known as EUROVOC[23].

Vold and Conrad (2021) describe implementing a RoBERTa Base question-answer classification model for practical use. In their study, they compared the performance of the RoBERTa-base classifier with a traditional machine learning model within the legal domain, specifically evaluating the differences between a trained linear SVM and results on the publicly available Privacy QA dataset. The findings revealed that RoBERTa outperformed the conventional SVM, showing a 31% improvement in F1-score and a 41% increase in Mean Reciprocal Rank.

Abualhaija et al. (2022) introduced an automated Question Answering (QA) system designed to assist requirements engineers in pinpointing legal text segments

[23]http://www.lt-innovate.org/lt-observe/resources/eurovoc-%E2%80%93-eus-multilingual-thesaurus

relevant to compliance requirements. Their approach utilized large-scale language models, including BERT, ALBERT, RoBERTa, and ELECTRA, all fine-tuned for QA tasks. The authors created a dataset with 107 questions and corresponding answers to evaluate their models. However, their publication did not provide a method for accessing this dataset.

Song et al. (2022) conducted an empirical study demonstrating that Custom LegalBERT achieved the highest accuracy in the Question Answering task. This notable performance is attributed to its domain-specific knowledge acquired through pre-training on 3.5 million U.S. cases. For the Information Retrieval task, evaluated using the COLIEE-2021 dataset[24], RoBERTa excelled in the Macro F1 metric, while LMIR Ponte and Croft (2017) delivered the best results for the Micro F1 metric.

2.2 Current Legal NLP Limitations

This section shows the main limitations highlighted in the literature regarding Legal NLP.

Fang and Zhao (2018) noted that the number of labeled judicial documents frequently needs to be higher than the feature dimensionality of these documents. This discrepancy can impair prediction accuracy when directly using these text features. To tackle this issue, they proposed non-linear dimension reduction techniques to preserve distances between data points in lower dimensions.

Pillai and Chandran (2020) identified two main limitations: the lack of standardized legal procedures across different countries and the limited availability of cross-country data for evaluating various legal texts. Additionally, they pointed out that legal texts often contain significant amounts of irrelevant information.

Zhong et al. (2020a) explored three main challenges in Legal Artificial Intelligence (AI), particularly within the Legal NLP domain:

[24] `https://github.com/sophiaalthammer/dossier_coliee`

Knowledge Modelling Legal texts are highly formalized and contain a wealth of crucial knowledge and concepts. However, many recent works need to utilize this extensive knowledge due to inadequate modeling of legal concepts across documents.

Legal Reasoning Legal reasoning differs significantly from general NLP tasks. It requires adherence to well-defined legal rules, which means NLP approaches must integrate these rules to align with legal reasoning requirements.

Interpretability Legal language is inherently complex, and any legal decision or prediction made by an AI system must be interpretable. This is essential for its application within the legal system, ensuring that decisions can be understood and justified.

Zhong et al. (2020a) raised ethical concerns regarding deploying high-performance Legal NLP algorithms. Such technology must avoid biases, racial discrimination, and difficult results to interpret or persuade individuals. The study stressed that advancements in Legal NLP should support legal professionals rather than replace them.

Ji et al. (2020) identified the extensive length of legal documents as a significant limitation and proposed a paragraph classification task to address this issue through joint training. Similarly, Shao et al. (2020) emphasized that, within the legal domain, relevance extends beyond mere topical relevance. Relevant cases often relate to the decision of the current case, involving similar circumstances and relevant statutes.

Shaheen et al. (2021) explored the limitations of legal resources in the context of Legal NLP, highlighting issues related to scarcity and multilingual constraints. They pointed out that it's more than just insufficient resources; many available resources are limited to a single language. Working with low-resource languages exacerbates the difficulties of achieving high performance in Legal NLP using current state-of-the-art methods.

Akça et al. (2022) highlighted the need for more resources in the Legal NLP field, noting issues such as missing benchmarks for specific tasks and a need for well-curated datasets across various Legal NLP subdomains. As discussed in works like Yoshioka et al. (2021), the complexity of legal language presents challenges for tasks such as Question Answering. Current state-of-the-art methods often need help to deliver optimal performance and results when dealing with complex questions.

Chalkidis et al. (2021) emphasized the limitations of multilingual legal resources, underlining the need for developing Legal NLP models in languages other than English. They discussed the difficulties of creating new datasets and resources in the legal field, regardless of language. Legal obstacles, such as copyright protections for essential documents like contracts and trade secrets, pose significant barriers to dataset creation. Additionally, bureaucratic hurdles often limit access to court decisions. The study also noted the need for more human evaluation in existing legal datasets, pointing out that resources like LexGLUE rely on ground truth labels automatically derived from sources like court decisions, which lacks a definitive and reliable quality benchmark.

Qin et al. (2022) identified a limitation where the classification efficacy of LLMs decreases as the semantic complexity and intricacy of documents increase, a challenge frequently encountered with legal documents. To address this issue, the authors suggested employing transformer variants that incorporate attention approximations to reduce complexity.

In a recent empirical study, Song et al. (2022) demonstrated the limitations of domain-specific Pre-trained Language Models (LLMs), such as Legal-BERT, due to variations in legal subdomains and language across diverse legal documents.

Cui et al. (2023) highlight that popular models such as ChatGPT and LLaMA, along with other general-purpose and law-specific LLMs, although powerful, are not immune to the hallucination problems of LLMs. The knowledge these models rely on can be incomplete or outdated, further exacerbating the risk of hallucinations. In legal

applications, where decisions can have profound consequences, the potential for these models to produce incorrect or misleading information presents a substantial risk.

Table 2.2 summarizes the limitations found in Legal NLP and possible solutions according to the literature reviewed.

Table 2.2. Legal NLP Limitations

Limitation	References	Proposed Solution
Lack of unified legal procedures across countries	Pillai and Chandran (2020); Shao et al. (2020)	Agree on a standardized legal procedure around the globe.
Knowledge Modelling	Zhong et al. (2020a); Shao et al. (2020)	Design a proper model to represent the knowledge across legal documents.
Legal Reasoning	Zhong et al. (2020a); Shao et al. (2020)	Include the rules already present in law when designing any solution.
Interpretability	Zhong et al. (2020a)	Explainability of the models used and a clear legal interpretation of any decision.
Ethical issues	Zhong et al. (2020a)	Inclusion of bias and fairness analysis and good interpretability.
Long Documents	Ji et al. (2020); Shao et al. (2020); Tran et al. (2020)	Paragraph classification task and train jointly in it.
Lack of resources	Song et al. (2022); Shao et al. (2020); Shaheen et al. (2021)	Data augmentation
Complexity and ambiguity of the Legal language	Song et al. (2022,?); Yoshioka et al. (2021); Shao et al. (2020); Ge et al. (2021); Lyu et al. (2022)	Trained LLMs in more legal complex data. More separation of concerns in the solutions.
LLMs effectiveness decreases when the semantic composition of the documents increases	Qin et al. (2022)	Using variants of transformers with approximations of attention with lower complexity.
Number of labeled judicial documents is less than the dimensionality of features	Fang and Zhao (2018)	Using non-linear dimension reduction techniques.
Lack of Transferability of Domain-Specific LLMs	Song et al. (2022)	Trained domain-specific LLMs in massive and more language-sparse legal data.
Hallucinations in LLMs	Cui et al. (2023)	High quality datasets, Mixture of Experts (MoE).

2.3 Discussion

The application of Legal NLP to software-related regulations needs to be improved, and several gaps and challenges need to be addressed for the field to progress effectively. One of the most pressing issues is the scarcity of well labeled

and curated datasets specific to various subdomains of legal language, particularly those intersecting with software regulations. While the development of large language models (LLMs) has achieved significant advancements, the performance of these models could be more consistent when applied to specialized legal tasks. This inconsistency highlights the limitations of current models, which may need to adequately capture the complexities and nuances of legal language regarding software-related regulations.

Moreover, domain-specific models like Legal-BERT offer a promising direction, but they often do not outperform general-purpose models such as BERT or RoBERTa across all legal tasks. This suggests substantial variation in the language used across different legal subdomains, which domain-specific models must still be fully equipped to handle. The lack of diverse and comprehensive datasets encompassing a wide range of legal subcategories exacerbates this issue, limiting the models' ability to generalize effectively across different regulatory contexts within the software industry.

Additionally, the complexity of legal language, particularly in the context of software regulations, poses a significant challenge for existing LLMs. The legal domain requires models that can navigate intricate regulatory language, interpret ambiguities, and handle the voluminous and often dense documents typical of legal texts. However, the current trend in the NLP community leans heavily towards developing larger models with more parameters, often at the expense of incorporating domain-specific knowledge. While successful in many domains, this approach may need to address the unique challenges posed by legal texts fully and, therefore, the regulations associated with software systems.

Another critical challenge that has gained increasing attention is hallucinations in LLMs, particularly in the context of Legal NLP. Hallucinations refer to instances where models generate plausible-sounding text or information that is factually incorrect or completely fabricated. In the legal domain, hallucinations can have severe implications. Misinterpretations or fabricated legal references can lead to incorrect

legal conclusions, potentially inaccurately influencing software compliance decisions and regulatory interpretations.

The impact of hallucinations in Legal NLP goes beyond just the generation of incorrect information; it undermines trust in the technology, particularly in high-importance areas such as legal compliance and regulatory analysis in software systems. This challenge emphasizes the need for more robust mechanisms to detect and mitigate hallucinations, including integrating external knowledge bases, enhanced model interpretability, and specialized fine-tuning techniques that align model outputs more closely with established legal principles and facts.

The field would benefit from developing more refined and specialized legal language models trained on a broader and more diverse array of legal datasets to address these challenges. Furthermore, integrating symbol-based approaches could provide significant benefits, particularly in knowledge modeling, legal reasoning, and interpretability. These methods could complement existing LLMs by enhancing their ability to understand and process the complex, rule-based nature of legal language, thereby improving their performance in tasks related to software regulation. As the field continues to evolve, a more balanced approach that combines the strengths of both large-scale neural models and symbolic reasoning techniques may be essential for advancing the application of Legal NLP in software-related regulations, especially in mitigating the risks posed by hallucinations.

2.4 Datasets

This section shows the list of available resources found during this literature review. Table 2.3, which maps every dataset extracted from the previous papers to its respective Legal NLP task, along with references to where each resource can be found. However, it's important to note that many of these studies have utilized

resources provided by the Competition on Legal Information Extraction/Entailment (COLIEE) [25].

2.5 Limitations

Due to the specific inclusion criteria, the selection of works may have inadvertently excluded some relevant studies, potentially leading to an incomplete or skewed representation of the field. This challenge is compounded by publication bias, as many studies focus on positive outcomes, while negative results or failed experiments are often underreported, further narrowing the scope. Additionally, the fast-paced evolution of Legal NLP means that some of the findings in this review may quickly become outdated as new developments emerge.

Additionally, the chapter prioritized the tasks that were more related in the earlier stage to developing an agent capable of understanding, answering questions, and mapping knowledge from legal documents. However, the chapter does not explore other Legal NLP tasks, such as Coreference-Resolution and the Multilingual perspective.

The conclusions of this literature review are focused on Legal NLP and cannot be generalized to other areas of NLP. Although some of the limitations and challenges identified also apply to NLP, the chapters must attempt to make such broad generalizations. Additionally, since the results presented from multiple primary studies were not validated, the limitations present in those studies may also be reflected in this chapter.

Another significant limitation stems from the diverse evaluation metrics employed across studies. The variability in datasets, baselines, and performance measures complicates direct comparisons, making it difficult to draw uniform conclusions. Furthermore, Legal NLP is inherently interdisciplinary, combining legal and computational perspectives, and this review may not have fully captured the nuanced legal reasoning required for practical applications, placing greater emphasis

[25] `https://sites.ualberta.ca/~rabelo/COLIEE2022/`

Table 2.3. List of datasets and their corresponding tasks, papers and access link.

Dataset	Legal NLP Tasks	References	Link
CAIL2018; CAIL-Long	LM	Xiao et al. (2021)	https://paperswithcode.com/dataset/chinese-ai-and-law-cail-2018
Case-HOLD	LM, IE	Zheng et al. (2021); Song et al. (2022)	https://paperswithcode.com/dataset/casehold
LexGLUE	LM, Mult. Class., IE, QA/IR	Chalkidis et al. (2021)	https://huggingface.co/datasets/lex_glue
SCOTUS	Mult. Class.	Undavia et al. (2018)	http://supremecourtdatabase.org
CJO	Mult. Class.	Zhong et al. (2018)	http://wenshu.court.gov.cn/
PKU	Mult. Class.	Zhong et al. (2018)	http://www.pkulaw.com/
CAIL	Mult. Class.	Zhong et al. (2018)	http://cail.cipsc.org.cn/index.html
OPP-115	Mult. Class.	Wilson et al. (2016)	https://www.usableprivacy.org/data
ECHR	Mult. Class.	Fang and Zhao (2018); Chalkidis et al. (2019); Ge et al. (2021)	https://archive.org/details/ECHR-ACL2019
SigmaLaw ABSA	Mult. Class.	Jayasinghe et al. (2021)	http://www.cs.ucy.ac.cy/gkapi/foss.html
Terms of Service	Mult. Class.	Lippi et al. (2019)	http://claudette.eui.eu/ToS.zip
DMOZ	Mult. Class.	Nokhbeh Zaeem and Barber (2021)	https://tinyurl.com/y43htvum
POSTURE50K	Mult. Class.	Song et al. (2022,?)	https://rb.gy/fzsp1
ILSI	Mult. Class.	Paul et al. (2022)	https://github.com/Law-AI/LeSICiN
Legal Cases from the Federal Court of Australia	Summ.	Anand and Wagh (2019)	https://archive.ics.uci.edu/ml/datasets/Legal+Case+Reports
BilSUM	Summ.	Kornilova and Eidelman (2019); Song et al. (2022)	https://github.com/FiscalNote/BillSum
LegalSUM	Summ.	Anand and Wagh (2019)	https://github.com/lauramanor/legal_summarization
Civil Trial Court Debate	Summ.	Duan et al. (2019)	https://github.com/zhouxinhit/Legal_Dialogue_Summarization
COLIEE Statute Law Task	IE	Song et al. (2022); Rabelo et al. (2019); Do et al. (2017); Yoshioka et al. (2021)	https://sites.ualberta.ca/~rabelo/COLIEE2021/
License texts	IE	Kapitsaki and Paschalides (2017)	http://www.cs.ucy.ac.cy/gkapi/foss.html
Contracts	IE	Chalkidis et al. (2017)	http://nlp.cs.aueb.gr/software_and_datasets/CONTRACTS_ICAIL2017/index.html
Query Generation	QA/IR	Locke et al. (2017)	https://github.com/ielab/ussc-caselaw-collection
EURLEX57k	QA/IR, Cross-Lingual	Song et al. (2022); Chalkidis et al. (2019)	https://paperswithcode.com/dataset/eurlex57k
ALQAC-2021	QA/IR	Tieu et al. (2021)	https://www.jaist.ac.jp/is/labs/nguyen-lab/home/alqac-2021/
CJRC	QA/IR	Duan et al. (2019)	https://paperswithcode.com/dataset/cjrc
JEC-QA	QA/IR	Zhong et al. (2020b)	https://jecqa.thunlp.org/
LeCARD	QA/IR	Ma et al. (2021)	https://github.com/myx666/LeCaRD
Bar Exam QA	QA/IR	Wyner et al. (2016)	http://www.kaptest.com/bar-exam/courses/mbe/multistate-bar-exam-mbe-change
CJO	QA/IR	Zhong et al. (2018)	https://wenshu.court.gov.cn/
CRC	CoRef. Res.	Pothong and Facundes (2021)	https://www.refworld.org/docid/3ae6b38f0.html
Luxembourg's Income Tax Law	CoRef. Res.	Sannier et al. (2017)	https://people.svv.lu/sannier/crossreferences/
License texts	CoRef. Res., Cross-Lingual	Kapitsaki and Paschalides (2017)	http://www.cs.ucy.ac.cy/gkapi/foss.html

on technical advancements over legal applicability. This is closely tied to the gap between theoretical progress and real-world challenges, where many academic findings may need more practical utility for deployment in legal practice or compliance systems.

2.6 Conclusion

The literature review presented in this chapter offers a comprehensive review of the current state of Legal NLP, highlighting both the progress made and the challenges that remain in the field. Through an analysis of diverse approaches used in various Legal NLP tasks such as Multiclass Classification, Language Modeling, Summarization, Information Extraction, Question Answering, and Information Retrieval, this study has mapped out the landscape of existing research and identified key resources, datasets, utilized in training machine learning models for legal applications.

The review shows the central role NLP is beginning to play in the legal sector, especially in automating tedious legal processes such as reviewing lengthy legal documents, retrieving relevant legal information, and evaluating contracts and privacy policies. These are areas where Legal NLP models hold the potential to drastically reduce the time and effort required by legal practitioners, providing significant efficiency gains.

However, the study also reveals that significant challenges persist, particularly in the processing of complex and lengthy legal texts. While pre-trained language models (PLMs) have revolutionized NLP, their application in the legal domain has exposed limitations.

General-purpose PLMs often fail to deliver accurate results for legal-specific tasks, while domain-specific PLMs need help transferring their learning across different legal subdomains. This highlights the need for more specialized models that can handle the intricacies of legal language while being flexible enough to apply to diverse legal contexts.

The lack of curated datasets and comprehensive ontologies further exacerbates these challenges, making it difficult to develop models that can generalize effectively across different legal scenarios. The privacy and ethical concerns inherent in the legal domain also pose significant barriers to progress. The digitization of sensitive legal data for NLP model training introduces risks related to data privacy. In contrast, biases in the models, whether related to gender, race, or other factors, raise critical ethical issues, particularly in applications like judgment prediction.

Hallucinations in LLMs present another layer of complexity. In the legal domain, where precision is crucial, hallucinations can have severe consequences. This further emphasizes the need for robust and reliable Legal NLP models that can be trusted in high-stakes environments.

The findings suggest that while areas within Legal NLP have seen considerable advancement and are ripe for tool development, the field must address its inherent difficulties to achieve meaningful progress. The creation of more curated datasets, the development of unified legal data accessible across multiple jurisdictions, and the integration of symbolic methods that leverage existing legal knowledge and rules are critical steps toward overcoming the current limitations.

While Legal NLP is on the path to achieving a level of automation that could transform legal practice, significant work remains. Addressing the challenges identified in this literature review, particularly those related to data availability, model reliability, and ethical considerations, will be essential to advancing the field and realizing its full potential in automating and enhancing legal processes, including the complete automation of legal compliance in software systems.

2.7 Credit

In this section, we will provide each author's contributions to the work presented in this chapter. For this, we will use the system CRediT(`https://www.elsevier.`

com/researcher/author/policies-and-guidelines/credit-author-statement) from Springer to make it easier.

Ernesto Quevedo Caballero: Conceptualization, Methodology, Validation, Formal analysis, Investigation, Resources, Data Curation, Writing-Original Draft, Writing-Review Editing, Visualization, Project administration.

Tomas Cerny: Conceptualization, Methodology, Validation, Investigation, Writing-review and editing, Visualization, Supervision.

Alejandro Rodriguez: Validation, Investigation, Data curation, Writing-review and editing, Visualization.

Pablo Rivas: Investigation, Data curation, Writing-review and editing.

Jorge Yero: Investigation, Data curation

Korn Sooksatra: Investigation, Data curation

Alibek Zhakubayev: Investigation, Data curation

CHAPTER THREE

Study the performance of state of the art of Legal NLP in legal regulations associated with software systems

The work detailed in Chapter Three has been written mainly from two publications: Caballero, Ernesto Quevedo, Mushfika Sharmin Rahman, Tomas Cerny, Pablo Rivas, and Gissella Bejarano "Study of Question Answering on Legal Software Document using BERT based models" In LatinX in Natural Language Processing Research Workshop 2022; Quevedo, Ernesto, Ana Paula Arguelles, Alejandro Rodriguez, Jorge Yero, Dan Pienta, Tomas Cerny, and Pablo Rivas "Creation and Analysis of a Natural Language Understanding Dataset for DoD Cybersecurity Policies (CSIAC-DoDIN V1. 0)" In 2023 International Conference on Computational Science and Computational Intelligence (CSCI), pp. 91-98, IEEE, 2023.

Considering all the background from Chapter 2, it is clear that many challenges block the perfection of automatic legal auditing of a software system. Among these challenges is that legal documents often require a legal background, which poses complications for individuals and software companies, particularly regarding software privacy policies and regulations.

With the rapid expansion of applications and extensive personal information monitoring, it's increasingly crucial to understand how data is managed, shared, and used. Companies are required to include this information in their privacy policies for each application.

Therefore, a high-performance Question Answering (QA) system tailored for legal documents, such as privacy policies and policy rules, can have significant practical applications. For instance, such a system could enable individuals to quickly verify if their questions are answered within lengthy documents before agreeing to terms and conditions related to a software application. An even bigger application is to use

such a system tailored to a software project and assess if the software automatically complies with policies and regulations in a question-answer form.

As the review of Chapter 2 shows, privacy policies have gained prominence in the digital age; the research community has introduced several datasets related to privacy policies in recent years. Despite this, empirical studies have revealed a need for more transferability between domain-specific language models within legal subdomains and those in more distinct subdomains Song et al. (2022). Models explicitly trained on privacy policies have yet to be evaluated for their effectiveness on other policies like cybersecurity policies. Privacy policies represent only a subset of the broader field of cybersecurity policies and practices aimed at defending against potentially disastrous threats. Existing datasets often focus solely on the policies themselves, needing more coverage of guidance, responsibilities, or procedural elements. Additionally, the need for more structured data to train deep learning models is a widespread issue in Legal Natural Language Processing (Legal NLP), and this challenge also extends to the domain software systems' legal regulations.

Chapter 2 also clarified that current state-of-the-art methods and results are being obtained by applying Large Language Models (LLMs) or their combination with other methods. This chapter will focus on applying LLMs of specific and general domains, comparing their performance on the question-answering benchmark datasets Standford Question Answering Dataset (SQuAD V2.0) and Policy Question Answering (PolicyQA). Since SQuAD V2.0 is general domain and PolicyQA is related to software systems, a comparison of how the models behave in general domain and domain-specific questions can be performed.

Additionally, this chapter creates the dataset CSIAC-DoDIN V1.0, which is used to evaluate the performance of a set of LLMs in the cybersecurity domain. This dataset is focused on the cybersecurity policies, responsibilities, and procedures of

the organizations involved. This first version offers classic Legal NLP tasks, such as several Multiclass Classification tasks and Text Co-occurrence.

This chapter is organized as follows: First, it presents the methodology, experiments, and results of state-of-the-art LLMs on the SQuAD V2.0 and PolicyQA datasets, including a thorough analysis and discussion. Next, it introduces a newly created dataset, CSIAC-DoDIN V1.0, detailing the methodology used in its creation. Finally, it presents the methodology, experiments, and results of state-of-the-art LLMs on CSIAC-DoDIN V1.0, along with an analysis and discussion.

3.1 LLMs evaluated on SQuAD V2.0 and PolicyQA

This section examined and compared the performance of various BERT-related models using two Question Answering datasets: SQuAD V2.0 is from the general domain, and PolicyQA pertains specifically to legal texts on privacy policies applied to software development.

The SQuAD V2 dataset is a reading comprehension resource comprising over 100,000 questions generated by crowdworkers based on a selection of Wikipedia articles. Each question is answered with a text segment from the corresponding passage Rajpurkar et al. (2016, 2018).

The PolicyQA dataset is a reading comprehension resource with 25,017 examples curated from a corpus of 115 website privacy policies. It includes 714 questions annotated by humans, covering a broad range of privacy practices Ahmad et al. (2020).

Both datasets are intended for extractive Question Answering, where the answer is a span of text extracted directly from the passage. Additionally, the passage may be unrelated to the question or may not contain the answer at all.

3.1.1 LLMs used

Several widely used BERT-related models are known for their strong performance on the SQuAD dataset, including ALBERT, RoBERTa, and the original BERT. Also, the DistilBERT model is a smaller and more efficient variant that maintains competitive performance while being more cost-effective and faster during inference Sanh et al. (2019). Additionally, the LEGAL-BERT model, a specialized version of BERT trained from scratch on legal documents Chalkidis et al. (2020) exists. These models are used to compare the performance in both datasets SQuAD and PolicyQA.

BERT, DistilBERT, and ALBERT were trained originally on BookCorpus [1] and English Wikipedia (excluding lists, tables, and headers).

RoBERTa was pretrained in the same data than BERT and also CC-News [2], OpenWebText [3] and Stories Trinh and Le (2018).

LEGAL-BERT was trained on documents of EU legislation from EURLEX, the UK legislation portal; 19,867 from the European Court of Justice (ECJ), also available from EURLEX; 12,554 cases from HUDOC, the repository of the European Court of Human Rights (ECHR); 164,141 cases from various courts across the USA, hosted in the Case Law Access Project portal; 76,366 US contracts from EDGAR, the database of US Securities and Exchange Commission (SECOM).

3.1.2 Experiments

Experiments using the pre-trained versions of the models BERT, ALBERT, RoBERTa, DistilBERT, and LEGAL-BERT are conducted. The performance in the

[1] https://yknzhu.wixsite.com/mbweb

[2] https://commoncrawl.org/2016/10/news-dataset-available/

[3] https://github.com/jcpeterson/openwebtext

Question Answering (QA) task was evaluated using the Exact Match (EM) and F1 metrics Jurafsky and Martin (2022).

The EM metric refers to the percentage of predictions that exactly match any of the True Answers(labeled datasets for QA can have more than one span of text that is accepted as a correct answer to a given question). On the other hand, the F1 metric is computed over the individual words in the prediction against the True Answer. The number of shared words between the prediction and the truth is the basis of the F1 score, which could be written as $F1 = \frac{2*precision*recall}{precision+recall}$.

The open-source code from Hugging Face's Transformers library [4] is utilized to conduct the experiments. Two notebooks for running experiments were created and made available in an open repository at [5].

The selected models were trained for 5 and 10 epochs on the PolicyQA and SQuAD V2.0 datasets. Table 3.1 compares the models across both datasets and epoch variations, using the EM (Exact Match) and F1 metrics for evaluation.

Table 3.1. Result of the models on SQUAD V2.0 and PolicyQA

Dataset Name	BERT		ALBERT		LEGAL-BERT		RoBERTa		DistillBERT	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
SQUAD V2.0 5 epochs	71.6	77.39	73.9	77.91	73.5	77.01	**76.9**	**80.1**	65.47	69.27
PolicyQA 5 epochs	**29.5**	56.11	28.76	**57.36**	28.08	54.66	27.23	54.88	25.42	52.34
SQUAD V2.0 10 epochs	71.7	75.39	72.71	**77.21**	71.5	75.30	**75.18**	74.30	64.79	68.93
PolicyQA 10 epochs	29.6	57.02	**29.7**	**58.43**	28.45	55.01	27.85	54.91	25.81	52.48

3.2 Creation and Analysis of a Natural Language Understanding Dataset for DoD Cybersecurity Policies (CSIAC-DoDIN V1.0)

This section details the development of the CSIAC-DoDIN V1.0 dataset, which concentrates on cybersecurity policies, responsibilities, and procedures within

[4]https://github.com/huggingface/transformers

[5]https://github.com/Fidac/Legal-SE-BERT-Study

organizations. This initial version includes traditional Legal NLP tasks, such as various Multiclass Classification tasks and Text Co-Occurrence analyses. Additionally, it presents a baseline for this dataset and its associated tasks through experiments utilizing established transformer-based language models, including BERT, RoBERTa, Legal-BERT, and PrivBERT.

The CSIAC-DoDIN V1.0 dataset focuses on the Department of Defense (DoD) cybersecurity policies, given the DoD's strategic significance and extensive global operations, which demand rigorous and comprehensive cybersecurity measures. The dataset includes detailed DoD policies and policies from the National Institute of Standards and Technology (NIST), which are widely adopted in the industry. The dataset provides a broad perspective on cutting-edge cybersecurity strategies and methodologies, incorporating both DoD-specific and industry "best practice" policies and NIST and ISO-type guidelines. Including industry wide practices reflects how many companies model their cybersecurity and privacy policies after DoD standards.

As shown in Chapter 2, the research on benchmarks such as those by Shankar et al. (2023), and Chalkidis et al. (2021) has revealed significant disparities among legal subdomains, indicating that general-purpose models in specific tasks and datasets can outperform domain-specific models like Legal-BERT or PrivBERT. This highlights the critical need for multiple curated and labeled datasets across various legal subdomains, addressing the scarcity of legal domain datasets in recent literature Song et al. (2022); Shaheen et al. (2021).

While privacy policies have garnered substantial attention and resulted in numerous datasets, other policies and documents have been largely overlooked. A new dataset focused on cybersecurity policies is introduced to address this gap. This dataset includes a comprehensive collection of documents and guidance related to cybersecurity and establishes a baseline for further research in this area.

3.2.1 Dataset

For constructing this dataset, a chart that clusters and classifies Cybersecurity-Related Policies and Issuances developed by the DoD Deputy CIO for Cybersecurity [6] is used. This chart organizes cybersecurity policies and guidance documents according to Strategic Goals and the Office of Primary Responsibility. It captures a broad array of policies within a comprehensive organizational scheme, providing detailed clusters of documents and their content. By leveraging this knowledge base, the dataset encompasses a wide range of policies from a globally influential entity in cybersecurity.

The chart's careful organization ensures that the policies included are current and relevant. Additionally, the DoD policies featured in the chart incorporate guidance from the National Institute of Standards and Technology (NIST), which is widely adopted in industry. The chart also integrates specific DoD policies and best practice industry standards, such as NIST and ISO-type policies, which serve as benchmarks for cybersecurity and privacy policies in the industry.

3.2.2 Annotation Scheme

The annotation scheme for each document and policy is based on the organizational structure provided in the chart. In 2009, the Department of Defense (DoD) developed clusters to achieve specific goals as part of its information assurance strategy, ensuring appropriate unit-level capabilities. This scheme consists of seven outer clusters and twelve subclusters. Each outer cluster represents a main class to which a document belongs, while each subcluster provides additional categorization. Table 3.2 illustrates the subclusters corresponding to each outer cluster and their descriptions. The seven outer clusters are described as follows:

Organize: Policies in this cluster focus on how enterprise units (e.g., departments) should organize to achieve unity and purpose. These policies offer guidance to ensure that unit capabilities are designed, organized, and managed in a

[6] `https://dodiac.dtic.mil/dod-cybersecurity-policy-chart/`

Table 3.2. Subclusters corresponding to each cluster and its description

Subcluster	Outer Cluster	Description
Lead and Govern	Organize	Provide vision and follow through to set the enterprise direction, foster a culture of accountability, and provide insight and oversight for the enterprise.
Design for the Fight	Organize	Deliver, synchronize and integrate capabilities across the organization in time by shaping capabilities, engineering for the entire enterprise, leveraging technology, investing for success, and balancing risk.
Develop the Workforce	Organize	Provide a learning continuum to recruit, retain, and educate qualified professionals while keeping capabilities current through education and training, proper structure of the workforce, and the cultivation of awareness of initiatives.
Partner for Strength	Organize	Leverage the unique capabilities of partners from various areas such as intra-goverment, academia, cybersecurity and IT industry, defense industry, and international/global partners.
Secure Data in Transit	Enable	Provide robust, state-of-the-art cryptographic products and key management services for secure data transmission.
Manage Access	Enable	Provide secure, authenticated access to authorized users for proper visibility, configuration, connection, and allocation of resources through managed identity credentials, privileges, and resources.
Assure Information Sharing	Enable	Allow for secure and seamless information flow and management across security domains by assuring publishing, discovery, and collaboration.
Understand the Battlespace	Anticipate	Align and leverage information from audits, sensors, forensics, and incident management across an enterprise through knowing adversaries, networks, and consequences.
Prevent and Delay Attackers and Prevent Attackers from Staying	Anticipate	Leverage knowledge of networks, vulnerabilities, and adversaries to harden systems, defend perimeters, and assess defenses. Lower adversarial capabilities through detecting, diagnosing, eliminating, preventing, and constraining attacks.
Develop and Maintain Trust	Prepare	Guarantee integrity and availability of systems by assuring use, engineering for survivability, and maintaining integrity.
Strengthen Cyber Readiness	Prepare	Harden response procedures by linking units across the enterprise, stress testing response procedures, identifying critical assets, and improving continuity planning.
Sustain Missions	Prepare	Enable enterprise mission with limited interruption during an attack by assessment for fighting through adverse events, sustaining critical systems during degradation, and rapidly restoring systems to a trusted state.

way that they are synergistic, flexible, and dynamic. This allows them to effectively respond to various events and support the overarching objectives of the entire enterprise.

Enable: Policies in this cluster pertain to information access. They guide the ensuring that information is available to authorized parties while safeguarded from adversaries. The aim is to ensure that all units have proper visibility, control, and management of information assets, all within a secure framework.

Anticipate: Policies in this cluster focus on anticipating and preventing attacks on data and networks. They provide guidance on thwarting attacks from outside

the network perimeter while allowing for flexibility and maneuverability when necessary. Additionally, the policies offer guidance for secondary defenses in the event that a network perimeter is breached.

Prepare: Policies in this cluster focus on preparing and operating during a data breach or cyber-attack. They guide enhancing system resilience by ensuring cyber assets can self-monitor, self-attest, and self-repair. The policies ensure that units affected by a cyber-attack have assurances that the enterprise remains functional or has a plan in place in case of complete system degradation.

Authorities: This cluster of policies specifies the agency or authority the policy applies to (e.g., Department of Defense, United States Coast Guard). It also outlines the acting authority to be followed when multiple entities from different enterprises work together.

National/Federal: This cluster of policies pertains to national and federal regulations that offer guidance for actions at these levels.

Operational/Subordinate Policy: This cluster pertains to policies designed for specific entities, which may be used with overarching policies.

3.2.3 Extraction and Annotation process

Each annotator was assigned a set of outer clusters and followed the following protocol:

- Access the link on the chart and find the correct document.
- Once the document is found, determine if the document is guidance, strategy, or policy.
- If the document is guidance or strategy, extract the document PDF.

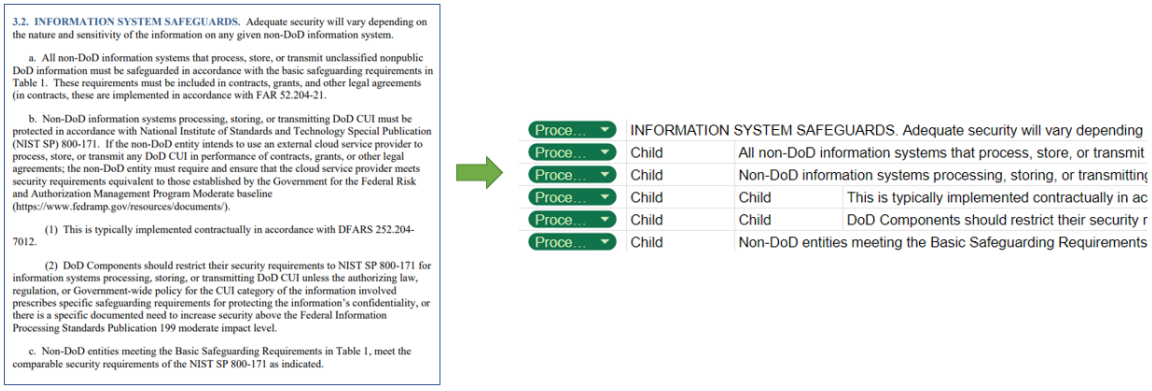

Figure 3.1. example of a set of procedures on the left extracted and annotated on the excel file on the right.

- If the document is a policy, then extract the PDF and also go through the document and extract every policy, responsibility, and procedure. Also, extract the general Purpose, Scope, and Applicability of the document.

- If the link wasn't accessible or there were any other reasons not to access the PDF, report it and ignore that document.

After analyzing the documents with a legal expert, it was observed that the policy documents were consistently structured into the following sections: Purpose, Authority, Scope, Policies, Responsibilities, Procedures, Definitions, and References. Based on discussions with the legal expert, the core content of the policy documents is primarily found in the Purpose, Scope, Policies, Responsibilities, and Procedures sections. Consequently, for this initial version of the dataset, the focus was on extracting text from these sections. However, the full documents are included in the dataset to ensure that any additional relevant information can be accessed. Each extracted item from a document is classified by type as a policy, responsibility, or procedure. Here is a more detailed description of each class:

Policy: A policy is designed to set parameters for decision makers while allowing for flexibility for decision makers.

Responsibility: Responsibilities within the policy designate which organization members are accountable to ensure the policy is adhered to.

Procedure: A procedure within the policy provides step by step instructions for performing a routine task.

The annotator would start by identifying and extracting each policy document's Policies, Responsibilities, and Procedures sections. Each item within these sections is then entered into an Excel file with a structured format. The Excel file includes several key fields: `Id` for a unique identifier, `Cluster` for the classification based on the organizational scheme, and `Classification` to specify whether the content is a Policy, Responsibility, or Procedure. Additionally, the file contains fields for `Purpose`, detailing the intent or objective of the policy, and `Scope and Applicability`, outlining the range and applicability of the policy. The `Type` field indicates the specific classification of the item, while `Text` contains the extracted text of the item. Multiple columns labeled `Child_Level#` are included to capture hierarchical or related sub-items. This methodical approach ensures that each item is accurately categorized and easily accessible for further analysis.

The rationale for including guidances, responsibilities, and procedures alongside policies is that these elements provide essential clarity on the implementation and execution of policies. Responsibilities outline the chain of command, ensuring that accountability is clearly defined in the event of an incident or failure to meet specific measures. Procedures offer a detailed, step by step guide on how to follow and achieve the goals outlined in the policies. Without such procedures, even well-crafted policies may be ineffective due to a lack of actionable directives. Additionally, incorporating guidances, responsibilities, and procedures enriches the context, which can enhance a LLMs ability to comprehend and interpret a policy and its broader implications.

3.2.4 Developed Legal NLP Tasks

This dataset was used to create Legal NLP tasks, including multiclass classification and text co-occurrence analysis. The specific tasks proposed are outlined below; however, the dataset supports a wider array of tasks beyond those listed.

Cluster Classification: Determine if a given policy, responsibility, or procedure belongs to a cluster.

Subcluster Classification: Determine if a given policy, responsibility, or procedure belongs to a subcluster.

Type Classification: Determine if a text is a policy, responsibility, or procedure.

Purpose-Text Co-Occurrence: Determine if a given policy, responsibility, or procedure co-occurs a Purpose of a document.

Scope/Applicability-Text Co-Occurrence: Determine if a given policy, responsibility, or procedure co-occurs with the Scope/Applicability of a document.

Text-Text Co-Occurrence: Determine if a given subpart of a policy, responsibility, or procedure co-occurs with another subpart of a policy responsibility or procedure.

The need for context relevance drives all Co-Occurrence tasks. This involves assessing how policies, responsibilities, and procedures align with the Purpose and Scope of a document. The Text-Text task aims to automatically identify whether a subitem within a policy, responsibility, or procedure has a semantic relationship with its preceding context. Another key focus is Verification and Comprehensiveness, which includes automated checks to detect misplaced policies, responsibilities, or procedures within a document. This also serves as a verification step before incorporating a new item, ensuring its alignment with the document's context.

Enhanced policy design is another major motivation. Identifying patterns and understanding which sections of a document frequently co-occur offer critical insights.

This allows policy designers to craft or refine policies more effectively, ensuring that all relevant components are comprehensively included. Additionally, cybersecurity policies' increasing complexity and volume highlight the need for Automated Recommendations. Large Language Models trained in co-occurrence tasks can assist professionals by suggesting suitable responsibilities or procedures that align with the goals or scope of a newly introduced or updated policy. They can also provide valuable insights into potential subitems for existing policies, responsibilities, or procedures.

3.2.5 Statistics of the Dataset

This section will outline the dataset's statistics and the different variants created based on the Legal NLP tasks.

Table 3.3. Documents distribution and extraction results by subcluster or cluster.

Subcluster/Cluster	Excels	Just PDF	Missing	Total
Lead and Govern	0	23	0	23
Design for the Fight	15	3	6	24
Develop the Workforce	8	2	4	14
Partner for Strength	4	3	3	10
Secure Data in Transit	17	2	3	22
Manage Access	13	3	8	24
Assure Information Sharing	6	0	0	6
Understand the Battlespace	3	5	0	8
Prevent and Delay Attackers and Prevent Attackers from Staying	10	9	7	26
Develop and Maintain Trust	4	0	2	6
Strengthen Cyber Readiness	9	7	0	16
Sustain Missions	10	2	8	20
Authorities	3	3	2	8
National/Federal	21	11	4	36
Operational/Subordinate	0	0	6	6
Total	123	73	53	249

Table 3.3 details the number of documents extracted by subcluster or cluster (if no subclusters are present) from the chart, including those with labeled text, those where only the PDF was extracted, and those inaccessible. Following the extraction, Table 3.4 presents a total of 7,698 examples in the dataset, categorized by type: Policy, Responsibility, and Procedures. The bar chart in Figure 3.2 illustrates the distribution of examples across the various clusters. It is important to note that each policy, responsibility, or procedure is assigned to a cluster.

Table 3.4. Dataset distribution by type

Type	Frequency
Policy	1531
Responsibility	4175
Procedures	1992
Total	7698

However, some examples are not assigned to a subcluster but only to an outer cluster. Table 3.5 provides the dataset's distribution by each subcluster. Notably, the Lead and Govern subcluster has zero text examples because all documents in this subcluster were strategy documents, which fall outside the current focus of the dataset.

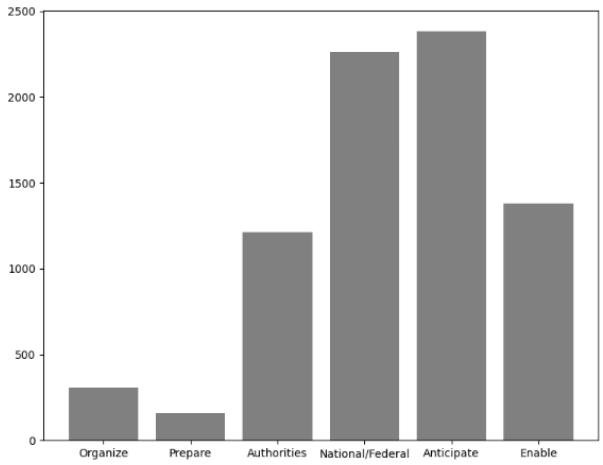

Figure 3.2. Distribution of examples in the dataset by cluster.

Additionally, Table 3.6 presents the number of positive examples available for each Text-Co-Occurrence task. Notably, some documents have fewer examples for the Purpose-Text and Scope/App-Text tasks due to the absence of Purpose or Scope/App sections.

Positive examples are derived from every tree constructed from a document's policy, responsibility, or procedure for the Text-Text Co-Occurrence task. Each edge in these trees represents a pair of Text-Text where the child co-occurs with the parent.

Table 3.5. Dataset distribution by subcluster

Subcluster	Frecuency
Lead and Govern	0
Design for the Fight	1002
Develop the Workforce	107
Partner for Strength	102
Secure Data in Transit	663
Manage Access	816
Assure Information Sharing	903
Understand the Battlespace	96
Prevent and Delay Attackers and Prevent Attackers from Staying	2165
Develop and Maintain Trust	43
Strengthen Cyber Readiness	151
Sustain Missions	112
Total	6160

In creating negative examples for all Text-Co-Occurrence tasks, the following strategy is followed:

(1) Randomly select the premise of a positive example.

(2) Choose the hypothesis of a different positive example that does not share the same premise.

(3) Generate a negative example using the randomly selected premise and hypothesis.

(4) Repeat this process until the number of negative examples matches that of positive examples.

Table 3.6. Positives examples in each Text-Co-Occurrence task.

Task	Frequency
Purpose-Text	7197
Scope/App-Text	6617
Text-Text	8355

3.2.6 Experiments and Results

This section discusses assessing five transformer-based language models that have set benchmarks for performance across numerous NLP tasks Bommasani et al. (2021). These models have been trained on extensive corpora of unlabeled text for Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) tasks. Each of these models is fine-tuned using the specific version of the dataset relevant to the task under evaluation.

BERT: Most popular transformer language model proposed by Devlin et al. (2018). It is trained in MLM and NSP tasks on the Wikipedia[7] and Bookcorpus Zhu et al. (2015) datasets.

RoBERTa: Liu et al. (2019) implemented the RoBERTa model to improve BERT using a larger vocabulary and a dynamic masking technique to eliminate the NSP task. It was pre-trained on the same datasets as BERT.

Legal-BERT: Is another BERT-based model by Chalkidis et al. (2020) pre-trained from scratch on English legal data consisting of contracts, legislation, and court cases. The original paper and the Hugging Face Model Card cite the data sources.[8] The sub-word vocabulary of Legal-BERT is built from scratch with additions to legal terminology.

PrivBERT: RoBERTa-based model proposed by Srinath et al. (2020). It was pre-trained from scratch on one million privacy policy documents.[9]

The pre-trained models used are available on Hugging Face.[10] Specifically, their base configurations feature 12 Transformer layers, 768 hidden units, and 12 attention

[7] `https://dumps.wikimedia.org`

[8] `https://huggingface.co/nlpaueb/legal-bert-base-uncased`

[9] `https://privaseer.ist.psu.edu/data`

[10] `https://huggingface.co/models`

heads. All models were trained using the Adam optimizer Kingma and Ba (2014) with a learning rate of 5×10^{-5} over six epochs. This duration was chosen because models typically reached optimal performance by the fourth or fifth epoch.

The dataset for the Multiclass Classification tasks is partitioned by randomly assigning 60% of examples per class to the training set, 15% to the validation set, and 25% to the test set. This distribution aimed to maintain balance and reduce the risk of overfitting to any class. In the Text-Co-Occurrence tasks, the datasets were already balanced; thus, the same percentage is applied: splits 60% for training, 15% for validation, and 25% for testing to ensure consistency and balance.

To evaluate the performance of all models across both Multiclass Classification and Text Co-Occurrence tasks, the micro-F1 (μ-F1) and macro-F1 (m-F1) metrics are used to address class imbalance. Additionally, for thoroughness and in line with the practices of several studies Shavrina and Malykh (2021); Chalkidis et al. (2021); Shankar et al. (2023), also arithmetic, harmonic, and geometric means across tasks are reported.

Table 3.7 displays the results of the models on all Multiclass Classification dataset variants, while Table 3.8 shows the results for all Text Co-Occurrence dataset variants. Furthermore, Table 3.9 provides the aggregated (averaged) results.

Table 3.7. Test results for all examined models across all Multiclass-Classification tasks.

	Type		**Cluster**		**Subcluster**	
Method	**μ-F1**	**m-F1**	**μ-F1**	**m-F1**	**μ-F1**	**m-F1**
BERT	0.963	0.95	0.96	0.921	0.911	0.734
RoBERTa	0.96	0.947	0.954	0.925	0.923	0.74
Legal-BERT	**0.969**	**0.96**	0.966	**0.954**	0.921	0.733
PrivBERT	**0.969**	0.959	**0.967**	0.938	**0.931**	**0.741**

Table 3.8. Test results for all examined models across all Text Co-Occurrence tasks.

Method	Purpose-Text		Scope/App-Text		Text-Text	
	μ-F1	**m-F1**	**μ-F1**	**m-F1**	**μ-F1**	**m-F1**
BERT	**0.749**	**0.735**	**0.581**	**0.503**	**0.715**	**0.699**
RoBERTa	0.539	0.427	0.494	0.397	0.607	0.556
Legal-BERT	0.738	0.721	0.574	0.495	0.688	0.666
PrivBERT	0.533	0.414	0.491	0.347	0.55	0.464

Table 3.9. Test results aggregated over all tasks: arithmetic (A), harmonic (H) and Geometric (G) mean.

Method	A-Mean		H-Mean		G-Mean	
	μ-F1	**m-F1**	**μ-F1**	**m-F1**	**μ-F1**	**m-F1**
BERT	**0.813**	**0.757**	**0.786**	**0.724**	**0.799**	**0.741**
RoBERTa	0.746	0.665	0.689	0.591	0.717	0.627
Legal-BERT	0.809	0.754	0.778	0.717	0.794	0.736
PrivBERT	0.74	0.643	0.676	0.549	0.707	0.595

Table 3.7 demonstrates that Legal-BERT and PrivBERT consistently outperformed other models across all Multiclass Classification tasks. This indicates that domain-specific models in the legal field have strong transferability to the domain of cybersecurity policies, surpassing general-purpose models in specific Multiclass Classification tasks. However, in Text-Co-Occurrence tasks, the BERT model outperformed all others across all metrics. The underperformance of RoBERTa and PrivBERT in these tasks may be due to their lack of pre-training on the Next Sentence Prediction (NSP) task. Despite this, BERT's superior performance to Legal-BERT suggests that domain-specific models like Legal-BERT and PrivBERT may struggle with more complex tasks when applied to different domains. These results show the importance of including NSP as a pre-training task for new transformer-based language models. Finally, the aggregated results in Table 3.9 show that BERT outperforms other models overall across all metrics. Although Legal-BERT's performance is close, the disparity between BERT and Legal-BERT is more pronounced in Text-Co-Occurrence tasks

than in Multiclass Classification tasks, where Legal-BERT has a slight edge over BERT.

3.3 Discussion

Let's start by discussing the implications of the CSIAC-DoDIN V1.0 dataset and the results obtained, emphasizing the specific challenges of cybersecurity policies.

Management research indicates that misunderstandings of cybersecurity-related policies can lead to severe consequences for organizations, such as increased risk of data breaches Schlackl et al. (2022), regulatory fines Haislip et al. (2021), erosion of consumer trust Goode et al. (2017), and a decline in firm value Cavusoglu et al. (2004).

Additionally, information security policies often need to be more complex, clearer, and consistent Cram et al. (2017). Studies reveal that there can be discrepancies between those involved in creating policies and those responsible for implementing them Schuetz et al. (2020); Cram et al. (2019). Additionally, the complexity of these policies can cause stress, uncertainty, and confusion among stakeholders, thereby increasing the likelihood of policy violations or improper adherence Milne and Culnan (2004); D'Arcy et al. (2014).

As organizations emphasize cybersecurity, they often need clearer policies from federal and state regulators, partners, and even within the organization itself. This makes it challenging for executives and managers to maintain, organize, and enforce compliance across multiple entities Cram et al. (2019). Lastly, many organizations need to be fully aware of the range of policies that may apply to the cybersecurity incidents they encounter, best practices for mitigating risks, and the correct steps to remediate the effects of a data breach. Consequently, this dataset serves as a valuable resource for academia and industry, enabling the development of innovative technologies to reduce misunderstandings, improve comprehension, and ensure proper adherence to cybersecurity policies.

Now, let's discuss this chapter's implications in the larger context of automatic legal analysis of software systems.

The experimental results highlight several key implications for automatic legal compliance analysis in software systems, particularly in using NLP models for tasks like Question Answering (QA) within legal domains. One surprising finding was that general-purpose pre-trained models like BERT and ALBERT outperformed LEGAL-BERT, trained explicitly on legal texts, in both the SQuAD V2.0 and PolicyQA datasets. This suggests that significant subdomain variations can affect model performance even within the legal domain. For example, PolicyQA focuses on software development and privacy in applications, a subdomain that might not be covered well by the legal texts used to train LEGAL-BERT. This highlights the importance of domain specificity in model training and suggests that models may need to be tailored more precisely to the specific legal subdomains relevant to software systems for accurate legal compliance analysis. Also, similar findings were obtained in the evaluation done with the CSIAC-DoDIN V1.0 dataset, where PrivBERT and Legal-BERT were not better than general-domain LLMs.

The findings obtained from the CSIAC-DoDIN V1.0 dataset show significant implications for legal compliance analysis. Organizations aiming to automate compliance checks using NLP tools must be cautious in selecting and fine-tuning models to ensure they are well-suited to the specific legal subdomains in which they operate. This could involve training models from scratch on datasets that are more representative of the legal issues related to software development and exploring ensemble methods to boost performance.

3.4 Limitations

First, studying this chapter is restricted to these specific models, omitting another transformer variant, such as GPT or T5, which may offer different insights. Additionally, focusing solely on the PolicyQA and SQuAD V2.0 datasets limits the

scope of the findings, as these datasets may only partially capture the diversity of legal texts or real-world software scenarios. The evaluation is based on standard performance metrics for the Question-answering task without considering additional criteria like robustness or practical usability.

Also, although the created dataset is limited to DoD cybersecurity policies, a small subset of the global cybersecurity policy landscape, it has the potential to serve as a representative English language dataset for future cybersecurity policy datasets and possibly become a benchmark similar to Shankar et al. (2023). In its current version, the dataset supports only English language evaluations and provides Multiclass-Classification and Text-Co-Occurrence tasks. However, it can be expanded with additional tasks in the future. Despite being built on a human knowledge base, this dataset cannot yet be compared against human expert performance on the provided tasks.

Regarding internal challenges, potential issues have arisen during the data extraction process. Data extraction is inherently complex and could have led to misclassified examples. The risk of misclassification using the DoD chart as a knowledge base is mitigated by ensuring that categories are pre-defined and not subject to annotator interpretation. The chart was adapted to a format suitable for NLP algorithms. Additionally, the authors consulted with a legal expert to verify classifications in cases of ambiguity.

The models evaluated on the dataset's tasks represent only a small fraction of the current transformer language models. The primary Hugging Face pipeline uses the [CLS] token encoding for classification as a baseline approach, which inherently carries certain limitations. Therefore, the results obtained for each model are subject to the limitations of both the models and the approach taken. Any conclusions drawn from this chapter should not be generalized to other NLP domains or beyond the scope of this dataset.

3.5 Conclusions

This chapter found that domain-specific models like LEGAL-BERT variants on the SQuAD V2.0 and PolicyQA datasets did not outperform general pre-trained models such as BERT and ALBERT. ALBERT emerged as the best-performing model, suggesting its suitability as a foundational contextual embedding encoder for future complex model designs. The findings indicate that training ALBERT from scratch on software development legal domain data could enhance its effectiveness as a base encoder, making it a promising avenue for future research.

In parallel, this chapter emphasizes the critical role of cybersecurity policies in the modern digital landscape and the need for dedicated datasets in this area, given the progress in Legal NLP and the exploration of various legal subdomains. While much of the research community has focused on privacy policies, this chapter broadens the scope by introducing the CSIAC-DoDIN (V1.0) dataset, which compiles cybersecurity-related policies and issuances developed by the DoD Deputy CIO for Cybersecurity. The baseline performances of classic and domain-specific transformer models, including BERT, RoBERTa, Legal-BERT, and PrivBERT, revealed good transferability from legal domain-specific models to cybersecurity policies in Multiclass-Classification tasks. However, this was not the case in Text-Co-Occurrence tasks.

These findings are significant for automating legal compliance in software systems. By identifying general domain LLMs as currently leading models for Legal NLP tasks and introducing the CSIAC-DoDIN dataset, this thesis not only enhances understanding of transformer models in specialized legal domains but also provides critical resources for the future development of models tailored to software development and cybersecurity policies.

3.6 Credit

In this section, we will provide each author's contributions to the work presented in this chapter. For this, we will use the system CRediT(`https://www.elsevier.`

com/researcher/author/policies-and-guidelines/credit-author-statement) from Springer to make it easier.

For the first paper : Caballero, Ernesto Quevedo, Mushfika Sharmin Rahman, Tomas Cerny, Pablo Rivas, and Gissella Bejarano. "Study of Question Answering on Legal Software Document using BERT based models." In LatinX in Natural Language Processing Research Workshop. 2022

Ernesto Quevedo Caballero: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data Curation, Writing-Original Draft, Writing-Review Editing, Visualization, Project administration.

Mushfika Sharmin Rahman: Validation, Investigation, Software, Formal analysis, Writing-review and editing, Visualization.

Tomas Cerny: Writing-review and editing, Visualization, Supervision.

Pablo Rivas: Writing-review and editing, Visualization, Supervision.

Gissella Bejarano: Writing-review and editing, Supervision.

For the second paper: Quevedo, Ernesto, Ana Paula Arguelles, Alejandro Rodriguez, Jorge Yero, Dan Pienta, Tomas Cerny, and Pablo Rivas. "Creation and Analysis of a Natural Language Understanding Dataset for DoD Cybersecurity Policies (CSIAC-DoDIN V1. 0)." In 2023 International Conference on Computational Science and Computational Intelligence (CSCI), pp. 91-98. IEEE, 2023.

Ernesto Quevedo Caballero: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data Curation, Writing-Original Draft, Writing-Review Editing, Visualization, Project administration.

Ana Paula Arguelles: Validation, Investigation, Resources, Data Curation, Writing-review and editing, Visualization.

Alejandro Rodriguez: Validation, Investigation, Resources, Data Curation, Writing-review and editing, Visualization.

Jorge Yero: Validation, Investigation, Resources, Data Curation, Writing-review and editing, Visualization.

Dan Pienta: Conceptualization, Validation, Investigation, Visualization, Writing-review and editing, Supervision.

Tomas Cerny: Visualization, Writing-review and editing, Supervision.

Pablo Rivas: Visualization, Writing-review and editing, Supervision.

CHAPTER FOUR

Survey in the Use of LLMs to Understand Software Systems

The advent of Large Language Models (LLMs) has opened new horizons in software engineering, offering unprecedented capabilities in understanding and managing complex software systems. Traditionally, code generation, code summarization, software design, and vulnerability detection tasks required significant human expertise and manual effort. However, with the development of LLMs, there is a growing potential for automating these tasks, leading to more efficient and intelligent software development processes. LLMs are now being explored for their ability to generate code or detect flaws and their potential to comprehend the intricate workings of software systems holistically.

Understanding software systems is a multifaceted challenge that involves grasping the underlying business processes, architectural structures, data flows, and the interaction between various components. With their vast knowledge bases and ability to process and generate human-like text, LLMs are increasingly being used to analyze and reason about these complex systems. Their application in software engineering extends beyond mere automation; LLMs have the potential to revolutionize how developers and engineers approach system design, debugging, maintenance, and compliance with legal and regulatory frameworks.

Additionally, the significance of LLMs in understanding software systems extends beyond technical efficiency and plays a crucial role in the broader context of ensuring legal compliance in software development. As software systems increasingly operate within complex legal and regulatory frameworks, designing and maintaining systems that adhere to these rules becomes imperative.

LLMs can facilitate this by automatically analyzing software systems for compliance with legal regulations, identifying potential violations, and suggesting corrective measures. By understanding the intricate interactions within a software system, LLMs can map these interactions to relevant legal standards, ensuring that the system functions correctly and operates within the bounds of the law. This capability is vital for industries where regulatory compliance is mandatory, such as finance, healthcare, and data protection, making LLMs a critical tool in mitigating legal risks and enhancing the overall integrity of software systems.

Therefore, this chapter will provide a literature review on the current state of the art of using LLMs to understand software systems.

4.1 Taxonomy of Software Engineering Area where LLMs have been applied

This section provides a summary of the research conducted on software understanding across various aspects of the software development process. Figure 4.1 shows a summary of the findings of the literature review.

4.1.1 Software Requirements and Documentation

Requirement engineering is a foundational discipline in software engineering that underpins the success of any software development project. Its significance lies in bridging the gap between stakeholders' abstract needs and a software system's tangible features. This process begins with a thorough understanding of stakeholder needs, encompassing various inputs, including business objectives, user expectations, and regulatory requirements. Effective requirement engineering ensures that these diverse inputs are accurately translated into clear, actionable, and testable requirements that guide the development process Jin et al. (2024).

A key aspect of requirement engineering is the thorough documentation of these requirements. This documentation serves as a blueprint for the development team, providing a detailed specification of what the software should do and how it should

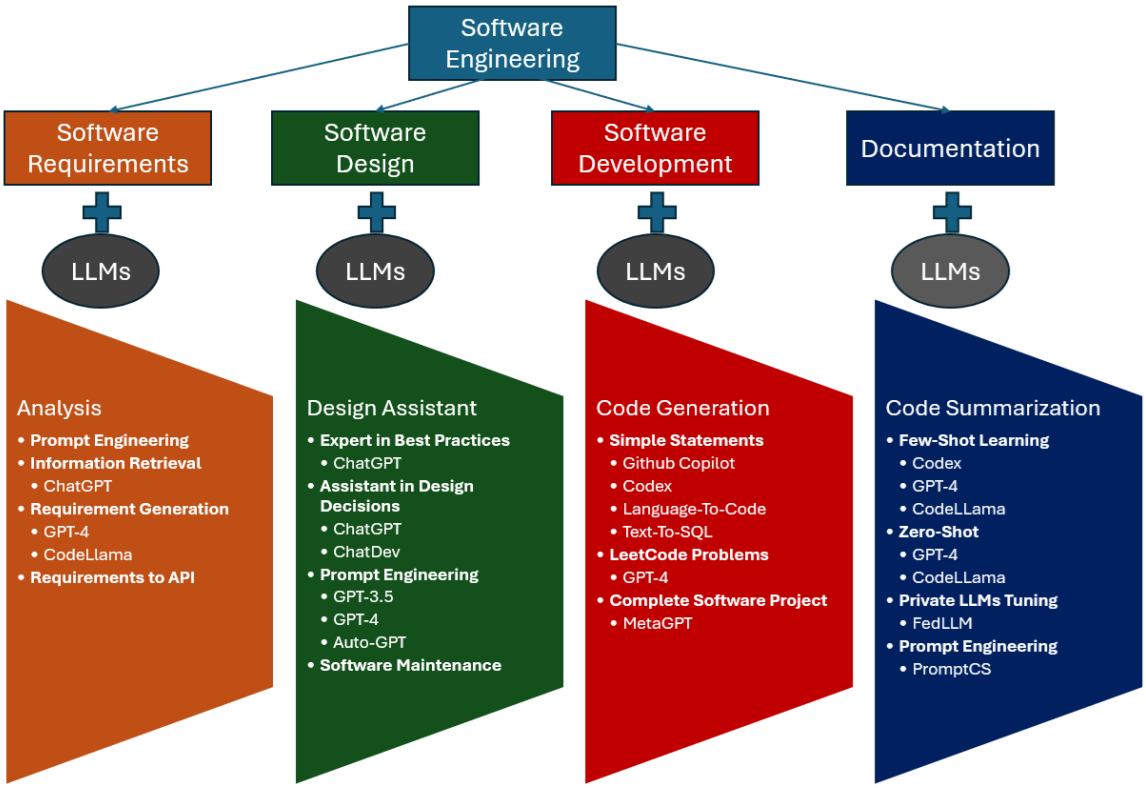

Figure 4.1. Summary of the review on LLMs applied to Software Engineering

perform. The accuracy and clarity of this documentation are critical; any ambiguity or error can lead to misunderstandings that might result in costly rework or project delays. Therefore, the process of requirements specification must be meticulously managed, with a focus on creating documents that are both comprehensive and accessible to all stakeholders involved.

Moreover, requirement engineering is not a one-time activity but an ongoing process that extends throughout the software development lifecycle. As projects evolve, new needs may emerge, or existing requirements may change due to shifting business goals, technological advancements, or regulatory updates. Requirement management, therefore, involves continuously tracking these changes and assessing their impact on the project. This proactive management ensures that the software remains aligned with stakeholder expectations and that any changes are incorporated without compromising the project's integrity or quality.

Beyond documenting what needs to be built, requirement engineering is critical in ensuring the final product meets its intended purpose. This is achieved through rigorous requirement verification, validating that the software correctly implements the requirements. This step is crucial to ensure that the development process is not just focused on building a system that works but one that works in the way stakeholders intended. This verification process can involve various techniques to ensure the software delivers its promised value, including reviews, prototyping, and testing.

Requirement engineering becomes even more critical in complex projects, especially those with significant legal or regulatory considerations. The ability to accurately capture, document, and manage requirements can be the difference between a project that smoothly navigates regulatory compliance and one that faces costly setbacks or legal challenges. Therefore, robust requirement engineering practices are essential for meeting user needs and ensuring that software systems adhere to legal and regulatory standards, safeguarding the project against potential risks and liabilities.

Here, some of the most essential works in this area are listed:

Luo et al. (2022) introduce a new approach to requirement classification using prompt learning with BERT-based pre-trained language models (PRCBERT). This method employs flexible, prompt templates to enhance classification accuracy through experiments conducted on two existing small datasets (PROMISE and NFR-Review) Hey et al. (2020); Wang et al. (2018) and a newly collected large-scale dataset (NFR-SO). The authors demonstrate that PRCBERT performs slightly better than both NoRBERT Hey et al. (2020) and MLM-BERT (BERT with a standard prompt template). On the de-labeled NFR-Review and NFR-SO datasets, the Trans_PRCBERT variant (fine-tuned on PROMISE) achieves satisfactory zero-shot performance, with F1-scores of 53.27% and 72.96% when using a self-learning strategy.

The research of Zhang et al. (2023) conducts an empirical evaluation of ChatGPT's performance in requirements information retrieval (IR) tasks to gain insights that could inform the design and development of more effective requirements retrieval methods or tools using generative LLMs. The authors created an evaluation framework considering four combinations of two widely used IR tasks and two common types of artifacts. The results from the zero-shot evaluation indicate that while ChatGPT shows a solid ability to retrieve generally relevant requirements information (high recall), it struggles to retrieve more specific requirements details (low precision). This preliminary evaluation of ChatGPT in the context of requirements IR under a zero-shot setting offers initial evidence for the potential to design or develop more effective requirements IR methods or tools based on generative LLMs.

The study of Krishna et al. (2024) investigates the capability of LLMs to produce accurate, coherent, and well-structured drafts of Software Requirement Specification (SRS) documents to expedite the software development process. They evaluate the performance of GPT-4 Achiam et al. (2023) and CodeLlama Roziere et al. (2023) in drafting an SRS for a university club management system, comparing the outputs against human-generated benchmarks across eight distinct criteria. The results suggest that LLMs can achieve a quality of output comparable to that of an entry-level software engineer, producing complete and consistent SRS drafts. Additionally, the study assesses the ability of LLMs to identify and correct issues within an existing requirements document. The experiments reveal that GPT-4 effectively detects problems and provides constructive feedback for improvement, while CodeLlama's performance in validation tasks was less promising. The authors repeat the SRS generation task across four different use cases to evaluate the time savings achieved using LLMs. The findings demonstrate that LLMs can significantly reduce entry-level software engineers' development time. Consequently, the study concludes that software

engineers can effectively utilize LLMs to enhance productivity by saving time and effort in generating, validating, and refining software requirements.

White et al. (2024) present prompt design techniques for software engineering, specifically patterns, to address common challenges when using LLMs like ChatGPT for automating typical software engineering tasks. These tasks include ensuring code is decoupled from third-party libraries and generating API specifications from requirement lists. Their work contributes to the field in two main ways: first, by providing a catalog of prompt patterns that classify these patterns based on the types of problems they solve, and second, by exploring how specific prompt patterns can be applied to improve various aspects of software engineering, such as requirements elicitation, rapid prototyping, code quality, deployment, and testing. The authors emphasize that while much focus has been placed on the mistakes made by LLMs in software engineering, applying prompt patterns can help mitigate these errors and reduce the frequency of mistakes. The authors highlight several capabilities of LLMs, such as simulating systems based on requirements and generating API specifications, which are challenging to automate using traditional technologies. However, the authors caution that significant human expertise and involvement are still required to effectively leverage LLMs, mainly due to issues like "hallucinations" where LLMs confidently generate incorrect output. While prompt patterns offer some mitigation, further research is needed to address prompt engineering aspects, such as quality assurance and versioning, to ensure that LLM-generated outputs are accurate and practical.

The research of Ronanki et al. (2022) investigates the use of ChatGPT for evaluating user story quality and compares its performance with an existing benchmark. The findings indicate that ChatGPT's evaluations are consistent with human evaluations, and the authors propose a "best of three" strategy to improve the stability of its outputs. They also discuss the concept of trustworthiness in AI and its implications for non-experts using ChatGPT's unprocessed outputs. In

addition to evaluating ChatGPT's performance, the paper addresses the broader issue of trustworthiness in AI, particularly concerning non-expert users relying on unprocessed outputs from ChatGPT. The study shows the importance of establishing high-level trustworthiness standards to ensure that ChatGPT and similar AI systems are integrated responsibly into Agile development processes. This includes considering the potential impact of AI tools on the development process and ensuring that their limitations are understood and managed effectively.

The research of Zhang et al. (2024) presents a novel AI-powered software development framework called AISD (AI-aided Software Development), which automates the construction of small programs from high-level user requirements. This framework differs from existing approaches by maintaining user involvement throughout the process, focusing on capturing feedback at multiple stages use cases, system designs, and prototype implementations. AISD allows software engineers to focus on more complex and interesting tasks, such as requirement engineering and system testing, rather than low-level coding. The system takes vague or high-level user inputs, generates detailed use cases, produces prototype system designs, and eventually implements the system. The evaluation results indicate that AISD improves the success rate of task completion while reducing resource consumption, specifically in terms of tokens used. This suggests the potential for a future where much software development is automated, with engineers focusing primarily on requirement engineering and system validation. The research demonstrates the effectiveness of human-AI collaboration in automating complex software tasks and offers a new perspective on the future of software engineering.

4.1.1.1 Overall Conclusions. Integrating (LLM-based agents in requirements engineering represents a transformative shift in how software requirements are generated, refined, and validated. LLMs' capabilities extend beyond merely producing textual content; when embedded within multi-agent systems, they enable a more

holistic approach to automating and optimizing the requirements engineering process. This shift is particularly impactful in environments where precision, consistency, and adherence to regulatory standards are essential, such as in industries with extensive legal compliance requirements.

In real-world applications, especially in high-level software design, simple LLMs may need to catch up due to the complexity and specificity required in generating and refining requirements. LLM-based agents, however, address these limitations through their collaborative nature. These agents are designed to work together, leveraging each agent's strengths in the system. Utilizing a shared database allows them to analyze and refine requirements from multiple professional perspectives, including legal, technical, and user-centered viewpoints.

Integrating LLM-based agents in requirements engineering offers a unique opportunity to embed legal compliance directly into the software development lifecycle. Traditionally, legal compliance has been a separate, often manual, process conducted after the initial requirements are defined. However, with LLM-based agents, legal requirements can be automatically incorporated into the initial requirements generation phase. These agents can be equipped to understand and apply legal texts, ensuring that all generated requirements are technically sound and legally compliant from the outset.

4.1.2 Code Generation and Software Development

Recent advancements in LLMs have significantly impacted code generation and software development, two core areas within software engineering. The primary goal of utilizing LLMs in these fields is to improve development efficiency and code quality through automation, thereby meeting the needs of both developers and end users. Over the past few years, applying LLMs in code generation has revolutionized how developers work, leading to a shift in automated development processes Jiang et al. (2024); Jin et al. (2024).

Compared to requirements engineering, where LLMs are still emerging, research on applying LLMs and LLM-based agents in code generation and software development is far more extensive and in-depth. LLMs, leveraging natural language processing and generation technologies, have demonstrated the ability to understand and generate complex code snippets, thus automating various stages of software development. This includes tasks ranging from code writing and debugging to software optimization, which are critical for improving development efficiency Jiang et al. (2024); Jin et al. (2024).

Decoder-based LLMs, such as GPT-4, have shown considerable potential in code generation by offering accurate code suggestions and automated debugging features. These capabilities speed up the development process and enhance the software's quality. Tools like GitHub Copilot, which integrate LLMs, have proven their effectiveness in boosting programming efficiency and ensuring higher code quality Jiang et al. (2024); Jin et al. (2024).

The growing attention toward LLM-based agents in software development marks a significant development in this field. These intelligent agents can perform complex code-generation tasks while engaging in autonomous learning and continuous refinement. This allows them to offer flexible assistance in dynamic development environments, adapting to the evolving needs of the software development process Jiang et al. (2024); Jin et al. (2024).

Moreover, integrating LLM-based agents in code generation brings about several benefits, including handling tasks requiring a deep understanding of code and natural language. For instance, these agents can be designed to comply with legal and regulatory standards, ensuring that the generated code adheres to specific legal requirements. This is particularly important in industries where software must meet extensive compliance standards, such as finance, healthcare, or public administration.

Applying LLMs and LLM-based agents in code generation and software development represents a significant advancement in software engineering. By automating complex tasks and continuously refining their outputs through autonomous learning, these technologies improve development efficiency and code quality and pave the way for more sophisticated, legally compliant, and user-friendly software solutions. As research continues to evolve, LLMs are likely to become even more integral to the software development process, further transforming the landscape of this critical industry.

Next, some of the most relevant work in this area are summarized.

Chen et al. (2021) introduce Codex, a GPT language model that has been fine-tuned on publicly available code from GitHub, and examine its capabilities in writing Python code. A distinct production version of Codex is the engine behind GitHub Copilot. In their evaluation using HumanEval Chen et al. (2021), a newly released dataset designed to measure functional correctness in synthesizing programs from docstrings, Codex successfully solves 28.8% of the problems. In comparison, GPT-3 solves 0%, and GPT-J solves 11.4% of the problems. The authors also discover that repeated sampling from the model is an unexpectedly effective approach for generating correct solutions to challenging prompts, achieving a 70.2% success rate with 100 samples per problem. However, a thorough analysis of Codex reveals certain limitations, such as difficulties with docstrings that describe long chains of operations and challenges in binding operations to variables. The authors also discuss the potential broader impacts of deploying advanced code generation technologies, considering safety, security, and economic implications.

The research of Ni et al. (2023) highlights that LLMs, particularly those pre-trained on code, have recently shown strong capabilities in generating programs from natural language inputs, even in few-shot or zero-shot settings. Despite these promising outcomes, more comprehensive evaluations must be conducted to assess

these models' language-to-code generation abilities fully. Existing research often focuses on specific tasks, model architectures, or learning paradigms, leading to a fragmented understanding of their overall performance. The authors introduce Language To Code Evaluation (L2CEval) to address this gap. This systematic evaluation framework assesses the language-to-code generation capabilities of LLMs across seven tasks, covering a wide range of domains, including semantic parsing, math reasoning, and Python programming. They analyze various factors influencing performance, such as model size, pretraining data, instruction tuning, and different prompting methods. Beyond evaluating model performance, they also assess the models' confidence calibration and conduct human evaluations of the generated code. This thorough analysis allows them to identify and explore typical failure modes across different tasks and models.

The work of Sun et al. (2023) explores the transformative potential of LLMs in the Text-to-SQL domain, where natural language is translated into Structured Query Language (SQL). Their work introduces the SQL-PaLM framework, a comprehensive approach to enhancing text-to-SQL tasks using LLMs, particularly in few-shot prompting and instruction fine-tuning settings. In the context of few-shot prompting, the authors examine the effectiveness of consistency decoding combined with execution-based error filtering. This method helps in refining the accuracy of generated SQL queries. For instruction fine-tuning, they delve into the key factors that affect the performance of fine-tuned LLMs, investigating how improvements can be achieved through expanded training data coverage, diverse and synthetic data augmentation, and the integration of query-specific database content. To further enhance accuracy, the authors propose a test time selection method that combines SQL outputs from multiple paradigms, using execution feedback as a guide. They also address practical challenges in navigating complex databases with numerous tables and columns by developing

efficient techniques for selecting relevant database elements, thereby improving text-to-SQL performance.

The study of Hu et al. (2024) recognizes that while LLMs have made notable advances in code generation tasks, they still struggle with programming problems that involve complex data structures and algorithms. The authors propose an in-context learning approach that guides LLMs in debugging by utilizing a "print debugging" method to address this limitation. This method involves inserting print statements to trace program execution and analyze logs to identify and fix bugs. To test their approach, the authors collect a dataset of Leetcode problems and evaluate their method's effectiveness using the Leetcode online judging system. Experiments conducted with GPT-4 demonstrate the success of this approach, showing a performance improvement compared to rubber duck debugging. Specifically, the method outperforms rubber duck debugging by 1.5% on easy Leetcode problems and by 17.9% on medium level problems.

Peng et al. (2023) investigate the potential of generative AI tools to enhance human productivity, focusing on GitHub Copilot, an AI-powered pair programming tool. In a controlled experiment, software developers were recruited to implement an HTTP server in JavaScript to complete the task as quickly as possible. The treatment group, which had access to GitHub Copilot, completed the task 55.8% faster than the control group that did not use the AI tool. The authors also observed heterogeneous effects, indicating that AI pair programming tools like GitHub Copilot could be particularly beneficial in helping individuals transition into software development careers. These findings suggest that AI tools not only improve task efficiency but also have the potential to support the development of new skills in aspiring software developers.

The research of Dong et al. (2023) addresses the limitations of LLMs in handling complex code generation tasks despite their notable achievements in simpler

scenarios. In real-world software development, complexity is often managed through collaborative teamwork, simplifying the development process and improving software quality. Drawing inspiration, the authors introduce a self-collaboration framework for code generation using LLMs, with ChatGPT as a key example. In this framework, multiple LLM agents are assigned specific roles as distinct "experts," each focusing on a particular subtask within a larger, complex task. These agents collaborate and interact as a virtual team through predefined role instructions, working together to complete code generation tasks without human intervention. The framework integrates software development methodologies to organize and manage this virtual team efficiently. The elementary team comprises three LLM roles: analyst, coder, and tester, who are responsible for the analysis, coding, and testing stages of software development. The authors conducted extensive experiments using various code generation benchmarks to evaluate the effectiveness of this self-collaboration approach. The experimental results demonstrate significant improvements, with the self-collaboration framework achieving a 29.9% to 47.1% increase in Pass@1 performance compared to a single LLM agent. Furthermore, the study highlights the potential of this self-collaboration framework to enable LLMs to tackle complex repository-level tasks, which are typically beyond the capabilities of a single LLM agent.

The work of Hong et al. (2023) discusses the significant advancements in automated problem solving using societies of agents based on LLMs. While existing LLM-based multi-agent systems can handle simple dialogue tasks, they encounter difficulties when addressing more complex tasks due to logic inconsistencies and cascading hallucinations that arise from naively chaining LLMs together. To overcome these challenges, the authors introduce MetaGPT, an innovative meta-programming framework that integrates efficient human workflows into LLM-based multi-agent collaborations. MetaGPT encodes Standardized Operating Procedures (SOPs) into prompt sequences, creating streamlined workflows that allow agents with human-like

domain expertise to verify intermediate results and minimize errors. The framework employs an assembly line approach, assigning diverse roles to various agents, which effectively breaks down complex tasks into manageable subtasks, enabling multiple agents to collaborate efficiently. In collaborative software engineering benchmark experiments, MetaGPT demonstrated its ability to generate more coherent solutions compared to previous chat-based multi-agent systems.

4.1.2.1 Overall Conclusions. Traditional LLMs typically utilize a single model to perform tasks such as code generation or completion. This approach faces limitations related to context window size and the need for ongoing feedback, which can hinder their ability to manage complex tasks effectively.

In contrast, LLM-based agents use a collaborative framework where multiple specialized agents work together to address different aspects of a task. For instance, in code generation, one agent might handle initial code generation, another could design test cases, and a third might execute tests and provide feedback. This division of labor facilitates iterative optimization and enhances efficiency. Similarly, in automatic legal compliance analysis, LLM-based agents could split tasks like interpreting legal texts, generating compliance reports, and verifying adherence to regulations, leading to more comprehensive and accurate compliance assessments.

LLM-based agents offer a more scalable solution by simulating real-world collaborative workflows. This system is adept at handling intricate software development challenges and can also manage complex legal compliance tasks. LLM-based agents can effectively handle broader and more complex compliance scenarios by dividing tasks among multiple agents and integrating iterative testing and refinement. This approach allows for better management of diverse legal requirements and ensures more robust compliance analysis.

4.1.3 Software Design and Evaluation

Software design is a critical early phase in the software development lifecycle, and its quality profoundly influences subsequent development stages. Modern methodologies advocate for seamless integration of design and development to ensure that design decisions translate into high-quality code. LLMs contribute significantly to this integration by enhancing design and development processes through specialized frameworks and architectures Jin et al. (2024).

In this context, LLMs automate and optimize various design tasks. Frameworks incorporating LLMs for software design often involve iterative refinement processes similar to code generation and development. For example, LLMs can assist in creating initial design drafts, generating design documents, and iterating on these drafts based on continuous feedback. This iterative approach ensures design outputs align closely with development requirements and user expectations.

Requirement elicitation and specification are integral to software design and intersect with LLM applications. LLMs assist in gathering and specifying requirements by analyzing user inputs, generating requirement documents, and refining these documents based on feedback. This role is pivotal in ensuring design outputs accurately reflect user needs and project goals.

Integrating LLMs in software design and evaluation extends to automatic legal compliance analysis. In this domain, LLMs can assist in interpreting legal requirements, generating compliance reports, and verifying adherence to regulations. For instance, LLM-based systems can automate the review of compliance documentation, assess legal risks, and provide recommendations for improving compliance.

LLMs equipped with autonomous learning and decision-making capabilities can enhance the accuracy and efficiency of legal compliance assessments. These systems can adapt to evolving legal standards and improve compliance analysis by continuously learning from new legal data and feedback.

Next, some of the most influential works related to this topic are summarized.

Sridhara et al. (2023) investigate the application of ChatGPT to common software engineering tasks. The study examines fifteen tasks ChatGPT performed and compares its outputs with state-of-the-art solutions and human expert evaluations. They gathered ten random samples for each task and asked ChatGPT to perform specific actions, such as determining if two provided code snippets were duplicates (code clone detection). The outputs generated by ChatGPT were then compared against human expert evaluations and, where available, state-of-the-art tool outputs. The study calculated the accuracy of ChatGPT for each task. The results indicate that ChatGPT performs credibly on many tasks, often providing detailed responses and, in some cases, superior to those from human experts or current best practices. However, there are specific tasks where ChatGPT's current capabilities fall short, leading to incorrect answers and highlighting areas where its application could be improved.

The research of Wan et al. (2024) initially explores optimization techniques to accelerate LLM performance, focusing on methods such as quantization, pruning, and operation level adjustments. A distinctive approach discussed is optimizing LLM inference through innovative software and hardware co-design strategies. Following these optimization efforts, the paper investigates the performance of LLMs in the context of circuit design and verification, with a particular focus on functional verification. Utilizing automated prompt engineering, the paper leverages the capabilities of the established LLM, GPT-4, to generate High-Level Synthesis (HLS) designs with predefined errors. The study uses a comprehensive dataset of over 1000 function-level HLS designs, each with up to 45 error combinations injected into the source code. This dataset, named Chrysalis, extends beyond existing HLS error models, providing a valuable resource for enhancing LLMs' ability to debug code effectively.

Jalil et al. (2023) investigates ChatGPT's performance in responding to common questions within a widely used software testing curriculum. The findings reveal that ChatGPT successfully addresses 77.5% of the questions examined, providing correct or partially correct answers in 55.6% of cases and correct or partially correct explanations in 53.0% of cases. Furthermore, using the tool in a shared question context slightly improves the accuracy of answers and explanations. Based on these results, the paper explores the potential benefits and drawbacks of employing ChatGPT in educational settings, considering its impact on students and instructors.

The research of Suri et al. (2023) highlights the benefits of using context-rich prompts for these autonomous agents. Employing various prompting strategies with and without contextual information demonstrates that prompts containing context significantly enhance the understanding of user requirements. This approach minimizes irrelevant details that can impede task comprehension and degrade model performance, especially when interacting with complex frameworks such as Spring Boot, Django, and Flask. The investigation uses Auto-GPT (v0.3.0) Yang et al. (2023), an open-source application that utilizes GPT-3.5 and GPT-4. Auto-GPT effectively integrates the "thoughts" of LLMs to achieve assigned goals or tasks independently. This exploration highlights the advantages of context-rich prompts in improving the performance and efficiency of autonomous agents in software development.

The work of Qian et al. (2024) introduces ChatDev, a framework that integrates chat-powered LLMs to streamline software development. ChatDev utilizes specialized agents guided by LLMs to manage both the content and the manner of communication through a chat chain and communicative dehallucination techniques. These agents contribute to the design, coding, and testing phases by engaging in multi-turn dialogues and providing unified language-based solutions. The research highlights the benefits of using natural language for system design and programming language for debugging within this framework. By facilitating multi-agent collaboration through linguistic

communication, ChatDev demonstrates how language can serve as a cohesive bridge for autonomous task-solving among LLM agents, improving the overall efficiency and coherence of the development process.

Weber (2024) proposes a taxonomy for LLM-integrated applications, providing a framework for analyzing and describing these systems. The research showcases various methods for integrating LLMs into applications and outlines different implementation options. The study identifies relevant dimensions and evaluates the proposed taxonomy against additional cases by analyzing a sample of recent LLM-integrated applications. It finds that these applications often consist of multiple LLM integrations, referred to as "LLM components." To understand an application's architecture, each LLM component is examined individually. The study identifies thirteen dimensions for characterizing these components, such as the specific LLM skills and output format. The taxonomy effectively describes LLM-integrated applications, offering a structured way to represent and visualize the integration of LLMs. This framework is intended to advance theory in the emerging field of LLM-integrated application engineering and support the development of such systems. Despite ongoing challenges, integrating LLMs holds the potential to transform software system development through innovative applications.

The study of Vallecillos Ruiz (2024) investigates using LLM-powered agents to enhance software maintenance. These agents leverage LLMs' iterative learning and adaptability to address common challenges in code generation, mainly focusing on the "last-mile problems" where errors occur during the final stages of code production. The research proposes a collaborative framework wherein multiple agents, each equipped with LLMs, interact to rectify and learn from each other's mistakes. This iterative feedback mechanism aims to refine the LLMs and improve their alignment with automated software enhancement tasks. The project seeks to significantly advance automatic software improvement by developing innovative tools and frameworks within

this collaborative environment. The ultimate goal is to enhance the efficiency and reliability of software development processes through these new methodologies.

4.1.3.1 Overall Conclusions. The integration of LLMs into software design and evaluation has the potential to significantly enhance the efficiency and quality of these processes. Traditionally, LLM applications in these areas have concentrated on automating specific tasks such as code generation and log summarization, focusing primarily on evaluating the capabilities of these models rather than their implementation during the design phases. This approach reflects a narrower scope, often overlooking the broader impact that LLMs can have when applied to higher-level design tasks and legal compliance. For instance, LLM-based agents can automate compliance checks by analyzing legal requirements and comparing them with software design specifications.

Applying LLM-based agents in software design extends beyond basic automation to encompass sophisticated design tasks. These agents can dynamically adapt to various application scenarios, significantly improving the flexibility and accuracy of the design process. Frameworks like ChatDev exemplify this shift by utilizing role distribution to separate the design phase from other development stages. This approach enhances the efficiency of later development phases by integrating collaborative and role-specific functionalities Jin et al. (2024).

Despite these advancements, LLM agents need help in certain areas. Regarding text generation and vulnerability detection, LLMs have demonstrated superior performance, but tasks such as software maintenance and root cause analysis demand more intricate architectures. These tasks benefit from advanced techniques such as multi-turn dialogues, knowledge graphs, and Retrieval Augmented Generation (RAG), which can further refine the design and evaluation processes.

4.1.4 Code Summarization

Code summarization, often called code commenting, generates natural language descriptions that explain the functionality and purpose of various components within a computer program. By translating the logic of source code into human-readable text, code summarization plays a critical role in enhancing program comprehension. This is particularly important in the software maintenance phase, widely recognized as one of the most resource-intensive and time-consuming stages in the software development life cycle. High-quality summaries can significantly aid developers by reducing the time required to understand complex codebases, improving the efficiency of code searches, and supporting overall system maintainability Zhang et al. (2022); Steidl et al. (2013); Xia et al. (2017).

However, the fast-paced evolution of software often leads to mismatched, outdated, or missing code comments, which can hinder the maintenance process. As a result, there is a growing need for reliable and automated methods to generate accurate code summaries. This section will explore research efforts in code summarization, focusing on the approaches and techniques to address these challenges.

Ahmed and Devanbu (2022) explore the potential of LLMs, specifically GPT-based models like Codex, for code summarization tasks. The authors highlight the effectiveness of few-shot learning, where the model can learn from only a few examples. This ability is particularly valuable in software engineering, as code patterns and terminologies are often project-specific. Still, relevant project data is often limited, especially in the early stages of development. The authors investigate how few-shot training with Codex can outperform traditional models trained on thousands of examples, demonstrating that using just ten examples from the same project can significantly improve code summarization. Their findings show that few-shot learning performs better than using data from different projects when applied to project-specific data and that the improvements are statistically significant. The authors suggest

that this approach is promising for software engineering tasks, as it allows for efficient training even with minimal data early in the project lifecycle. They argue that project-specific few-shot training could extend beyond code summarization to benefit other tasks in software engineering as well.

The research of Sun et al. (2024) extensively studies using LLMs for code summarization, aiming to cover various aspects of the LLM-based workflow. The authors start by evaluating automated methods for assessing the quality of summaries generated by LLMs, finding that the GPT-4 evaluation method aligns most closely with human assessments. The paper also investigates five prompting techniques: zero-shot, few-shot, chain-of-thought, critique, and expert to adapt LLMs to code summarization tasks. Interestingly, the results show that simple zero-shot prompting can perform better than or better than more advanced techniques. The study also examines how different LLM model settings, particularly top-p and temperature parameters, influence the quality of code summaries. The impact of these parameters is found to vary depending on the base LLM and the programming language in question, but the overall effect is similar across different languages. The authors also assess how LLMs handle code summarization across different programming languages, revealing that the models need help summarizing code written in logic programming languages compared to procedural and object-oriented languages. Finally, the study finds that CodeLlama-Instruct, with 7B parameters, outperforms the more advanced GPT-4 model in generating summaries that describe code implementation details and assert code properties. The authors hope that their findings will serve as a valuable resource for future research, offering insights into the performance of LLMs in code summarization and helping to drive advancements in this area.

The work of Kumar and Chimalakonda (2024) focuses on addressing the limitations of applying LLMs to Software Engineering (SE) tasks such as code clone detection, code summarization, and program comprehension. While LLMs have shown

promising results, their performance often requires fine-tuning with task-specific datasets. However, the proprietary nature of SE data, particularly in closed-source environments, presents a significant challenge, as most LLMs are trained exclusively on open-source data. To tackle this issue, the authors propose a Federated Learning (FL) approach combined with LLMs, specifically targeting code summarization. The study introduces a Federated Large Language Model (FedLLM), which allows different clients to collaboratively train a model by exchanging only model weights without sharing the underlying private code data. The authors fine-tuned Llama-2 using Parameter Efficient Fine-Tuning (PEFT) techniques such as Low-Rank Adaptation (LoRA) Hu et al. (2021); Fan et al. (2023) and conducted experiments using a high-performance GPU setup. Their results show that the federated model performs comparably to a centrally trained one, even with minimal training epochs and only a small fraction of the updated parameters.

Sun et al. (2023) address the challenge of adapting LLMs for code summarization, which involves generating natural language summaries for code snippets to help developers better understand and maintain source code. While LLMs have shown success in various fields, applying them to code summarization has primarily relied on two strategies: instruction prompting and task-oriented fine-tuning. Instruction prompting, which involves crafting prompts for zero-shot or few-shot learning, demands professional domain expertise, while task-oriented fine-tuning incurs high training costs. To overcome these limitations, the authors propose a novel framework called PromptCS, which introduces a prompt learning approach tailored for code summarization. Rather than relying on manually designed prompts, PromptCS trains a prompt agent that generates continuous prompts, allowing LLMs to better comprehend and summarize code. By freezing the parameters of the LLM during prompt agent training, PromptCS reduces the computational resources typically required for fine-tuning. The authors evaluate PromptCS on the CodeSearchNet dataset Husain et al. (2019), which spans

multiple programming languages, and find that it significantly outperforms traditional instruction prompting methods across several evaluation metrics. In certain LLMs, such as CodeGen-Multi-2B and StarCoderBase-1B and -3B, PromptCS surpasses task-oriented fine-tuning in performance. Additionally, PromptCS demonstrates superior training efficiency, especially with larger LLMs. Human evaluations further confirm that PromptCS generates higher-quality summaries compared to existing approaches.

4.1.5 Overall Conclusions

The integration of LLMs into code summarization has the potential to revolutionize how software developers understand and maintain codebases. Traditionally, code summarization efforts have focused on generating basic summaries or comments, primarily addressing developers' immediate need to comprehend code snippets quickly. This traditional approach often emphasized simple automation without fully exploiting the advanced capabilities of LLMs to enhance the depth and context of code documentation.

Despite these advancements, applying LLMs to more complex code summarization presents challenges. While LLMs excel in generating coherent and contextually relevant summaries, tasks such as handling ambiguous code or summarizing intricate code interactions require more nuanced architectures. Advanced techniques like multi-turn dialogues and Retrieval Augmented Generation (RAG) offer promising solutions to refine these summaries by integrating additional context and iterative feedback, further enhancing the effectiveness of code documentation.

An emerging and critical area of integrating LLMs into code summarization is their role in improving documentation for diverse codebases and programming languages. By leveraging LLM-based agents, developers can achieve more precise and contextually relevant summaries, addressing the immediate and long-term needs of code maintenance and understanding. This approach enhances code readability

and supports ongoing development and compliance with coding standards and best practices.

More importantly, code summarization is critical in automatic legal compliance as software becomes more intertwined with legal requirements—from data privacy laws to industry-specific standards. LLMs can elevate code summarization from a functional tool to a compliance mechanism by identifying code sections that may introduce legal vulnerabilities. LLM-based summaries can flag code related to data handling or encryption, ensuring adherence to regulations like GDPR. This is especially valuable in industries such as healthcare and finance, where non-compliance can result in severe penalties. By offering a layer of legal insight through summaries, LLMs enable developers to address potential compliance issues early in the development process, saving time and resources and reducing legal risk.

4.2 Discussion

Integrating LLMs into software systems understanding represents a profound shift in the software engineering landscape. This transformation is particularly evident in the growing trend of using LLMs like ChatGPT to evaluate and comprehend complex software systems. This approach promises to revolutionize how software is designed, developed, and maintained, including areas requiring extensive legal compliance.

Traditional software engineering has long relied on human expertise to understand, design, and maintain complex systems. The introduction of LLMs into this process marks a significant departure from this norm. These models, particularly when deployed as LLM-based agents, are not merely tools for generating code or summarizing logs; they represent a new paradigm for understanding software systems holistically. By leveraging vast amounts of data and sophisticated Natural Language Processing (NLP) capabilities, LLMs can analyze software requirements, design patterns, and even codebases in previously unattainable ways.

However, the use of LLMs in this context has challenges. While these models excel in generating text and automating specific tasks, they often need help with the nuances and complexities inherent in high-level software design. The collaborative nature of LLM-based agents offers a potential solution to these limitations. By distributing tasks among multiple specialized agents, each focusing on a particular aspect of the software development process, LLM-based systems can more effectively manage the intricacies of software design. This collaborative approach allows for a more accurate and efficient understanding of software systems, particularly in environments where precision and consistency are critical.

The current trend of using ChatGPT for software system evaluations extends this broader movement toward LLM integration. ChatGPT, with its conversational capabilities, offers a unique advantage in evaluating and understanding software systems. By engaging in dialogue with developers, ChatGPT can clarify requirements, suggest improvements, and even identify potential issues in the code. This interactive approach contrasts with more traditional, static methods of software evaluation, where tools and processes are often limited by their need for adaptability and context awareness.

However, the use of ChatGPT in this context also raises important considerations. While the model's ability to engage in multi-turn dialogues and provide context-aware suggestions is valuable, it is still limited by its training data and underlying architecture and hallucinations. For instance, ChatGPT may need help understanding the broader implications of design decisions or the specific legal requirements associated with certain software systems. This limitation shows the need for more sophisticated LLM-based agents that can integrate multiple sources of information, including legal texts and domain-specific knowledge, to provide more comprehensive evaluations.

The integration of LLMs, particularly in the context of automatic legal compliance, represents a significant advancement in software engineering. Traditionally, ensuring that software systems comply with legal regulations has been a labor-intensive process, often conducted manually after the software has been developed. This approach increases the risk of non-compliance and delays the software development lifecycle, as compliance checks are performed as a separate, often isolated, phase.

LLM-based agents offer a transformative approach to this challenge. By embedding legal compliance directly into the software development process, these agents can automatically analyze legal requirements and ensure that the software adheres to these standards. This capability is particularly valuable in industries where legal compliance is critical, such as healthcare, finance, and defense. In these environments, even minor deviations from legal standards can have significant consequences, making the integration of LLM-based compliance checks not just a convenience but an important step towards improvement.

The impact of this shift extends beyond mere efficiency gains. By automating legal compliance, LLM-based agents can help ensure that software systems are more robust, reliable, and secure. This is particularly important as software systems become increasingly complex and interconnected, with greater potential for legal and regulatory challenges. Moreover, by integrating continuous feedback and iterative refinement into the compliance process, LLM-based agents can more effectively adapt to changing legal standards and regulations than traditional methods.

4.3 Limitations

In this chapter on using LLMs to understand software systems, several limitations were encountered that may have influenced the overall breadth of the analysis. The chosen criteria for selecting research works could have unintentionally omitted important works, resulting in an incomplete field view. As a result, the scope of the review may be more restricted than intended. Furthermore, given the rapid

advancements in LLMs and software development, some insights from this review might quickly become outdated as new methods and approaches emerge.

The review also emphasized tasks closely related to the early stages of developing LLM-driven systems for software understanding, such as knowledge extraction, code generation, and system evaluation. However, it did not address other important tasks like debugging support, integration with legacy systems, and logging, which are crucial for fully operational LLM-based solutions.

From the reviewed works, it became clear that hallucinations in LLMs present a significant challenge to their full integration across all phases of software development. The review highlighted the limited research on how this phenomenon specifically impacts software development, which may be a limitation of the review but also reflects the fact that addressing hallucinations in LLMs is still in its early stages. Additionally, while the chapter covers applications of LLMs in various phases of software development, there is a lack of studies that attempt to apply LLMs across all phases and explore the combined results, such as in Hong et al. (2023).

The conclusions drawn here are specific to applying LLMs in software understanding and may not be easily extended to other domains where LLMs are applied. While some of the challenges highlighted may overlap with other areas, this review does not attempt to generalize beyond its intended focus. Additionally, the results presented across the various studies were not independently verified, meaning any limitations in those original studies may have carried over into the analysis provided in this review.

4.4 Conclusions

Integrating LLMs into software engineering marks a significant leap forward in understanding, designing, and maintaining complex software systems. This chapter has explored how LLMs are being applied across different domains within software engineering, from requirements engineering to code generation, software development,

and design. The insights gained highlight the transformative potential of LLMs in automating tasks traditionally reliant on human expertise and enhancing the overall quality, efficiency, and compliance of software systems.

With their vast knowledge bases and sophisticated natural language processing capabilities, LLMs are invaluable tools in the initial stages of software development. They facilitate a more accurate and comprehensive gathering of requirements, ensuring that all stakeholder needs, including legal and regulatory considerations, are addressed early in the development process. This proactive approach to embedding legal compliance into the requirements phase represents a fundamental shift in how software is designed, moving from post-hoc compliance checks to a more integrated and continuous process.

In code generation and software development, LLMs like Codex have demonstrated their ability to significantly accelerate development processes, generating code that meets functional requirements with increasing accuracy. However, challenges still need to be addressed, particularly in managing the complexity and specificity of tasks required in high-level software design. The collaborative nature of LLM-based agents offers a promising solution, enabling a more modular and iterative approach to development, where multiple agents work together to refine and optimize software outputs.

The exploration of LLMs in software design and evaluation further illustrates their potential to enhance the quality and consistency of software systems. By integrating LLMs into design processes, developers can ensure that design decisions are consistently translated into high-quality code, reducing the risk of errors and improving overall system reliability. The ability of LLMs to provide continuous feedback and iterative refinement during both design and evaluation phases is precious in complex, compliance-heavy industries, where precision and adherence to standards are critical.

Finally, the role of LLMs in automatic legal compliance analysis cannot be overstated. As software systems increasingly operate within extensive regulatory frameworks, the ability to automatically analyze and ensure compliance with legal standards is crucial. With their capacity to interpret legal texts and generate compliance reports, LLM-based agents offer a scalable solution to the growing complexity of legal requirements in software development. By integrating these capabilities into the software development lifecycle, organizations can mitigate legal risks and ensure that their systems operate within the bounds of the law from the outset.

Applying LLMs in software engineering is not just about automating routine tasks; it represents a paradigm shift in how software systems are understood, designed, and maintained. As LLMs evolve, their integration into software engineering practices will likely lead to even more sophisticated tools and frameworks, further enhancing software systems' quality, efficiency, and compliance. This chapter has provided a comprehensive overview of the current state of the art in using LLMs to understand software systems, setting the stage for future research and development in this rapidly advancing field.

CHAPTER FIVE

Assessing ChatGPT's Ability to Comprehend and Respond to Microservice Architecture Questions Using Source Code Insights

The work detailed in Chapter Five has been written mainly from the publication: Quevedo, Ernesto, Amr S. Abdelfattah, Alejandro Rodriguez, Jorge Yero, and Tomas Cerny "Evaluating ChatGPT's Proficiency in Understanding and Answering Microservice Architecture Queries Using Source Code Insights" SN Computer Science 5, no. 4 (2024): 422.

After exploring the capabilities of state-of-the-art LLMs in the classification and question-answering of software-related legal subdomains, such as privacy policies and cybersecurity, this thesis has provided an analysis of the results on how to use LLMs in the phase of understanding legal documents and identified the current gaps that still exist. However, as mentioned at the start of this thesis, the focus is on automatically interpreting legal documents and understanding how a software system operates. This understanding is crucial to map legal regulations to software workflows and determine whether the regulations are being complied with.

Therefore, this chapter will focus on the second problem: understanding a software system. Given that the current state of the art in automatic software understanding also relies on LLMs, the chapter aims to develop a tool based on an agent capable of answering questions about a software system, demonstrating its knowledge. This thesis will use ChatGPT as the LLM agent for the experiments. This choice is based on its strong performance and extensive publications highlighting its success across multiple domains, as discussed in Chapter 4.

The exploration of ChatGPT's effectiveness in answering questions about microservice systems directly aligns with the broader goal of auditing software compliance with legal regulations. In both contexts, the ability of Large Language

Models (LLMs) like ChatGPT to comprehend, process, and accurately respond to complex queries is crucial. For auditing software compliance, LLMs must navigate intricate regulatory requirements, interpret legal language, and correlate it with the technical details of the software systems. This requires a robust understanding of both the code and higher level system interactions.

This chapter will focus on three research questions to analyze ChatGPT's performance in understanding software systems. Given the current trend of using complex architectures like Microservices, this chapter will center the analysis on a microservice-based project. The chapter is organized as follows: first, present the research questions; next, the methodology followed; then, the experiments, results, and statistical analysis. Finally, the chapter concludes with a discussion of the results and limitations.

5.1 Research Questions

This chapter evaluates the service view and service interaction perspectives while exploring how different knowledge sources (such as source code and intermediate representation of the software system: Persistent Operation Component Call Graphs PO-CCG) influence the model's question-answering capabilities. Furthermore, it identifies potential challenges and limitations that may emerge.

RQ_1: Can ChatGPT leverage source code to answer questions about microservice service and interaction views, and in the process, what levels of effectiveness are achieved and what challenges are faced? This research question assesses ChatGPT's capability to answer structured questions about microservice system views. It seeks to evaluate the model's effectiveness and uncover any limitations in its responses.

RQ_2: Does ChatGPT show improved performance in answering questions about microservice systems when using source code instead of PO-CCG? Given that

source code can include extraneous information and often requires processing a large volume of tokens, the PO-CCG representation provides a more targeted and manageable flow of information. This research question aims to determine if ChatGPT can be more effective in answering questions by using source code than the PO-CCG representation.

RQ_3: Does integrating source code with PO-CCG enhance ChatGPT's effectiveness in answering questions about microservice systems? Following the assessment of source code and PO-CCG knowledge bases separately, this research question investigates whether combining these resources improves ChatGPT's ability to answer questions about microservice views.

5.2 Methodology

This section presents the methodology illustrated in Figure 7.1. This pipeline demonstrates the interconnection of various phases, guiding how the ChatGPT model supplies the necessary information and context to address questions about microservice systems effectively. Next, each of the phases involved is listed.

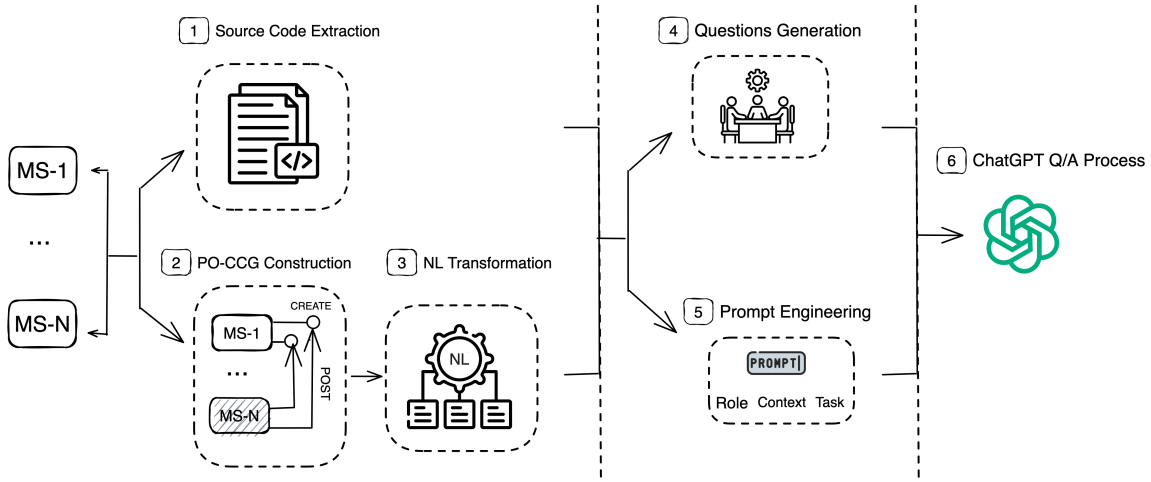

Figure 5.1. Methodology Phases.

(1) *Source Code Extraction:* Extract source code components from microservice projects.

(2) *PO-CCG Construction:* Extract PO-CCG representation from the microservice source code.

(3) *NL Transformation:* Transform the PO-CCG to natural language.

(4) *Question Generation:* Generate questions for study evaluation.

(5) *Prompt Engineering:* Perform prompt engineering to construct a complete prompt for each question.

(6) *ChatGPT Question Answering Process:* Provide the information and context required to perform the study.

5.2.1 Source Code Extraction

The methodology adheres to enterprise architecture standards that organize communications into distinct layers. Each project is categorized into Controller, Service, and Data Repository components. The method begins by inspecting the microservices' codebases to extract their source code files. After identifying these files, they are parsed to locate method declarations and their associated bodies. The content from the method declaration and body is then extracted and labeled as the method's source code. In addition to the source code, supplementary information is gathered as detailed in Table 5.1. For each method, it is identified its corresponding class and collects essential details about that class, which are also included in Table 5.1.

The PO-CCG Construction process unfolds in two phases. First, the Component Call Graphs (CCGs) are constructed and then enhanced to incorporate awareness of persistent operations. Listing 1 illustrates a source code example from a Java application.

Table 5.1. Information extracted during source code extraction phase

Extracted field	**Description**
methodName	The name of the method
methodSourceCode	The source code of the method including method declaration and method body
methodOffset	The location in the file where the source code is available, it contains the start and the end
methodModifiers	The modifiers of the method including accessibility and other available modifiers like static
symbolId	The complete identifier of the method including the class identifier
className	The name of the class in which the method is declared
classComponentType	The component type of the class, it could be Controller, Service, Repository or Other
classDeclarationType	The declaration of the class, it could be class, interface, enum or others
classFilePath	The file path where the class body is declared
classModifiers	The modifiers of the class, similar to the method modifiers
packageName	The name of the package in which the class is declared
microserviceName	The name of the microservice in which the class is declared

The CCGs are built using the methodology outlined in Abdelfattah et al. (2023). This involves static analysis of the application's source code to identify low-level constructs, including methods and classes across the entire application or specific modules. A method call graph is created by detecting method calls within each method's body, highlighting the relationships between methods.

Key to understanding the application are the entry point methods not called by others. These are identified using a depth-first search technique. The initial call graphs are then enhanced to form Component Call Graphs (CCGs), which include component types and their properties. As shown in Figure 5.2, component types are

denoted in brackets ([ ]), and properties are illustrated with rectangles connected to the respective components.

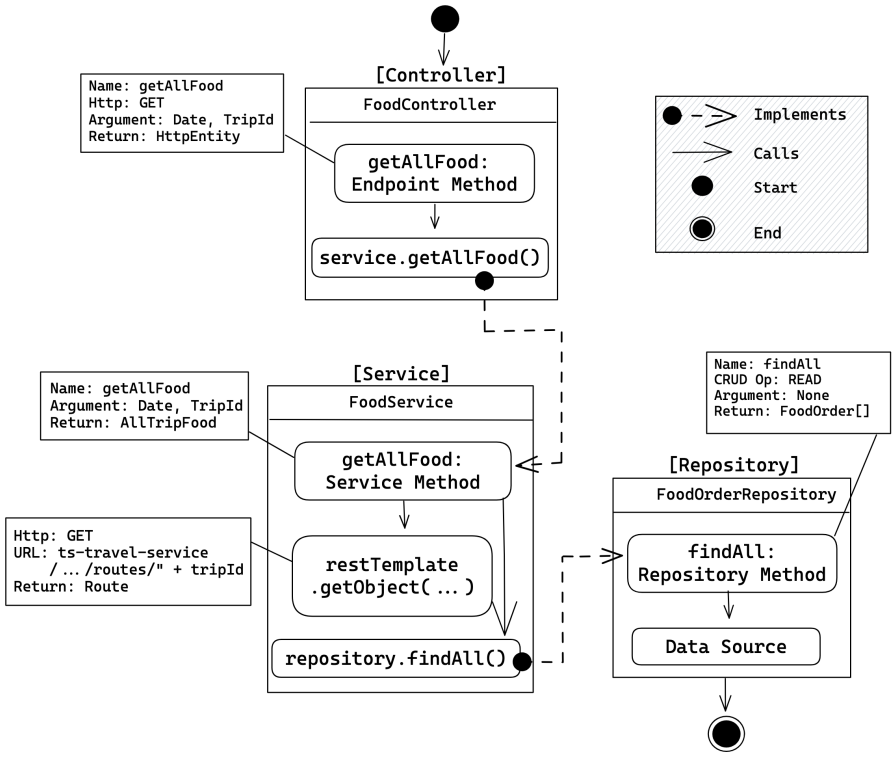

Figure 5.2. PO-CCG Example.

For example, Listing 5.1 depicts the getAllFood endpoint method in the FoodController component, which calls the getAllFood method in the FoodService component. The RecordService then makes two additional calls: to a third-party API using restTemplate and the findAll method in FoodOrderRepository. The resulting call graph, displayed in Figure 5.2, starts from the FoodController's endpoint interface and traces each method invocation to build the complete graph. Properties associated with the findAll method in FoodOrderRepository, such as "CRUD Op" with a value of "READ," indicate the method's involvement in persistent operations.

Listing 5.1. Source code example. Note *FoodOrder, AllTripFood, Route* are domain objects.

```
@Controller
```

```
public class FoodController {
    @Autowired
    private FoodService service;

    @GETMapping
    public HttpEntity getAllFood(String date, String tripId) {
        return service.getAllFood(date, tripId);
    }
}

@Service
public class FoodService {
    @Autowired
    private FoodOrderRepository repository ;

    public AllTripFood getAllFood(String date, String tripId) {
        Route route = restTemplate
                      .getObject("/ts-travel-service/service/routes/"
                       + tripId);
        FoodOrder[] foodOrders = repository.findAll();
        return new AllTripFood(route, foodOrders)
    }
}

@Repository
public interface FoodOrderRepository {
    FoodOrder[] findAll();
}
```

In the second phase, the CCGs are augmented to include persistence awareness. This involves extracting endpoints from the entire system and analyzing their source code to generate a call graph. Then, identify data entity accesses and CRUD operations

for each element in the graph. Extend this analysis to inter-service interactions to provide a comprehensive view, incorporating the persistent operations used by dependent endpoints in other microservices. This step ensures that the CCGs are fully aware of persistence operations.

5.2.2 NL Transformation

The PO-CCG components are translated into natural language using a heuristic-based approach. Specifically, the Control Flow Graph (CFG) is processed using a Visitor pattern to navigate the graph's structure. Currently, only flow paths are parsed, meaning the structure visited is essentially a list. However, the Visitor pattern is designed to be flexible and not tied to any specific traversal method.

Visiting a node may involve storing relevant information in the Visitor's context during traversal. For instance, when a node representing a controller is visited, its details are stored for potential use by subsequent nodes. Additionally, each visited node may generate a message that is appended to a sequence of messages maintained by the Visitor class. The final natural language description is created by concatenating all these generated messages.

Table 5.2 outlines which nodes in the graph produce messages and which merely contribute information to the context.

Conversely, parsing the Partial Operations is more straightforward. First, a header message is generated with the text: ``CRUD operations are performed over the following entities:". Following this, each entity's itemized list of CRUD operations is appended. For example, an entry might read ``Person: GET, PUT, POST", indicating that the Get, Put, and Post operations are performed on the Person entity.

Table 5.2. Nodes in the Control Flow Graph that produce a message in natural language.

Node	Message
Controller Method	`The '{controller_name}' has a '{http_method_type}' endpoint handled by the method '{controller_method}'. The endpoint returns an '{return_type}' object.`
Service Method	`The '{controller_method}' method in the '{controller_name}' controller calls the method '{service_method}' of the '{service_name}' service through its field '{service_field}'. This call returns a '{return_type}' object.`
Repository Method	`The '{service_method}' method in the '{service_name}' service calls the method '{repository_method}' of the '{repository_name}' respository through its field '{repository_field}'. This call returns a '{return_type}' object.`

Rest Calls	`The '{service_method}' method in the '{service_name}' communicates with other microservices in the following ways:`
	`- It communicates with {endpoint_message} endpoint. This request expects '{return_type}' object as response.`
	`- It communicates with...`

5.2.3 Questions Created

The methodology for preparing the questions revolves around assessing the model's accuracy in responding to queries about microservice systems from both service view and service interaction perspectives. These questions were crafted to be neutral and broadly applicable, ensuring they could be used across various scenarios without bias. To maintain objectivity, the questions are designed to elicit consistent answers, allowing for uniform evaluation of actual responses without subjective judgment. The aim extends beyond simply testing the model's knowledge but also seeks to gauge its ability to provide insightful and practical solutions for the complexities inherent in microservice systems.

The questions are divided into three categories: Endpoint Details, Remote Calls, and Dependency. The Endpoint Details category addresses the service view perspective, focusing on aspects such as the responsibilities of endpoints, their response mechanisms, internal operations like CRUD operations, and the configured HTTP methods. The Remote Calls category explores interactions between microservices, including the sequence of requests and responses between different microservices or endpoints. Lastly, the Dependency category examines the service dependency graph, investigating how microservices rely on each other and identifying the endpoints responsible for establishing these dependency relationships.

Additionally, each question requires data from several microservices to provide a complete answer. Questions related to Endpoint Details generally need information from just one microservice. In contrast, questions about Remote Calls and Dependencies often necessitate data from multiple microservices to address the interactions and dependencies involved fully.

The questions cover various aspects, including some focusing on specific functions and others addressing their responsibilities. They also use particular terminology and keywords with specific meanings; for instance, the term "communicate" in the context of microservices indicates a remote call relationship between them.

Questions encompass various microservice variations to ensure unbiased results, preventing skewing towards any specific use case. This approach ensures that the findings reflect different aspects of the system and are not biased by any particular scenario.

5.2.4 Prompt Engineering

To efficiently utilize ChatGPT for answering questions about microservices, particularly those concerning code and static analysis, prompts were designed with three key elements:

Role Prompt: As showed by the MetaGPT research by Hong et al. (2023) indicates the role the agent has to play affects the performance of the LLM. The decided role was of "a great developer with best practices in Microservices project". This foundational step ensures the model aligns its responses within the specified role, offering insights relevant to microservices. In this case, it was something like "Microservices Analyst" or "Code Reviewer" to test its capabilities with any questions, not more specific ones. In addition, this prompt also clarifies to ChatGPT that there will be a lot of upcoming messages before the actual questions come along, and to save tokens and get feedback; it is asked to answer only with "got it".

Context Prompts: ChatGPT with a specific context to see if it could accurately respond to a question based on that information. The whole context would conform to the class file path on the source code project, the methods' name, the method's source code, and also the PO-CCG in natural language form. Depending on what to test, some information will be excluded or not. In addition, not all the information was given in a single prompt, but a set of prompts, one per method. In addition, this prompt reminded ChatGPT that still more messages were coming before the actual questions and to refrain from answering anything rather than "got it".

Task Prompt: The task prompt asks the question after ChatGPT has all the context injected. However, depending on the question, perform a more targeted prompt engineering approach: (i) For questions that necessitated an understanding of specific definitions, ChatGPT was asked to define the concept; if the answer was correct, ChatGPT was prompted to rephrase the original question, including at the beginning of this definition in a way that was clear to understand and give that final answer as the Task Prompt. (ii) Additionally, technical words were replaced with more natural language words like instead of "Microservice

A calls Microservice B," which would be "Microservice A communicates with Microservice B." This step by step format aimed to facilitate deeper comprehension and more accurate responses from ChatGPT.

The exact Role and Context Prompt are:

Role Prompt: *"I want you to act as a great developer with best practices in Microservices project. I will give you the information of a Microservice during the following messages. I will expect that in each of those messages, you can answer as simply as: got it. When I am finished I will let you know and start asking you questions about it."*

Context Prompt: *"In the path $< classFilePath >$. There is a method called $< methodName >$. With this code: $< sourceCode >$. Finally, this is information about Crud and the Control Flow (sequence of calls): $< PO - CCG >$. Remember that for this message, you must answer only got it."*

5.2.5 ChatGPT Question Answering Process

The approach for integrating the results from Source Code Extraction, PO-CCG Construction, and subsequent Natural Language Transformation, along with the Engineered Prompts, to consolidate and present the information to ChatGPT for answering a specific question was as follows:

(1) Pass first to ChatGPT the Role Prompt.

(2) Next, given the predefined microservices involved in getting the answer to the given question, filter the Source Code Extraction data and the PO-CCG natural language information related to only those microservices. This filtering specifically targets the methods within their controllers and services.

(3) The extracted information and PO-CCG in the natural language form corresponding to each method is used to build the Context Prompt.

(4) Each Context Prompt is passed to ChatGPT as a separate message until no Context Prompts are left. Every message corresponds to a single method from the microservices involved.

(5) Finally, the task prompt is constructed based on the question and passed to ChatGPT to generate the answer.

5.3 Experimental Design

This section comprehensively examines the experiment conducted and its related artifacts. The experiment aims to evaluate the effectiveness of the ChatGPT model in answering questions about microservice systems, which is central to addressing the research questions outlined in this study.

The following subsections will cover several key elements. First, they describe the testbench used in the study. Next, they detail the implementation of the proposed methodology. They then explain the variations of the experiments carried out. Finally, they discuss the analytical methods applied to the collected data and how these contribute to answering the research questions.

5.3.1 Testbench

The TrainTicket[1] is used, a widely recognized microservice testbench introduced in Zhou et al. (2018). This testbench simulates a real world microservice architecture and is well regarded in software engineering.

The system features a decentralized database architecture, where each microservice is linked to its database. Most microservices use MongoDB, with one exception utilizing MySQL. It also incorporates modern cloud-native features such as containerization and advanced routing mechanisms. The setup includes 41 microservices and a dedicated UI project. Of these, 37 microservices are developed in Java, 2 in Python, 1 in Go, and 1 in NodeJS. The Java microservices follow

[1]TrainTicket v0.1.0: `https://github.com/FudanSELab/train-ticket/tree/0.1.0`

enterprise conventions, with a layered architecture including controllers, services, and repositories, and use the Spring Boot framework [2] along with standard Spring annotations. Communication between services is handled via REST API calls.

The system's Java codebase consists of 584 files with a total of 29,003 lines of code and 4,445 lines of comments, according to cloc statistics from GitHub [3]. The total token count for these Java files is 263,846, as calculated using tiktoken [4] with the 'cl100k_base' encoding scheme. The source code is publicly available at this GitHub repository[5].

5.3.2 Customized Questions for TrainTicket Testbench

A set of generic questions was first used to align the evaluation with the TrainTicket system, which was then customized to fit the specifics of the TrainTicket microservice architecture. This adaptation resulted in a set of tailored questions. The complete dataset of these questions, along with their categories and answers, is available at this link[6].

The study encompasses 33 questions, detailed in Table 5.3 and Table 5.4. These questions are organized into three categories: 8 questions address endpoint details, 13 focus on remote calls, and 12 examine dependencies. This range of questions covers various scenarios within the TrainTicket system, targeting different aspects of its microservices.

The questions vary in complexity, with answers requiring anywhere from 1 to 4 microservices, as detailed in the table. The selection of microservices for each question is carefully considered to avoid bias toward specific microservices and to

[2]Spring Boot: `https://spring.io/projects/spring-boot`

[3]Cloc: `https://github.com/AlDanial/cloc`

[4]Tiktoken: `https://github.com/openai/tiktoken`

[5] Prototype: `https://github.com/cloudhubs/Microservice-AI-Reasoning`

[6]Dataset: `https://zenodo.org/record/8358519`

Table 5.3. Experimental Questions

Cat.	Index	**Question**	**Involved microservices**
Endpoint Details	1	How many endpoints exist in ts-contacts-service microservice? Could you list them?	1. ts-assurance-service
	2	What are the input parameters for "modifyOrder" GET endpoint in ts-order-service, and what do they represent?	1. ts-order-service
	3	What is the expected output or result of modifyOrder GET endpoint in ts-order-service? Give me the exact Response object built in every case.	1. ts-order-service
	4	Which CRUD operations (CREATE, READ, UPDATE, DELETE) does the "modifyOrder" GET endpoint in "ts-order-service" execute?	1. ts-order-service
	5	Describe the data transfer object returned from the "findAllFoodOrder" GET endpoint in ts-food-service request in the User microservice.	1. ts-food-service
	6	What are the microservices and the endpoint for adding a new station?	1. ts-station-service
	7	7. What is the endpoint for creating a new route?	1. ts-route-service
	8	What HTTP methods are the path and the supported by the ts-assurance-service microservice's "modifyAssurance" endpoint?	1. ts-assurance-service
Remote Calls	9	How many total endpoint requests exist in ts-food-service microservice to other microservices? Could you list them?	1. ts-food-service
	10	How does "getAllFood" function endpoint in Microservice "ts-food-service" interact with other functions or services within this microservice architecture? And list the URLs.	1. ts-food-service
	11	How does "getTrainTypeByTripId" function endpoint in microservice "ts-travel2-service" interact with other functions or services within this microservice architecture? Please specify the other function calls, rest api requests, and the accessed data entities.	1. ts-travel2-service
	12	Can you identify the chain of calls (call graph) from the getAllContacts endpoint in ts-admin-basic-info-service to the findAll repository method in the ts-contacts-service microservice?	1. ts-admin-basic-info-service 2. ts-contacts-service
	13	Which endpoints in the "ts-food-service" microservice make calls to the "ts-station-service" microservice?	1. ts-food-service
	14	14. Can you identify the inter-service calls from the "getAllFood" GET endpoint in "ts-food-service" to the queryById GET endpoint in the "ts-route-service" microservice? Could you list them?	1. ts-food-service 2. ts-travel-service 3. ts-route-service
	15	Can you identify the chain of calls (call graph) between the "ts-food-service" and "ts-route-service" microservices in the following scenario?: A user calls the service in charge of showing all food on the user's trip. Please include services and controllers calls too.	1. ts-food-service 2. ts-travel-service 3. ts-route-service
	16	Can you identify any cyclic dependencies in the call graph of the ts-food-service microservice? If so, which endpoints are involved?	1. ts-food-service 2. ts-travel-service 3. ts-route-service

prevent the influence of a single microservice's characteristics across multiple questions within the same category. This approach ensures a fair and unbiased examination of various aspects of different microservices in the system. Consequently, these questions collectively address 15 distinct microservices, providing comprehensive coverage of the main components and interactions in the examined TrainTicket system.

5.3.3 Prompt Engineering

The previous section on Prompt Engineering within the Methodology covered the three types of prompts: Role, Context, and Task, and provided details on the

Table 5.4. Experimental Questions Cont.

Cat.	Index	**Question**	**Involved Microservices**
Dependency	17	How many endpoint routes exist between microservices ts-food-service and ts-food-map-service? Could you list them?	1. ts-food-service 2. ts-food-map-service
	18	How many endpoint routes exist between microservices ts-food-service and ts-station-service? Could you list them?	1. ts-food-service 2. ts-station-service
	19	How many endpoint routes exist between microservices ts-food-service and ts-price-service? Could you list them?	1. ts-food-service 2. ts-price-service
	20	How many distinct microservices are called from microservice ts-preserve-service?	1. ts-preserve-service
	21	How many distinct microservices are called from microservice ts-food-service?	1. ts-food-service
	22	Identify the microservice endpoints that the "ts-preserve-service" microservice has a direct dependency on them.	1. ts-preserve-service
	23	Identify the microservice endpoints that the "ts-preserve-service" microservice has a direct dependency on them.	1. ts-preserve-service
	24	Which microservice endpoints are impacted if the ts-price-service microservice's endpoint for food ordering experiences an outage?	1. ts-basic-service 2. ts-food-service
	25	Identify the microservice endpoints that directly or indirectly rely on the "ts-user-service" microservice.	1. ts-preserve-service 2. ts-preserve-other-service
	26	Describe the service dependency graph for ts-price-service.**	1. ts-price-service 2. ts-basic-service 3. ts-ticketinfo-service
	27	Describe the service dependency graph for ts-food-service. **	1. ts-food-service 2. ts-perserve-service 3. ts-perserve-other-service 4. ts-station-service
	28	Identify the microservice endpoints that have a dependency on the "ts-food-service" microservice's endpoints. **	1. ts-food-service 2. ts-preserve-service 3. ts-preserve-other-service
	29	Identify the microservice endpoints that the "ts-preserve-service" microservice depends on when preserving an order ticket. **	1. ts-preserve-service
	30	Identify the microservice endpoints that directly rely on the "ts-food-service" microservice. **	1. ts-preserve-service 2. ts-preserve-other-service
	31	Identify the microservice endpoints that the "ts-preserve-service" microservice depends on when preserving an order ticket. **	1. ts-preserve-service
	32	Identify the microservice endpoints that directly rely on the "ts-food-service" microservice.**	1. ts-preserve-service 2. ts-preserve-other-service
	33	Identify the microservice endpoints that the "ts-food-service" microservice has a direct or indirect dependency on them. **	1. ts-food-service

** Indicates the questions necessitating contextual information related to the Task prompt, such as integrating the definition of service dependency graph.

Role and Context Prompts. This section elaborates on how the Context Prompt is adjusted for different experiments and illustrates how to transform an initial question into the final Task Prompt.

Modifications are made to the Context Prompt based on the experiment's requirements. For example, if the experiment focuses solely on using source code, all sections related to the PO-CCG are omitted. Conversely, the source code section is excluded if the aim is to test responses based only on PO-CCG in natural language

form. Regardless of the modifications, the class file path and method name are consistently included in the Context Prompt.

For the Task prompt, provide an example of a question (marked by ** in Table 5.4) that it was going to turn out confusing to the model based on the ambiguity of the term dependency and the adaptation by the Prompt Engineering strategy described in the Methodology section:

Original Question: "Identify the microservice endpoints that directly rely on the "ts-food-service" microservice." This is question number 32 in Table 5.4

Then, ChatGPT was asked to define the term "service dependency graph" and, with that definition and the original question, create a prompt that it could understand, but always including the definition at the beginning.

Adapted Question: "A service dependency graph is a representation that shows the interdependencies between various services in a system, especially in the context of microservices architecture. Each node in the graph represents a microservice, and the directed edges between nodes represent one service's dependency on another. In other words, if Microservice A calls Microservice B, there would be an edge from A to B. Given that context answer, I want you to identify the microservice endpoints on which the "ts-food-service" microservice is directly dependent.

Now, let's also discuss this example because there is one more modification. After some analysis, it was decided the word *call* was too technical, counterintuitive, and potentially ambiguous for ChatGPT. Therefore, the call was replaced with communicate and the final task prompt was:

Task Prompt: *"A service dependency graph is a representation that shows the interdependencies between various services in a system, especially in the context of microservices architecture. Each node in the graph represents a microservice, and the directed edges between nodes represent one service's dependency on another. In other words, if Microservice A communicates with Microservice B, there would be an*

edge from A to B. Given that context, answer the following: I want you to identify the microservice endpoints that the "ts-food-service" microservice has a direct or indirect dependency on."

5.3.4 Methodology Implementation

This section provides a technical overview of the implementation details for the proposed methodology, covering the following phases: 1. Source Code Extraction,PO-CCG Construction, NL Transformation, and ChatGPT Question Answering Process.

5.3.4.1 Source Code Extraction:. The prototype is tailored for Java microservices designed around a layered application structure comprising controllers, services, and repositories built atop the Spring Boot framework. The tool requires the path to the source code to begin the analysis.

Java source files are identified using Apache Commons IO[7], which filters files by the ".java" extension. Each file is then analyzed using the javaparser[8] library. This static analysis process identifies key components of the code, including packages, classes, interfaces, and specific methods, while excluding basic methods such as getters, setters, and constructors.

A CSV file as output is generated, cataloging the methods detected and the classes to which they're associated, as referenced in Section 5.2.1. Although most data points are directly extracted using the javaparser library, class categorization is based on specific Spring annotations: classes with @Controller are classified as controllers, @Service as services, @Repository as repositories, and classes without these annotations are labeled as "other." It's important to note that the source code extraction primarily targets methods, so the CSV file's row count reflects the total number of methods in the system.

---

[7]Apache Commons IO: `https://commons.apache.org/proper/commons-io/index.html`

[8]Javaparser: `https://javaparser.org/`

5.3.4.2 PO-CCG Construction:. The prototype for CCG extraction Abdelfattah et al. (2023) (accessible at GitHub [9]) was employed to generate the CCG in a JSON format. Furthermore, the persistence operation extension prototype Abdelfattah et al. (2023) (available at [10]) was used to pinpoint persistent operations, which were also outputted in a JSON format. The newly developed prototype tool takes these two JSON files as input. It then associates each previously extracted method with its corresponding CCG and persistent operation, provided they exist in the inputted JSON files.

5.3.4.3 NL Transformation:. The implementation of the procedure described in Section 5.2.2 was conducted in the Python programming language (the source code is available at this repository [11]). The script takes as input a Comma-Separated-Values (CSV) file that contains columns for the Persistent Operations and the Control Flow Graph and produces a new CSV file with an added column of the description.

5.3.4.4 ChatGPT Question Answering Process:. The experiments were primarily conducted on Jupyter Notebook [12] using the Python programming language. The main libraries used were the Pandas libraries [13] to handle the dataset of information condensed in the CSV file [14] and the OpenAI [15] library as the primary tool for interfacing with the ChatGPT model. This library facilitated real-time communication and data processing with the model, enabling us to pose questions and receive responses systematically.

---

[9] `https://github.com/cloudhubs/Distributed-Systems-Semantic-Clone-Detector`

[10] `https://github.com/cloudhubs/authz-flow-analysis`

[11] `https://github.com/cloudhubs/Microservice-AI-Reasoning`

[12] Jupyter: `https://jupyter.org/`

[13] Pandas Lib:`https://pandas.pydata.org/`

[14] `https://zenodo.org/record/8358519`

[15] OpenAI: `https://github.com/openai/openai-python`

With the above tools, created a function called **askMicroserviceExpertChatGPT(question, microservicesNames)** that receives as input a set of microservice names and a question, and the experimental workflow was the following:

(1) For each question, compile a list of the involved microservices and use this list, along with the question text, as inputs for the function.

(2) Next, utilize Pandas to read the CSV file containing detailed information about the Microservice Project, including class file paths, method names, method source code, and PO-CCG in natural language form for each method across all microservices in the project.

(3) Then, filter the CSV information based on the microservice names by examining the class file path column.

(4) Apply an additional filter using the class file path to ensure that only methods from the Controllers and Services of the involved microservices are included.

(5) Another filter was applied to exclude any methods related to testing.

(6) Next, create a list of messages and set the Role Prompt as the initial message.

(7) Then, iterate over each remaining row in the CSV (after applying the filters). The classFilePath, methodName, methodSourceCode, and PO-CCG-NL columns were used to construct a Context Prompt for each row. This prompt was then added to the list of messages.

(8) Next, iterate through the list of messages, sending each one to ChatGPT using the OpenAI library until all messages were processed.

(9) Finally, send the last message containing the Task Prompt and returned the final answer.

5.3.5 Execution of the Experiments

For the experimental execution, ChatGPT's "gpt-3.5-turbo-16k," a variant of OpenAI's Generative Pre-trained Transformer (GPT) model Radford et al. (2018); Ray (2023); Hörnemalm (2023) was used. This specific version is optimized for efficiency and performance, making it suitable for various applications, from casual conversations to specialized tasks requiring a nuanced understanding. One of the key features of the "gpt-3.5-turbo-16k" variant is its ability to process up to 16,000 context tokens in a single interaction. This extended context window enables the model to handle longer conversations or detailed content more effectively, capturing intricate details and providing coherent, context-aware responses throughout extended interactions.

The general experimental execution involved passing each question and the corresponding microservice names to **askMicroserviceExpertChatGPT(question, microservicesNames)** function to obtain the answers for further evaluation, as described in the subsequent sections. Additionally, to ensure the stability and repeatability of the results, this process was repeated for each question three times. The primary objective was to confirm that the output remained consistent across these iterations.

However, the description above applies only to the general case where included all the information class file path, method name, method source code, and PO-CCG in natural language form in the Context Prompt. In addition to this, two other experiments were done: one using only the source code without the PO-CCG, and another using only the PO-CCG in natural language form without the source code.

It's important to note that ChatGPT has context limitations, with the maximum available context length for API access being 16k tokens in the gpt-3.5-based versions [16]. Due to these constraints, the focus was specifically on the Services and Controllers of each microservice involved. The inherent limitations associated with

[16] `https://help.openai.com/en/articles/7102672-how-can-i-access-gpt-4`

ChatGPT's context capacity were addressed by deliberately narrowing the scope to only Services and Controllers.

In summary, the experimental execution comprises three distinct scenarios to assess the impact of various data types on the responses:

Source Code Context: In this scenario, remove from the Context Prompt the PO-CCG information.

PO-CCG Context: In this scenario, remove from the Context Prompt the method's source code information and keep only the method's name and class file path beside the PO-CCG in natural language form.

Comprehensive Context: In the final scenario, integrate the source code and PO-CCG in the Context Prompt as described previously in the Prompt Engineering section.

All implementation and parameters set to reproduce these experiments can be found on a GitHub repository [17].

5.3.6 Evaluation of Answers

The evaluation is started by thoroughly analyzing the source code to derive the correct answers for each question directly from the system. Following this, compare these actual answers with those provided by ChatGPT during the experiment. The evaluation encompassed responses obtained from different knowledge bases: source code, the PO-CCG approach, and a combination of both source code and PO-CCG.

Given that the questions were intentionally designed to elicit specific lists of items, it was recognized that ChatGPT's responses might occasionally include either more or fewer items than anticipated. The focus was on assessing the accuracy of the answers, particularly their effectiveness in identifying the expected responses. In instances where ChatGPT generated additional correct answers unrelated to the

[17] `https://github.com/cloudhubs/Microservice-AI-Reasoning`

specific question, they were excluded from the formal evaluation but considered in the discussions. Consequently, the evaluation was guided by the following criteria:

- Correct: The number of items that the system correctly identified.
- Spurious: The number of items incorrectly predicted by the system.
- Missing: The number of items that the system was expected to mention but failed to do so.

To quantify the evaluation, the F_1 score was calculated, which measures the answers' accuracy. It combines the notions of correct, spurious, and missing into Precision and Recall scores, which are then aggregated into a single F_1 score. Here are the equations that:

$$P = \frac{C}{C+M} \qquad R = \frac{C}{C+S} \qquad F_1 = \frac{2 \cdot P \cdot R}{P+R} \tag{5.1}$$

Where C, M, S, P, and R represent Correct, Missing, Spurious, Precision, and Recall, respectively.

The F1 score offers a balanced evaluation of a model's performance by taking into account both its precision how accurately it identifies positive predictions and its recall how effectively it captures all relevant positive instances.

5.3.7 Analysis

To address RQ1, the focus was on evaluating the average F_1 score achieved across different scenarios and analyzing the standard deviation of these results. The outcomes were examined based on the categories defined in Section 5.2.3 and considered the results according to the number of microservices involved in the answer.

Subsequently, a statistical analysis was conducted to determine whether the model performed better in certain question categories than others. The hypothesis tests were designed as independent samples omnibus mean tests, as each category

contains different questions. In cases where the omnibus test yielded significant results, post-hoc tests were performed.

For RQ2 and RQ3, another series of tests was conducted to investigate whether statistical evidence supports the claim that the system performs better using one knowledge base compared to others. These hypothesis tests were formulated as paired samples omnibus mean tests, as different conditions (knowledge bases) were tested across the same subjects (questions). Post-hoc tests were carried out if the omnibus test indicated a significant result. Additionally, these tests were performed independently across various categories to identify any significant differences within specific categories rather than just the overall results.

Notably, several scenarios are used to evaluate the system and address the research questions. Each scenario is based on a specific subset of the questions. These scenarios include:

Overall: All the questions

Full Context: Questions where the Source Code-based scenario received the prompt for the full context of the question, namely, the relevant Controller and Service source code.

Non-full Context: The complement of Full Context.

Remote Calls: Questions in the Remote Calls category.

Endpoint Details: Questions in the Endpoint Details category.

Dependency: Questions in the Dependency category.

One Microservice: Question whose answer involved one microservice.

Two Microservices: Question whose answer involved two microservices.

More-than-two Microservices: Question whose answer involved more than two microservices.

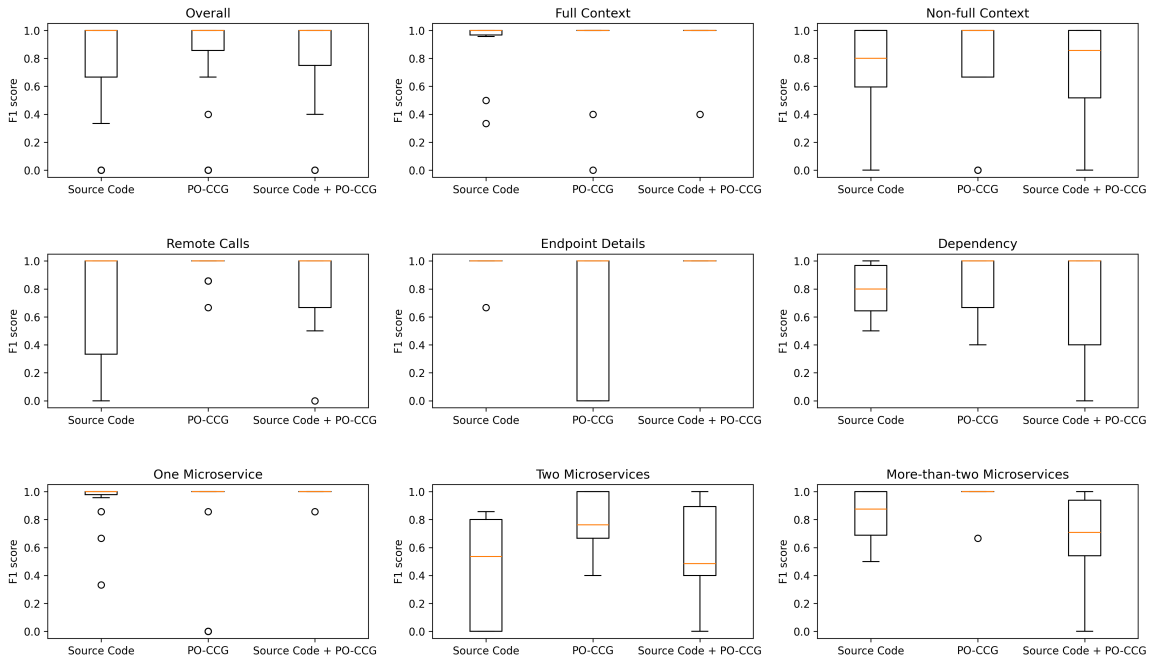

Figure 5.3. Boxplot graphs of the F_1 score in each evaluation scenario.

Figure 5.3 shows bloxplot representations of the results in each evaluation scenario described.

Note that in nearly all instances, the median result represents the highest possible score. The most challenging questions seem to be those involving multiple microservices. Although the 50th percentile scores are high, there is significant variability in the quality of the answers. Based on these results and qualitative assessment of the responses, this was the answer provided to RQ_1.

RQ_1 *ChatGPT can leverage source code to answer questions about microservice and interaction views at varying levels of complexity with differing performance outcomes. When using only the source code, it achieves a mean F_1 score of 0.794, accompanied by a relatively high standard deviation of 0.307. The primary challenges observed during the experimentation that impacted these results include: Context Limitation: Even with a reduced context focused solely on Controllers and Services, questions involving many microservices were challenging to test with source code alone. Prompt Brittleness: The prompts used could yield incorrect results, with minor changes*

potentially causing significant variations in the answers. Hallucinations: This issue involves ChatGPT generating outputs that appear plausible but are factually incorrect or not present in the provided context. The model might produce non-existent methods, classes, or connections that were never included in the source code.

To illustrate the hallucination phenomenom that was experienced we provide the following examples:

(1) When the task was: "Which CRUD operations (CREATE, READ, UPDATE, DELETE) does the *modifyOrder* GET endpoint in "ts-order-service" execute?" The correct answer, in summary, was: "Read and Update". However, we got the hallucinated answer: "Create and Read".

(2) When the task was: "Which endpoints in the *ts-food-service* microservice make calls to the *ts-station-service* microservice?" The correct answer was simply: "The GET endpoint getAllFood". However, we got the hallucinated answer: "getAllFood, updateFoodOrder, findByOrderId ".

(3) When the task was: "Identify the microservice endpoints that directly or indirectly rely on the *ts-user-service* microservice." The correct answer was in summary: "POST PreserveController.preserve and POST PreserveOther-Controller.preserve". However, we got a large hallucinated answer that lists three microservices which is not part of the task and lists from them multiple endpoints not involved in this like: *calculate, updateUser, getAllUser*, and others.

Next, Table 5.5 shows the results of running a Friedman test Friedman (1937) to test the null hypothesis that the means of all results are the same. Repeated-measures ANOVA was not considered because the data failed to pass the Shapiro-Wilk normality test.

It can be observed that for a significance of $\alpha = 0.05$, there is no scenario in which there exists evidence to reject the null hypothesis. Under a significance

Table 5.5. Results of Friedman's test in several scenarios. In each case, the final statistic and the p-value are reported.

Scenario	Statistic	P-value
Overall	2.10	0.350
Full Context	1.20	0.549
Non-full Context	2.80	0.247
Endpoint Details	3.50	0.174
Dependency	1.93	0.381
One Microservice	3.71	0.156
Two Microservices	1.75	0.417
More-than-two Microservices	4.80	0.091

Table 5.6. Friedman's post-hoc tests for finding statistically relevant pairwise differences among the three knowledge bases in the More-than-two Microservices scenario.

	Source Code	PO-CCG	Source Code+PO-CCG
Source Code	1.000000	0.648103	0.648103
PO-CCG	0.648103	1.000000	0.193400
Source Code+PO-CCG	0.648103	0.193400	1.000000

(a) Nemenyi test.

	Source Code	PO-CCG	Source Code+PO-CCG
Source Code	1.000000	0.309092	0.450886
PO-CCG	0.309092	1.000000	0.087673
Source Code+PO-CCG	0.450886	0.087673	1.000000

(b) Conover test

of $\alpha = 0.1$, the null hypothesis can be rejected in the More-than-two Microservices scenario. Nemenyi Nemenyi (1963) and Conover Conover and Iman (1981) post-hoc tests were done to assess the pairwise differences between system's capabilities with different knowledge bases. Pairwise p-values are shown in Tables 5.6a and 5.6b, respectively.

The Conover test showed a significant result at the 0.1 level, indicating a notable difference between using PO-CCG combined with source code and using only PO-CCG. In contrast, the Nemenyi test, which is more conservative, did not show a significant difference. This outcome aligns with expectations, as PO-CCG alone provides clearer, more direct answers to service interaction and dependency questions.

When combined with source code, the additional information may introduce ambiguity or confusion, potentially reducing the accuracy of ChatGPT's responses. This is consistent with the known limitation that large prompts and extensive contexts can impair the accuracy of LLMs.

These results set the stage for answering the remaining research questions.

RQ$_2$ *The experiments did not find statistical evidence that ChatGPT's performance improves with the use of Source Code compared to PO-CCG knowledge. However, the results suggest a slight advantage in using PO-CCG information.*

RQ$_3$ *Overall, the experiments did not provide statistical evidence that combining Source Code and PO-CCG improves ChatGPT's performance on microservice-related questions. Instead, in scenarios involving more than two microservices, the system performed better using PO-CCG information alone.*

5.4 Limitations

The metrics used to evaluate ChatGPT's performance such as Completeness, Relevance, Clarity, and Accuracy might not capture all aspects of an answer. The questions designed for the study may only cover specific facets of the Service View and Service Interaction perspectives, potentially limiting the evaluation. The study focused on a particular microservice project, so the results might differ for more complex or different projects. Additionally, the selected prompts might not be optimal for all contexts, and the best prompt for every scenario may not have been identified. The methodology was based on common Object-Oriented Programming (OOP) principles, which might not apply to projects using different paradigms. Although the alignment between source code and PO-CCG was validated manually, discrepancies could still exist.

Human evaluators assessing ChatGPT's responses could introduce biases or inconsistencies. To mitigate this, evaluators were provided with clear criteria and

cross-checked their assessments. Despite these measures, some subjectivity might still affect the results.

The findings may not generalize beyond the specific microservice project evaluated, even though they represent typical microservice architectures. Results might also be specific to the ChatGPT gpt-3.5-turbo-16 version and may not apply to other versions or models. The prompts' specificity could limit the results' applicability to different microservice types or contexts.

The number of questions used might be insufficient for drawing robust conclusions, despite rigorous development and broad applicability. Even with repeated questioning, the potential for variability in answers remains a concern. Prompt brittleness, where minor changes in prompts can significantly alter responses, is a known limitation of LLMs. Additionally, ChatGPT might produce hallucinatory responses, such as non-existent connections or endpoints, which was not addressed in the current study but will be considered in future work.

5.5 Discussion

The experiment highlighted the effectiveness of PO-CCG as a contextual source for ChatGPT, particularly within the constraints of token limits. For 18 out of 33 questions, using only controller and service details exceeded ChatGPT's token limit, while PO-CCG data, which fits within these constraints, provided satisfactory answers. However, PO-CCG struggled with questions requiring detailed implementation specifics or data model descriptions, whereas source code could have been more effective in extracting routes between microservices.

Effective prompt engineering significantly impacts ChatGPT's responses. Modifying keywords and clarifying concepts in prompts improved the model's understanding. For instance, explicitly defining the "service dependency graph" led to more accurate answers. ChatGPT also adeptly connected technical terms like

"outage" to their context. However, without proper definitions, the model sometimes misinterprets concepts.

ChatGPT showed a notable ability to infer correct answers from provided data, even when not all details were explicitly included in the prompt. For example, it accurately identified an endpoint from a related context. Yet, this adaptability can lead to fabricated responses when data is insufficient. The model occasionally missed parts of multi-faceted questions but performed well when questions focused on specific aspects. This behavior reflects ChatGPT's contextual adaptability and highlights areas for prompt refinement.

Now, from the perspective of the higher topic of automatic compliance checks of legal regulations in complex software systems, the results showed insights into the challenges and potential of using ChatGPT or similar for the automatic auditing of legal regulations in software systems. As evidenced by the varying performance outcomes across the research questions, the effectiveness of LLMs like ChatGPT in this context is contingent on several factors, including the system's complexity, the context provided, and the robustness of the prompts.

The finding that ChatGPT struggles with context limitations when analyzing complex microservice architectures highlights a significant challenge for automatic audits of legal regulations. In real-world applications, legal compliance often requires a comprehensive understanding of how different software system components interact, especially in complex architectures like microservices. The model's difficulty in managing and accurately interpreting many microservices based solely on source code suggests that, without adequate contextual framing, the model may overlook critical interactions that could indicate compliance or non-compliance with legal standards. Besides, any approach that will attempt to pass the agent not only the software source code, but lengthy legal regulations will be further affected by this limitation.

The brittleness of prompts and hallucinations are particularly concerning in legal compliance auditing. Inaccurate responses generated due to slight changes in prompts or fabricated information can lead to erroneous conclusions about a system's compliance status. Given the high stakes of legal audits, where incorrect interpretations could result in legal liability, these issues show the need for meticulous, prompt engineering and validation when using LLMs. The potential for hallucinations further complicates using LLMs in this domain, as it raises the risk of generating incorrect information that could mislead auditors, developers, or any user.

The results indicate that PO-CCG data generally performed better than source code in answering questions about microservice systems, particularly in scenarios involving multiple microservices. This finding suggests that more abstracted and structured representations like PO-CCG might be more effective for tasks requiring an understanding of high-level system interactions, which are often crucial in legal audits. The lack of significant improvement when combining source code with PO-CCG highlights the importance of selecting the right knowledge base for the task. For auditing purposes, using targeted, high-level abstractions may be more beneficial than relying on raw code, which may overwhelm the model with unnecessary detail and increase the risk of errors.

5.6 Conclusion

This chapter examines the rapidly evolving software development landscape, where microservice architecture has emerged as a dominant pattern. Microservices' decentralized nature and intricate cross-service dependencies make understanding and documenting such systems a complex task. While valuable, traditional documentation, summarization, and semantic analysis methods often need help to keep pace with these systems' dynamic evolution.

The exploration of the potential of (LLMs), exemplified by ChatGPT, as practical tools for understanding and querying microservice systems using source

code and the PO-CCG as a knowledge base, showed the following revealing findings: PO-CCG demonstrated a significant advantage in accurately responding to a broader range of questions within the constraints of the model's token limitations. This suggests a promising direction for organizations aiming to gain insights into their microservices landscape without the extensive overhead associated with traditional documentation methods.

However, while PO-CCG excelled in addressing service interactions and dependencies queries, it encountered challenges when dealing with granular implementation details or specific data model descriptions. ChatGPT's versatility proved to be both a strength and a limitation, as the model occasionally deduced answers beyond the specificity of the query, sometimes resulting in fabricated or incomplete responses. The findings highlighted the critical role of prompt engineering in guiding the model toward more accurate and meaningful outputs.

This chapter shows the potential and challenges of integrating LLMs like ChatGPT into software development, documentation, analysis, and legal compliance. While these models can only partially replace traditional documentation or legal human expertise, they are powerful tools for developers and architects, particularly in complex microservice environments. As LLMs become increasingly integrated into software development practices, their ability to assist in system comprehension, provide insights, and answer queries could revolutionize how organizations manage legal compliance. By working with human experts, LLMs can contribute to a more efficient and accurate understanding of complex systems, ultimately leading to more robust and compliant software architectures. Further research and exploration in this domain could pave the way for innovative software development practices, where LLMs play a critical role in ensuring legal and regulatory compliance across decentralized and rapidly evolving systems.

5.7 Credit

In this section, we will provide each author's contributions to the work presented in this chapter. For this, we will use the system CRediT(`https://www.elsevier.com/researcher/author/policies-and-guidelines/credit-author-statement`) from Springer to make it easier.

Ernesto Quevedo Caballero: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data Curation, Writing - Original Draft, Writing-Review Editing, Visualization, Project administration.

Amr S. Abdelfattah: Conceptualization, Software, Validation, Formal analysis, Data curation, Visualization, Writing-review and editing.

Alejandro Rodriguez: Conceptualization, Software, Validation, Formal analysis, Visualization, Writing-review and editing.

Jorge Yero: Conceptualization, Software, Validation, Formal analysis, Visualization, Writing-review and editing.

Tomas Cerny: Conceptualization, Investigation, Validation, Data curation, Visualization, Writing-review and editing, Supervision.

CHAPTER SIX

LLMs Hallucinations Detection Survey

The rapid development of Natural Language Generation (NLG) has revolutionized the field of natural language processing, driven by advancements in deep learning technologies, particularly Transformer-based Large Language Models (LLMs) like BART, LLaMa, GPT-3.5, GPT-4. These models have captured significant attention due to their impressive performance across various tasks, leading to an increased focus on their potential and limitations. Among the critical concerns is the phenomenon known as "hallucination," where models generate nonsensical text, unfaithful to the source content or factually incorrect. Such outputs pose significant risks, particularly in sensitive domains like legal, medicine, or privacy-sensitive applications, where inaccurate information can have severe consequences. For that reason, research interest in understanding and mitigating hallucinations across different NLG tasks has sparked. The growing body of research aims to detect and correct these errors and refine the underlying models to reduce their occurrence.

Despite the progress made, a comprehensive understanding of hallucinations across all major NLG tasks still needs to be completed. Chapters 2 and 4 clearly show that hallucinations in LLMs are a significant barrier to achieving the pinnacle of automatic legal compliance analysis in software systems. Therefore, detecting when an LLM generates hallucinations is essential to prevent the production of nonsensical or unfaithful answers. Such errors could lead developers or legal auditors to mistakenly assume that a software system is compliant when it is not or that a legal violation exists when it is not. Given its critical importance in the global landscape of machine learning and Natural Language Processing (NLP) and the scope of this thesis, this

chapter provides a literature review on LLM hallucination detection methods in this chapter.

Chapter 7 uses the insights gained from this review to address the challenging problem of detecting hallucinations. However, this chapter will focus primarily on methods devoted to detection. In contrast, Chapter 8 will delve into methods for mitigation, which often complement the original detection methods.

6.1 Definition

Based on the classifications established in earlier studies Dziri et al. (2021); Ji et al. (2023); Huang et al. (2023), hallucinations are divided into two main categories: intrinsic and extrinsic.

Intrinsic hallucination: occurs when a generated text is inconsistent or nonsensical based on the information within the text itself Ji et al. (2023). Example: "Albert Einstein, renowned botanist, discovered the relativity laws of physics."

Extrinsic hallucination: occurs when the generated output cannot be confirmed or disproven by the source content (i.e., the output is neither supported nor contradicted by the source) Ji et al. (2023). Example: "Albert Einstein devoted much of his free time to thinking about creating a new intellectual game."

It is also important to differentiate hallucination errors from other types of errors:

Biased outputs: emerge when an AI model's training data or algorithms carry embedded prejudices, resulting in outputs that mirror these biases, such as stereotypes or discriminatory patterns. These outputs aren't typically hallucinations but rather accurate reflections of the biases in the data the model was trained on. The model replicates aspects of reality, even when that reality is flawed or problematic. Addressing biases in AI is a crucial challenge

for ensuring the safety and fairness of AI systems. However, it is a distinct issue that can be addressed separately in the future.

Out-of-distribution errors: occurs when an AI model encounters input data vastly different from what it was trained on; it can generate unpredictable or illogical outputs. These situations reveal the model's limitations, making it clear that the input is beyond its capacity to handle effectively. This issue is essentially a generalization error, indicating either that the model's hypothesis space is too limited to encompass the true distribution of data fully or that the data and training methods used are inadequate for identifying a sufficiently broad hypothesis.

Hallucinations are more problematic than out-of-distribution errors because they occur within the input distribution, where the model is expected to perform reliably. Unlike biases, hallucinations involve the model generating information entirely fabricated or detached from reality, making detection much more challenging. The stochastic nature of generative models exacerbates this issue, as hallucinations can happen randomly once in 100 instances for the same input, rendering them nearly impossible to evaluate or debug consistently. The model's confident and coherent responses further complicate the identification of these errors, as there are no clear indicators for human evaluators to detect them easily.

6.2 Taxonomy of Hallucination Detection Methods

The analysis will be divided to present current works in hallucination detection, and the present works will correspond to the method used. Figure 6.1 illustrates the summary of the methods used to detect hallucinations in LLMs found in this chapter.

6.2.1 Retrieval methods

One intuitive strategy for pinpointing factual inaccuracies in LLM outputs is to compare the model-generated content against reliable knowledge sources.

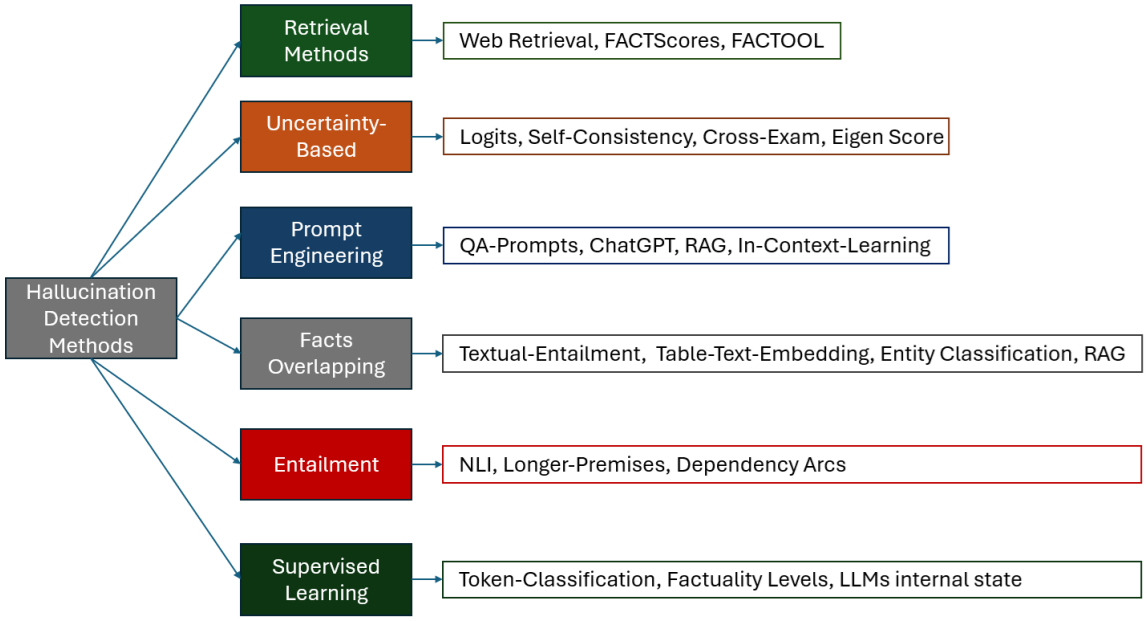

Figure 6.1. Summary of the review on Hallucination Detection in LLMs

This approach has gained significant attention in hallucination detection, where retrieval-based methods play a pivotal role. By leveraging external databases, search engines, or specialized corpora, these methods aim to verify the truthfulness of LLM-generated statements by cross-referencing them with authoritative sources Saxena and Bhattacharyya (2024); Huang et al. (2023); Ji et al. (2023).

This section reviews notable research efforts that utilize retrieval methods to detect hallucinations in LLM outputs. These studies explore various retrieval techniques, ranging from simple keyword-matching systems to more sophisticated neural retrieval architectures, each offering distinct advantages in enhancing the factual reliability of generative models. By integrating retrieval with LLMs, these works aim to balance efficiency and accuracy in identifying factual inconsistencies, demonstrating promising pathways for improving the trustworthiness of LLM-generated content. Retrieval-based approaches for hallucination detection involve comparing the generated text against relevant information retrieved from external databases or knowledge sources. These methods identify potential hallucinations by cross-referencing the generated content with verified data, ensuring that the information aligns with known

facts. The content is flagged as a potential hallucination if discrepancies between the generated output and the retrieved data are found. This approach leverages the vast amount of information available in external sources to enhance the accuracy of detecting factually incorrect or unsupported content in the generated text Saxena and Bhattacharyya (2024).

Chen et al. (2023), the authors tackle the challenge of automating fact-checking for real-world claims by creating a pipeline that retrieves raw evidence from the web. Traditional methods for fact-checking often make unrealistic assumptions, such as having access to curated evidence or information published after a claim has been made. In contrast, this study restricts the retrieval process to documents that were available before the claim was made, which better reflects real-world scenarios where claims emerge and must be verified in real time. The proposed pipeline comprises five key components: claim decomposition, raw document retrieval, fine-grained evidence retrieval, claim-focused summarization, and a final veracity judgment. Tested on complex political claims from the CLAIMDECOMP dataset Chen et al. (2022), the system demonstrated that aggregating retrieved evidence could improve the accuracy of fact-checking. Human evaluations further validated that the system's claim-focused summaries were reliable and relevant, without hallucinating information, thus supporting fact-checkers by providing useful summaries even if the full evidence set wasn't available. The study highlights the limitations of web searches in capturing all the necessary information to verify claims, emphasizing the complexity of evidence retrieval in real-world contexts. The authors suggest that integrating human oversight into the automated fact-checking process would further improve reliability, pointing toward a future where human-in-the-loop systems enhance the effectiveness of automated fact-checking workflows.

In the research Min et al. (2023), authors perform extensive human evaluations to calculate FACTSCOREs for biographies generated by several leading LLMs, such

as InstructGPT, ChatGPT, and PerplexityAI, revealing that models like ChatGPT achieve only 58%. To address the cost of human evaluation, the paper also presents an automated model for estimating FACTSCOREs using retrieval and a robust language model, achieving less than a 2% error rate. This automated metric is then applied to assess 6,500 generations from 13 recent LLMs, providing insights into model performance, such as GPT-4 and ChatGPT being more factual than public models and Vicuna and Alpaca ranking among the best public models.

The work of Galitsky (2023) presents "Truth-O-Meter," a fact-checking system designed to improve the factual accuracy of content generated by LLMs like GPT-4. LLMs often produce texts that contain inaccuracies or hallucinations, and Truth-O-Meter addresses this by comparing the generated text against web data and other information sources. The system uses text mining and web mining techniques to find correct corresponding sentences and employs syntactic and semantic generalization to enhance content quality. The system relies on defeasible logic programming for argumentation analysis to manage inconsistencies in the sources it analyzes. The authors compare Truth-O-Meter's performance to alternative methods, such as reinforcement learning-based approaches and token-based hallucination detection, and find that the system significantly enhances LLM-generated content's factual correctness and meaningfulness. Additionally, the system leverages advancements in content correction algorithms, which have improved alongside developing more sophisticated LLMs.

Huo et al. (2023) investigates whether LLMs can detect their hallucinations by verifying generated answers using an information retrieval system. The proposed pipeline presents a question to an LLM, receives the generated answer, and then queries a corpus combining the question and answer to retrieve supporting evidence. The LLM is then prompted to assess whether the retrieved evidence supports the generated response. The experiment is based on the MS MARCO (V1) test collection,

using various retrieval approaches, from the simple BM25 Robertson et al. (1995) to a more advanced question-answering stack with an LLM-based reader. The study finds that in most cases, the LLM can verify its responses when provided with relevant supporting material, achieving an accuracy of 70-80% in detecting hallucinations. However, the LLM needs help with answers involving numbers. While this method shows promise, it cannot be solely relied upon to detect hallucinations consistently. The retrieval method is crucial in improving answer verification, with more effective methods yielding higher-quality responses but often at the cost of efficiency. The research demonstrates that even a computationally inexpensive method like BM25 can perform acceptably in this task.

The research of Chern et al. (2023) introduces FACTOOL, a task and domain-agnostic framework designed to address the growing challenge of detecting factual errors in texts generated by LLMs, such as ChatGPT. It highlights key issues with generative models, including (1) the increasing risk of factual inaccuracies across a wider range of tasks, (2) the tendency for generated texts to be lengthy and lacking clear granularity for individual facts, and (3) the scarcity of explicit evidence available for fact-checking. With these challenges in mind, FACTOOL aims to provide a comprehensive and adaptable solution for identifying factual errors across various domains. FACTOOL is a five-step tool-augmented framework comprising claim extraction, query generation, tool querying, evidence collection, and verification. It integrates external tools such as Google Search, Google Scholar, code interpreters, and Python to detect errors in generated content. The framework also leverages LLMs themselves for fact-checking when applicable. The authors demonstrate the efficacy of FACTOOL through experiments across diverse tasks, including knowledge-based question answering (QA), code generation, mathematical reasoning, and scientific literature review writing. The results indicate that FACTOOL can detect factual

errors across different domains, and the framework can be easily extended to other use cases.

6.2.2 Uncertainty-based

Several techniques focus on detecting hallucinations without depending on external knowledge sources. These approaches are based on the idea that hallucinations arise from the model's inherent uncertainty. By measuring this uncertainty in the generated factual content, it is possible to identify hallucinations without retrieving additional evidence. The internal states of LLMs can reveal their uncertainty, which is often reflected in metrics such as token probability or entropy Saxena and Bhattacharyya (2024).

Varshney et al. (2023) propose a method for actively detecting and mitigating hallucinations during text generation. Their approach involves identifying potential hallucinations using the model's logit outputs, validating these candidates, and mitigating the detected issues while continuing with the generation. Experiments with GPT-3.5 demonstrate that their detection method achieves about 88% recall, and their mitigation technique successfully addresses 57.6% of detected hallucinations without introducing new errors. The overall hallucination rate is reduced from 47.5% to 14.5% on average. The approach is also tested on different question types, and another LLM, Vicuna, shows its broad effectiveness. This work advances the reliability and trustworthiness of LLMs, which is crucial for their practical deployment.

The research of Manakul et al. (2023) introduces "SelfCheckGPT," a straightforward sampling-based technique designed for fact-checking responses from black-box models without needing an external database. The core idea is that if an LLM genuinely knows a concept, its responses will be consistent, whereas hallucinated facts will lead to varied and conflicting responses. The approach was tested by generating passages about individuals using GPT-3 and manually assessing their factual accuracy. The results show that SelfCheckGPT effectively identifies factual

and non-factual sentences and ranks passages based on their factuality. By comparing SelfCheckGPT against various baseline methods, the authors show that their approach achieves significantly higher AUC-PR scores for sentence-level hallucination detection and higher correlation scores for passage-level factuality assessments than other fact-checking methods, including grey-box approaches.

The study Agrawal et al. (2023) focuses on hallucinated book and article references, which are particularly useful for investigating hallucinations due to their clear and identifiable nature. The critical insight is that a model should have adequate information if it cites a reference. The researchers demonstrate that hallucinated references can be detected by querying the model about the details of these references without needing external resources. These queries, or "consistency checks," reveal that while LLMs like GPT-4 often produce inconsistent information for fictitious references, they usually recall accurate details for genuine ones. This approach provides a method to understand and analyze the nature of hallucinated references in LLMs.

Cohen et al. (2023) propose Language Model Versus Language Model (LMVLM), a novel evaluation framework inspired by legal cross-examination techniques. The author's approach identifies inconsistencies in a model's claims by using a secondary LLM as an examiner who questions the original model. This multiturn interaction helps uncover discrepancies and factual errors. The authors empirically evaluate LMVLM on factual claims made by various recent LMs across four different benchmarks. Their findings indicate that LMVLM significantly outperforms existing methods and baselines in detecting factual errors, often achieving superior results by a considerable margin. They tested this method on several recent LLMs across four benchmarks and found that it significantly outperforms existing methods. Their findings highlight the effectiveness of using interacting LLMs to detect factual inaccuracies.

The research of Chen et al. (2024) introduces a novel method that explores the dense semantic information preserved within the internal states of LLMs for

hallucination detection, termed INSIDE (Internal States for Hallucination Detection). Central to this method is the EigenScore metric, a simple yet effective tool for evaluating the self-consistency of model responses by analyzing the eigenvalues of the responses' covariance matrix, thereby assessing semantic consistency and diversity within the dense embedding space. Additionally, the authors propose a test time feature clipping technique aimed at truncating extreme activations in the internal states, which reduces overconfident predictions and enhances the detection of overconfident hallucinations. The authors demonstrate their proposed approach's effectiveness through extensive experiments and ablation studies on several popular LLMs and question-answering (QA) benchmarks: CoQA Reddy et al. (2019), SQuad Rajpurkar et al. (2016), TriviaQA Joshi et al. (2017) and NaturalQA Kwiatkowski et al. (2019).

6.2.3 Prompting-based

LLMs' remarkable ability to adhere to instructions has demonstrated their potential for automating various tasks, including evaluating factual accuracy in generated content Chiang and Lee (2023). This inherent capability has led to innovative approaches in hallucination detection through prompting-based methods. By leveraging the LLM's capacity to interpret and act on specific instructions, researchers have developed techniques that utilize well-designed prompts to guide the model in identifying and addressing potential inaccuracies within its outputs.

This section explores research on prompting-based methods for detecting hallucinations in LLM-generated content.

Luo et al. (2023) investigate ChatGPT's ability to evaluate factual inconsistency in text summarization within a zero-shot setting, focusing on coarse-grained and fine-grained tasks. Recognizing the issue of factually inconsistent summaries generated by pre-trained language models, the authors explore ChatGPT's performance on three key tasks: binary entailment inference, summary ranking, and consistency rating. The results indicate that ChatGPT generally outperforms previous state-of-the-art

evaluation metrics across six out of nine datasets, showcasing its potential as an effective factual inconsistency evaluator. However, despite its strong performance, the authors identify several limitations in ChatGPT's evaluation process. These include a bias toward lexically similar candidates, instances of false reasoning, and occasional misunderstandings of instructions. To further improve ChatGPT's performance, the authors test the chain-of-thought (CoT) prompt, significantly enhancing the model's evaluation capabilities. However, they also note limitations in the CoT approach, emphasizing the need for continued research in model alignment to address these shortcomings.

The research Gao et al. (2023) investigates ChatGPT's ability to perform human-like text summarization evaluations using four established human evaluation methods: Likert scale scoring, pairwise comparison, Pyramid Nenkova and Passonneau (2004), and binary factuality evaluation. Conducted across five datasets, the results show that ChatGPT can smoothly complete annotation tasks and, in some cases, outperform widely used automatic evaluation metrics. The authors highlight that ChatGPT's effectiveness in summarization evaluation is highly dependent on the design of prompts. Key findings include ChatGPT's strong correlation with human judgments in some instances, particularly compared to existing evaluation metrics. The model's evaluations were more cost-effective and reproducible than human assessments and demonstrated consistency between the generated explanations and scoring. However, the authors emphasize that prompt design is crucial in determining ChatGPT's performance, underlining the importance of careful prompt engineering in achieving optimal results.

The work Adlakha et al. (2023) examines the effectiveness of retriever-augmented instruction following models in information-seeking tasks like question answering (QA) as an alternative to fine-tuned models. These models adapt to various domains by prepending retrieved documents with an instruction without additional

fine-tuning. Although the model outputs are generally natural and fluent, their verbosity makes traditional QA metrics, like exact match (EM) and F1, less reliable for evaluating performance. The authors use automatic and human evaluations to assess these models across three QA tasks. They focus on two key aspects: correctness (how well they meet the user's information needs) and faithfulness (whether they base their responses on the provided knowledge). The study finds that traditional metrics fail to capture the actual performance of these models, leading the authors to propose new token overlaps and model-based metrics that better reflect their capabilities. While the instruction-following models often perform as well as or better than fine-tuned models in terms of correctness, they need help maintaining faithfulness and frequently produce hallucinations. The authors advocate for a more comprehensive evaluation of these models in QA tasks.

Jain et al. (2023) explore the potential of LLMs to serve as multi-dimensional evaluators using in-context learning, eliminating the need for extensive training datasets. Their experiments demonstrate that evaluators based on in-context learning are competitive with traditional learned evaluation frameworks, achieving state-of-the-art results in relevance and factual consistency for text summarization. The research also examines how factors like the selection and number of in-context examples influence performance. Additionally, the authors investigate the effectiveness of in-context learning-based evaluators in assessing zero-shot summaries generated by LLMs like GPT-3.

The research of Laban et al. (2023) proposes a new protocol for creating benchmarks focused on inconsistency detection and introduces a 10-domain benchmark called SummEdits. This benchmark is 20 times more cost-effective per sample than previous ones and demonstrates high reproducibility, with an estimated inter-annotator agreement of about 0.9. However, most LLMs perform poorly on SummEdits, with results close to random chance. Even the best model, GPT-4, lags by 8% behind

estimated human performance, highlighting significant challenges in LLMs' ability to reason facts and detect inconsistencies accurately.

6.2.4 Facts Overlapping

It is essential for LLMs to accurately follow the context or user instructions in practical applications like summarization and interactive dialogue systems. Detecting faithfulness hallucinations centers on ensuring the generated content aligns with the provided context, avoiding irrelevant or conflicting information. To assess faithfulness, a typical approach involves comparing the overlap of key facts between the generated output and the source material Saxena and Bhattacharyya (2024); Huang et al. (2023).

Maynez et al. (2020) investigate the shortcomings of standard training and decoding methods in neural text generation models, particularly for tasks like abstractive document summarization. They found that these models often generate unfaithful content in the input document. To explore this issue, the authors conducted a large-scale human evaluation of various neural abstractive summarization systems, revealing that all models produced significant amounts of hallucinated content. However, they also found that pre-trained models performed better regarding standard metrics like ROUGE and generated more faithful and factual summaries according to human evaluations. Additionally, their analysis suggests that textual entailment measures correlate better with faithfulness than traditional metrics, pointing to potential improvements in automatic evaluation and training methods.

The research of Wang et al. (2020) addresses the challenge of generating text from a knowledge base while ensuring that the generated descriptions remain faithful to the original content. They identify that many existing methods fail to maintain this faithfulness, often resulting in text that includes information not in the source table. To solve this, the authors introduce a novel Transformer-based generation framework incorporating two key techniques: a table-text optimal-transport matching loss and a table-text embedding similarity loss designed to enforce faithfulness. They also propose

a new automatic metric for evaluating faithfulness in table-to-text generation. The authors demonstrate through detailed experiments and analyses that their framework significantly outperforms existing state-of-the-art methods, according to automatic and human evaluations.

The work of Nan et al. (2021) introduces new metrics specifically designed to measure the entity-level factual consistency of summaries called precision-source score (precs) to quantify entity hallucination and additional metrics (prect and recallt) to assess entity accuracy in generated summaries. The study finds that existing datasets, like XSUM Narayan et al. (2018), contain a high level of entity hallucination in ground truth summaries. To address this, the authors propose a data filtering technique that removes hallucinated entities from training data. Their experiments demonstrate significant improvements in entity-level factual consistency, with the precs score increasing from below 94% to above 98% on the XSUM dataset. Additionally, the paper explores multi-task learning and joint sequence generation approaches to improve entity accuracy further. These combined methods effectively reduce entity hallucination while maintaining strong ROUGE scores, offering a robust solution to improve factual consistency in abstractive summarization.

Goodrich et al. (2019) introduce a model-based metric designed to assess the factual accuracy of the generated text, complementing standard scoring metrics like ROUGE and BLEU. They create and release a large-scale dataset derived from Wikipedia and Wikidata, which is used to train relation classifiers and end-to-end fact extraction models. These models effectively extract complete fact sets from full-text datasets. The authors then evaluate various models for estimating factual accuracy in the context of Wikipedia text summarization and demonstrate their effectiveness compared to ROUGE and other model-free methods through a human evaluation study.

The study of Shuster et al. (2021) investigates using neural retrieval-based architectures to enhance knowledge-grounded dialogue systems, addressing a common issue in current dialogue models: factual inaccuracies and hallucinations. They explore various architectures incorporating components like retrievers, rankers, and encoder-decoders to improve knowledge accuracy while maintaining conversational fluency. Their best models achieve state-of-the-art performance on two knowledge-grounded dialogue tasks, showing solid conversational skills, effective generalization to new scenarios, and a significant reduction in knowledge hallucination, as confirmed by human evaluations.

6.2.5 Entailment methods

Leveraging Natural Language Inference (NLI) to evaluate the trustworthiness of text has become a widely adopted approach. This method involves analyzing the logical relationship between statements to determine their validity and coherence. By applying NLI techniques, it can be assessed whether the content of a text aligns with facts or other reliable information sources, helping to identify discrepancies and potential inaccuracies. This approach is precious in distinguishing between credible and unreliable information, enhancing the overall quality and reliability of text-based assessments Ji et al. (2023); Saxena and Bhattacharyya (2024); Huang et al. (2023).

Falke et al. (2019) evaluates the summaries generated by leading models through crowdsourced assessments, revealing frequent factual inaccuracies, especially with more abstractive approaches. The authors investigate whether textual entailment predictions can help detect these errors and whether reranking alternative summaries can improve accuracy. They find that current entailment models, trained on Natural Language Inference (NLI) datasets, still need to meet performance expectations for this task. To address this, they provide their annotations as additional test data to support future evaluations of entailment models.

The work of Mishra et al. (2021) explores why by examining the multiple-choice reading comprehension (MCRC) and factual correctness checking of textual summarization (CFCS) tasks. The research reveals that traditional NLI datasets, which contain shorter premises, pose a challenge for downstream tasks that benefit from longer contexts. The authors propose converting extensive reading comprehension datasets into NLI datasets with more extended premises to address this. They find that models trained on these newly created longer-premise datasets perform better downstream tasks than those trained on traditional short-premise NLI datasets. This improvement is attributed to the length of the premises used in training.

The research of Goyal and Durrett (2020) introduces a novel approach to entailment by breaking it down into dependency arcs within sentences. Instead of making overall judgments, the proposed method examines whether the input supports each dependency arc in the generated text. Since obtaining human judgments for this task is challenging, the authors suggest creating automatic data from existing entailment or paraphrase corpora. Their experiments demonstrate that this dependency arc entailment model, trained on this data, is more effective at detecting and localizing factual inconsistencies in paraphrasing and summarization tasks than traditional sentence-level methods or those based on question generation.

6.2.6 Supervised Learning methods

Supervised learning methods for hallucination detection in LLMs leverage labeled datasets to train classifiers that identify incorrect or misleading outputs generated by these models. These approaches involve training a model on a corpus where hallucinations are annotated, allowing the system to learn patterns and features associated with erroneous content. One advantage of supervised learning is its ability to create highly specialized detectors tailored to specific hallucinations by learning from real-world examples. However, these methods require extensive labeled data, which can be costly and time-consuming to produce. Additionally, the effectiveness

of supervised models can be limited by the quality and representativeness of the training data, potentially resulting in detectors that may not generalize well to novel or unforeseen types of hallucinations

Zhou et al. (2020) propose a task that predicts whether each token in the generated sequence is a hallucination (i.e., not found in the input). They also create new manually annotated evaluation sets for this task. Additionally, they introduce a method that detects hallucinations by fine-tuning pre-trained language models on synthetic data containing automatically inserted hallucinations. Through experiments on machine translation (MT) and abstractive summarization, they show that their approach consistently outperforms strong baselines across all benchmark datasets. Moreover, they demonstrate that token-level hallucination labels allow loss calculation over the target sequence in low-resource MT, leading to significant improvements over solid baseline methods. The authors also apply their method to word-level quality estimation in MT, proving its effectiveness in supervised and unsupervised scenarios.

The research of Dziri et al. (2021) introduces the Benchmark for Evaluation of Grounded Interaction (BEGIN). BEGIN includes 8,113 dialogue turns generated by LLM-based systems and human annotations describing the relationship between the system's responses and the background information. These annotations are based on an extended natural language inference framework. The authors use this benchmark to show how adversarially generated data can enhance an evaluation metric derived from existing natural language inference datasets. Using BEGIN, the authors evaluate eight automatic metrics commonly used for response attribution. Their analysis reveals that these metrics rely heavily on spurious correlations, such as word overlap, and fail to distinguish between attributable and unattributable abstractive responses reliably. Furthermore, they perform significantly worse when the knowledge source is longer, highlighting their limitations in more complex contexts. The study emphasizes the

need for more sophisticated and robust evaluation metrics for knowledge-grounded dialogue systems.

The work of Santhanam et al. (2021) investigates factual correctness in knowledge-grounded neural response generation models. They introduce a human annotation framework to categorize responses into three types: those that are factually consistent with the input knowledge, those that include hallucinated knowledge, and non-verifiable chitchat-style responses. Using this framework, they annotate responses generated by various state-of-the-art models, knowledge snippets, and decoding strategies. To further aid the development of a factual consistency detector, they created a new Conv-FEVER dataset, adapted from the Wizard of Wikipedia dataset, which contains both factually consistent and inconsistent responses. The authors show that models trained on the Conv-FEVER dataset perform well in detecting factually inconsistent responses relative to the provided knowledge, as demonstrated by evaluations of their human-annotated data.

Azaria and Mitchell (2023a) present evidence that the internal states of LLMs can be leveraged to assess the truthfulness of statements. This assessment applies to statements provided to the LLM and those generated by the LLM itself. The proposed approach involves training a classifier that predicts the likelihood of a statement being truthful based on the hidden layer activations of the LLM as it processes or generates the statement. Experimental results demonstrate that the trained classifier achieves an accuracy rate between 71% and 83% in distinguishing true statements from false ones, depending on the base LLM model used. The authors further investigate the relationship between their classifier's performance and traditional methods that rely on the probability assigned to a sentence by the LLM. They find that while LLM-assigned probabilities correlate with truthfulness, they are also influenced by sentence length and word frequency factors. As a result, their trained classifier offers a more reliable

method for detecting truthfulness, showcasing its potential to improve the reliability of LLM-generated content and its practical applicability in real-world settings.

6.3 Discussion

The phenomenon of hallucination in LLMs has emerged as a significant challenge, particularly in fields where accuracy and fidelity are essential, such as legal compliance and the assessment of software systems. As the deployment of LLMs in these domains becomes more widespread, the risks associated with hallucinations cannot be overstated. Hallucinations, which can manifest as factually incorrect, misleading, or nonsensical outputs, can undermine the reliability and credibility of automated systems, leading to severe legal and financial consequences.

Hallucinations pose unique and complex challenges in the context of automatic legal compliance and software assessment. Legal texts are often characterized by their precision, specificity, and reliance on established precedents. An LLM generating hallucinatory legal advice or misinterpreting compliance requirements can lead to flawed legal interpretations, resulting in incorrect compliance reports or misguided legal actions. Such errors compromise the integrity of legal processes and expose organizations to significant risks, including regulatory penalties, legal disputes, and reputational damage.

The detection of hallucinations in LLM outputs is, therefore, a critical area of research, with direct implications for the reliability of LLM-driven legal systems. Several detection methodologies have been proposed, each with its own strengths and limitations. Fact overlap-based methods, for instance, are effective in identifying hallucinations by comparing the generated output against a reference source. However, these methods may struggle with the nuanced language of legal texts, where slight variations in wording can have significant implications.

Entailment-based approaches offer a more sophisticated mechanism by evaluating whether the generated output logically follows from the input text. This method

is particularly relevant in legal contexts, where the logical coherence of arguments is crucial. However, the complexity of legal reasoning, which often involves multiple layers of interpretation and context, can pose challenges for entailment-based systems, leading to false positives or negatives in hallucination detection.

Classifier-based methods, which rely on pre-trained models to identify hallucinations, offer scalability and adaptability across different tasks. These models can be fine-tuned on legal datasets to enhance their detection capabilities. Nevertheless, detection quality is heavily dependent on the training data, and in domains like Legal NLP, where annotated datasets are scarce, the effectiveness of classifier-based methods may be limited.

Recent advancements in question-answering and retrieval-based methods provide promising avenues for hallucination detection, particularly in legal applications. By cross-referencing generated outputs with external knowledge bases or legal repositories, these methods can verify the accuracy of LLM outputs in real time. However, the dynamic and evolving nature of legal knowledge and jurisdictional differences necessitates continuous updates to these knowledge bases, which can be resource-intensive.

Uncertainty-based approaches, which measure the confidence level of LLM outputs, offer an additional layer of hallucination detection by flagging outputs with low confidence scores. In the legal domain, where the stakes are high, integrating uncertainty measures can serve as a valuable tool for legal practitioners, alerting them to potential inaccuracies that require human verification.

Prompting-based methods introduce innovative dimensions to hallucination detection. Prompting strategies, which involve crafting specific input prompts to elicit more accurate outputs, can be tailored to the legal domain by incorporating legal terminology and context.

The impact of hallucination detection on automatic legal compliance and software assessment is profound. Effective detection mechanisms enhance the reliability of LLM-driven systems and pave the way for more robust and trustworthy automated legal tools. By mitigating the risks associated with hallucinations, these methods contribute to developing legal AI systems that can be confidently relied upon for critical decision-making processes.

6.4 Limitations

The criteria applied in this review may have unintentionally excluded some relevant research, which could result in an incomplete or biased representation of the field of hallucination detection in LLMs. Moreover, given the rapid pace of advancements in LLM technology, findings may quickly become outdated as new methods and innovations emerge. The fast-evolving nature of this field necessitates ongoing updates to the review to incorporate the latest developments and provide a more comprehensive understanding of hallucination detection strategies.

This review has focused on initial strategies for detecting hallucinations in LLM outputs. However, it has yet to extensively cover other significant aspects, such as advanced techniques for managing and mitigating hallucinations or applying these methods across various domains and types of tasks. Chapter 8 of this thesis covers mitigating hallucinations.

The review covers research utilizing various methodologies; however, few works were found that attempt to combine multiple approaches to address hallucinations in LLMs comprehensively. This gap may be due to the early stage of research on hallucination detection in LLMs, where most studies focus on isolated techniques rather than integrative solutions. Alternatively, it could be a limitation of this review, as it may have yet to capture all possible interdisciplinary or hybrid approaches. The lack of a unified framework to tackle hallucinations across different stages of LLM deployment highlights an area for future research, emphasizing the need for

multi-faceted strategies that combine the strengths of various methodologies for more robust outcomes.

The conclusions drawn from this review are specific to hallucination detection and may not be broadly applicable to other areas where LLMs are involved. While some challenges identified here may be shared across LLM research, the review concentrates on detecting hallucinations. The lack of cross-validation of results from the presented works means that limitations in these studies may also influence this review's findings.

6.5 Conclusions

This chapter delves into the multifaceted challenge of hallucination detection in LLMs, a critical issue at the intersection of advanced natural language processing and real-world applications. A literature review of various detection methodologies, each offering unique advantages and facing specific challenges, was done. The review has highlighted the diversity and adaptability of current detection methods, from fact overlap-based methods that compare outputs to reference sources to entailment-based approaches that assess logical coherence and classifier-based techniques that leverage pre-trained models. Additionally, question-answering, retrieval-based, uncertainty-based, and prompting-based detection methods have shown promise in enhancing the accuracy of hallucination detection across different tasks.

The insights gained from this literature review provide a solid foundation for advancing the field of hallucination detection. Future research can contribute to developing more reliable, trustworthy, and legally compliant LLMs by addressing the outlined challenges and continuing to innovate detection methodologies. These advancements will improve the practical utility of LLMs across various domains and ensure that they meet the accuracy and reliability required in critical applications such as legal and regulatory compliance.

CHAPTER SEVEN

A Token Probability Method for Detecting Hallucinations in Large Language Model Outputs

The work detailed in Chapter Seven has been written mainly from the pending for publication: Quevedo, Ernesto, Jorge Yero, Rachel Koerner, Pablo Rivas, and Tomas Cerny "Detecting Hallucinations in Large Language Model Generation: A Token Probability Approach" arXiv preprint arXiv:2405.19648 (2024).

As stated at the beginning of this thesis, Chapter 6 already conducted a limited study on how current state-of-the-art methods perform in understanding legal documents related to software systems and has examined how top-performing chat-based Large Language Models (LLMs) like ChatGPT can understand workflows in complex software systems. Chapters 2, 4, and 6 clearly show that hallucinations in LLMs pose a significant problem with implications for performance and ethical considerations across all domains where LLMs are used, including the legal domain. Given the gravity of this issue, this will be the main topic for the remainder of the thesis, before the concluding chapter, to study this problem and provide contributions to detecting hallucinations in LLMs.

Detecting hallucinations in LLMs has significant implications for the automatic auditing of legal regulations in software systems. In legal audits, where the accuracy of the LLM's responses is essential, detecting and flagging hallucinations ensures that the conclusions drawn about a software system's compliance are based on reliable information. This contributes significantly to the robustness and credibility of automated auditing systems, reducing the risk of misinterpretation and erroneous compliance assessments.

This chapter explores a methodology for detecting hallucinations in LLMs, followed by the experiments and results obtained. It closes with a discussion of the limitations and a summary of everything presented.

7.1 Methodology

Two classifiers were implemented, Logistic Regression (LR) and a Simple Neural Network (SNN), utilizing four numerical features derived from token and vocabulary probabilities obtained during a forward pass through an LLM using the conditional generation approach Zhang et al. (2023). This section outlines a comprehensive methodology for detecting hallucinations in text generated by an LLM based on a given text.

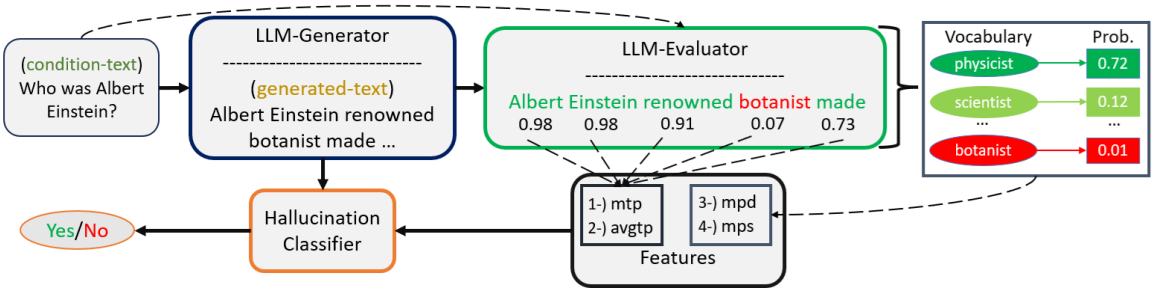

Figure 7.1. General Pipeline of the Proposed Methodology.

7.1.1 Problem Statement

Given a pair of texts *(condition-text, generated-text)* that represent the text used to condition the LLM to its generation. The objective is to detect if a given *generated-text* is a hallucination.

7.1.2 General Pipeline

Given a set of pairs of texts of the type *(condition-text, generated-text)* from an LLM (called the LLM-Generator (LLM_G)), four numerical features are extracted

based on the generated tokens and vocabulary tokens probabilities from another LLM that is called the LLM-Evaluator (LLM_E).[1]

Then, using these four features, two different classifiers are trained: a Logistic Regression (LR) and a Simple Neural Network (SNN). Finally, these classifiers were evaluated on a test set they did not see before. Figure 7.1 illustrates the process.

7.1.3 Features Description

This section provides more details on each feature extracted. Every feature is computed using token probabilities and the vocabulary probability distribution corresponding to each token on the *generated-text*. These can be formally defined as follows: (i) The token probability of each token of the Vocabulary of the LLM_E corresponding to t_i as $P_{LLM_{E_i}}(v_k) = (v_k|t_{i-1}, ..., t_1, c_m, ..., c_1; \theta)$ for every k. (ii) The token with the highest probability at position i in the *generated-text* according to LLM_E as the $v^* = \arg\max_k P_{LLM_{E_i}}(v_k)$. (iii) The token with the lowest probability at position i according to LLM_E as the $v^- = \arg\min_k P_{LLM_{E_i}}(v_k)$.

Next is a natural language description of the four features and, the mathematical definition:

- **Minimum Token Probability (mtp)**: Minimum of the probabilities that the LLM_E gives to the tokens on the *generated-text*.

- **Average Token Probability (avgtp)**: Average of the probabilities that the LLM_E gives to the tokens on the *generated-text*.

- **Maximum LLM_E Probability Deviation (mpd)**: Maximum from all the differences between the token with the highest probability according to LLM_E at position i and the assigned probability from LLM_E to t_i which is the token generated by LLM_G.

[1] Which could be the same as LLM_G.

- **Minimum LLM_E Probability Spread (mps)**: Maximum from all the differences between the token with the highest probability according to LLM_E at position i (v^*) and the token with the lowest probability according to LLM_E at position i (v^-).

Formally, these features can be defined as:

$$mtp = \min_i P_{LLM_E}(t_i) \quad avgtp = \frac{\sum_{i=1}^{n} P_{LLM_E}(t_i)}{n}$$

$$mpd = \max_{1 \leq i \leq n} (P_{LLM_E}(v^*) - P_{LLM_E}(t_i))$$

$$mps = \min_{1 \leq i \leq n} (P_{LLM_{E_i}}(v^*) - P_{LLM_{E_i}}(v^-))$$

These four numerical features are inspired by the mathematical investigation of the GPT model in Lee (2023), and recent results in Manakul et al. (2023); Su et al. (2024), suggesting there is a correlation between the minimum token probability on the generation, the average of the token probabilities, and the average and maximum entropy.

Lee (2023) proposes that a reliable indicator of hallucination during GPT model generation is the low probability of a token being generated. This is based on the assumption that forcing the model to generate such a low-probability token occurs when the difference between the token with the highest probability and all other tokens is less than a small constant δ. Here, *mps* is an estimator to avoid the cost of calculating differences across a large vocabulary and generated text.

Additionally, Azaria and Mitchell (2023b) trained a simple classifier called Statement Accuracy Prediction, based on Language Model Activations (SAPLMA) that outputs the probability that a statement is truthful, based on the hidden layer activations of the LLM as it reads or generates the statement. The authors showed that SAPLMA, which leverages the LLM's internal states, performs better than prompting the LLM to state explicitly whether a statement is true or false. Different from Azaria and Mitchell (2023b), Su et al. (2024) introduced MIND, an unsupervised training

framework that leverages the internal states of LLMs for hallucination detection requiring no manual annotations.

Diverging from previous studies Azaria and Mitchell (2023b); Lee (2023); Manakul et al. (2023); Su et al. (2024), this chapter presents an empirical approach rather than a theoretical one. Unlike Manakul et al. (2023), Self-Consistency is not used, and the method is not based on zero-shot or few-shot learning. Instead, a supervised learning approach is adopted similar to Azaria and Mitchell (2023b), focusing on four features derived from token and vocabulary probabilities rather than contextual embeddings or hidden layers. A simpler logistic regression classifier alongside a simple dense neural network will be used to evaluate this approach.

Additionally, the method considers using different LLM-Evaluators (LLM_E) alongside the generating model (LLM_G). Varying LLM-E models, differing in architecture, size, parameters, context length, and training data, might provide reliable but quantitatively distinct results compared to LLM_G's probabilities. To account for this, a new numerical feature is introduced, *mpd* (Maximum LLM_E Probability Deviation), which measures the deviation between the maximum probability token in LLM_E's vocabulary and the token generated by LLM_G.

Using different LLMs as evaluators leverages the diversity in training data and linguistic patterns across various models. By analyzing probability distributions from multiple LLMs, detecting hallucinations in outputs from a given model, such as LLM_G may be possible. This approach addresses the potential limitations of individual models, which might miss specific patterns or hallucinations that other models, specialized in certain topics, could detect. Evaluations from multiple LLMs enhance the robustness and reduce biases inherent in the training data of any single model.

7.1.4 Feature Extraction

The LLM_E models suitable for the Conditional Generation Task are used to extract these features. In this case, the process involves forced decoding because the tokens of the *generated-text* may have been produced by a different LLM (LLM_G). Instead of allowing LLM_E to generate the answer token-by-token based solely on the *condition-text*, the tokens predicted by LLM_G at each step are provided. This method forces LLM_E to replicate the generation path of LLM_G, enabling us to extract token probabilities from LLM_E for that specific sequence. These probabilities are then used to compute the four numerical features previously described.

7.1.5 Models Specification

A Logistic Regression (LR) and a Simple Neural Network (SNN) are the classifiers employed. Both classifiers use the four numerical features extracted from the *(condition-text, generated-text)* pairs for their analysis. The LR was chosen for its simplicity, rapid training, and effectiveness in binary classification tasks. In contrast, the SNN was implemented to capture complex non-linear relationships in the data. The SNN architecture includes an input layer with four neurons, each representing one of the features, two hidden layers with 512 neurons, each using the ReLU activation function, and an output layer with a single neuron activated by a sigmoid function, which is appropriate for binary classification tasks.

7.2 Experimental Setup and Results

This section describes the details of the experimental setup, the dataset used and the results obtained.

7.2.1 Datasets

- **HaluEval**: Hallucination Evaluation for Large Language Models (HaluEval) benchmark is a collection of generated and human-annotated hallucinated samples for evaluating the performance of LLMs in recognizing hallucinations.

HaluEval includes 5,000 general user queries with ChatGPT responses and 30,000 task-specific examples (10,000 per task) from three tasks: question answering, knowledge-grounded dialogue, and text summarization Li et al. (2023).

- **HELM**: Hallucination detection Evaluation for multiple LLMs (HELM) benchmark is a list of 3582 sentences from randomly sampling 50,000 articles from WikiText-103 Merity et al. (2016) where the selected LLMs were tasked with prompt-based continuation writing. The resulting sentences were annotated as hallucination or not Su et al. (2024).

- **True-False**: Comprises 6,084 sentences divided into the topics of "Cities," "Inventions," "Chemical Elements," "Animals," "Companies," and "Scientific Facts." All sentences in each category are conformed of true statements and false statements Azaria and Mitchell (2023b). Unlike HaluEval, this dataset only has *generated-text* and does not include any *condition-text*.

7.2.2 LLM Evaluators Used

The LLMs selected as evaluators to study the impact of factors such as architecture, training method, and training data include *GPT-2*, its large version (`gpt2-large`) Radford et al. (2019); Bidirectional and Auto-Regressive Transformers (*BART*), its CNN-Large version (`bart-large-cnn`) Lewis et al. (2020); Longformer Encoder-Decoder (*LED*) Beltagy et al. (2020), with the version fine-tuned on the arXiv dataset (`led-large-16384-arxiv`). Also, four bigger LLMs were used like OPT-6.7B (*OPT*) Zhang et al. (2022), GPT-J-6.7B (*GPT-J*) Wang and Komatsuzaki (2021), LLaMA-2-Chat-7B (*LLC-7b*) Touvron et al. (2023) and Gemma-7b (*Gemma*) Team et al. (2024). The known transformers library was also used.[2] In most cases, the `Conditional Generation setup` was used. For *GPT-2*, the `GPT2LMHeadModel` setup

[2] `https://huggingface.co/`

was employed. Additionally, when forwarding inputs to these models with a pair of *(condition-text, generated-text)*, the challenge of context limitation came to light, which varied depending on the LLM. To address this issue, the *generated-text* was not truncated if possible. Instead, if truncation was necessary (with a truncation length of `truncate_len`), the excess from the *condition-text* was removed. If additional knowledge was included, the truncation between the knowledge and the *condition-text* was evenly split.

7.2.3 Training Process of the Classifiers

To train LR, the sklearn library was used [3] with the Limited-memory Broyden-Fletcher-Goldfarb-Shanno Algorithm solver Saputro and Widyaningsih (2017) and default parameters. The SNN was trained during 10^4 epochs. The Adam Kingma and Ba (2014) optimizer was used with learning rate of 10^{-3}. All experiments, including feature extraction, training, and evaluation of the classifiers, were conducted on an NVIDIA L40S GPU core with 48GB of memory. Given a dataset and a single LLM_E, training takes anywhere from 30 minutes (HELM) to 4.5 hours (HaluEval), depending on the size of the training data and the length of the *condition-text* and *generated-text*.

7.2.3.1 HaluEval. Both classifiers were trained for each of the tasks in the HaluEval benchmark. Each data point is split into two data points: *(condition-text, right-answer)* and *(condition-text, hallucinated-answer)*. Therefore, the datasets would be of 20,000 examples for each of the Question Answering (QA), Knowledge-Grounded Dialogue (KGD), and Summarization tasks where in each case half of the dataset is comprised of data points of the type *(condition-text, right-answer)* and the other half are of the type *(condition-text, hallucinated-answer)*. In the case of the General-User Queries, the dataset is already in that format, with each data point classified as a hallucination. Therefore, the dataset size is the same, which is 5,000.

[3] `https://pypi.org/project/sklearn/`

Then, with this adaptation of the HaluEval benchmark dataset when approaching a given task, 10% of the data points were sampled (half with the *right-answer* and the other with a *hallucinated-answer*) randomly. These 10% data points are used to train both classifiers, and test the model capabilities on the remaining 90% of the dataset for a given task.

7.2.3.2 HELM. In the HELM dataset Su et al. (2024) the sentences were separated into categories depending on which LLM generated them. Therefore, to evaluate the sentences generated by a given LLM, like `LlaMa-Chat-7b` (LLC-7b) all other sentences produced by other LLMs were used for training.

7.2.3.3 True-False. The same process as Azaria and Mitchell (2023b) was followed. In all other categories, a category like "animals" is picked for testing and training.

7.2.4 Results

7.2.4.1 HaluEval. Each classifier was trained on 10% of the data, using the remaining 90% for testing. The metrics selected for evaluation are Accuracy, F1 Score, and Precision-Recall Area Under Curve (PR_AUC). Table 7.1 presents the state-of-the-art results from HaluEval, as reported in the papers by Li et al. (2023) and Li et al. (2024), which introduced the HaluEval benchmark and discussed factuality hallucination in LLMs.

Table 7.2 displays the accuracy results on the test set for each task using various LLM_E models and Logistic Regression as the classifier. The results demonstrate that Logistic Regression performs strongly compared to previous approaches on 90% of the dataset. Finally, Table 7.3 provides the average results derived from three random sampling runs across different models and metrics for Summarization, QA, and KGD tasks.

Table 7.1. Results taken from Li et al. (2024) measured in Accuracy (%) on the Halu-Eval dataset.

Models	QA	KGD	Summ.	GUQ
ChatGPT	62.59	**72.40**	**58.53**	79.44
Claude 2	**69.78**	64.73	57.75	75.00
Claude	67.60	64.83	53.76	73.88
Davinci-003	49.65	68.37	48.07	80.40
Davinci-002	60.05	60.81	47.77	**80.42**
GPT-3	49.21	50.02	51.23	72.72
Llama-2-Ch	49.60	43.99	49.55	20.46
ChatGLM 6B	47.93	44.41	48.57	30.92
Falcon 7B	39.66	29.08	42.71	18.98
Vicuna 7B	60.34	46.35	45.62	19.48
Alpaca 7B	6.68	17.55	20.63	9.54

Table 7.2. Results for each LLM_E and task using the LR classifier and measure in accuracy on the test set.

Model	Summ.	QA	KGD	GUQ
GPT-2	0.66	0.77	**0.62**	0.77
BART	0.65	**0.94**	0.49	0.54
LED	0.55	0.87	**0.62**	0.52
OPT	0.73	0.75	0.61	0.80
GPT-J	**0.90**	0.75	0.61	**0.81**
LLC-7b	0.70	0.74	0.55	**0.81**
Gemma	0.52	0.73	0.53	0.80

The classifiers, trained on just 10% of the data, show strong performance on the test set for Summarization and Question Answering (QA) tasks, using *GPT-J* and *BART* respectively as LLM_E. The best results achieve over 98% accuracy and F_1 scores with the SNN classifier in both tasks and over 90% accuracy and 95% F_1 scores with the LR classifier in Summarization and QA. These results surpass those of previous methods on the same test set, indicating a significant improvement.

Although state-of-the-art performance was not achieved for the dialogue task, the results remain competitive. In the GUQ task, results for the SNN classifier across different LLM_E models showed overfitting to the negative class, with an accuracy of 81%, F_1 of 1%, and PR_{AUC} of 10%. This overfitting is likely due to data imbalance,

Table 7.3. Average results in the test set for each task in the HaluEval benchmark given the selected LLM_E. NC_A stands for accuracy without *condition-text* and K_A for accuracy including extra knowledge. Here, AUC is the PR_{AUC}.

	Summarization				**Question Answering**					**KGD**				
Models	Acc	F1	AUC	NC_A	Acc	F1	AUC	NC_A	K_A	Acc	F1	AUC	NC_A	K_A
GPT-2	0.82	0.78	0.89	0.90	0.82	0.82	0.86	0.78	0.88	0.62	0.60	0.70	0.64	0.62
BART	0.77	0.78	0.83	**0.99**	**0.95**	**0.95**	0.97	**0.96**	**0.94**	**0.66**	0.63	0.74	0.65	0.60
LED	0.97	0.76	0.81	0.97	0.88	0.88	0.91	0.86	0.87	0.62	0.61	0.70	0.62	0.60
OPT	**0.98**	**0.98**	**0.99**	**0.99**	0.79	0.78	0.85	0.76	0.79	**0.66**	**0.67**	**0.74**	**0.67**	0.61
GPT-J	**0.98**	**0.98**	**0.99**	**0.99**	0.77	0.78	0.84	0.73	0.83	**0.66**	**0.67**	**0.74**	**0.67**	**0.66**
LLC-7b	0.67	0.68	0.69	0.77	0.73	0.69	0.81	0.75	0.77	0.60	0.54	0.64	0.63	0.61
Gemma	0.51	0.51	0.52	0.57	0.76	0.73	0.82	0.79	0.71	0.58	0.58	0.66	0.62	0.55

with only about 20% of the examples being non-hallucinations. Attempts to address this with a balanced training set of 500 positive and 500 negative examples still resulted in suboptimal performance, with a maximum accuracy of 69% and an F_1 score of 0.23%.

In the QA and KGD tasks, incorporating knowledge in the *condition-text* generally improved accuracy, though the benefit could have been more consistent. Surprisingly, when LLM_E models provided probabilities for the *generated-text* alone, unusually high results in Summarization and QA tasks were observed, reaching up to 99% accuracy with *GPT-J* and 96% with *BART*. This anomaly was not due to data leakage or errors, as similar results did not occur in KGD and GUQ or with all LLM_E models. This may indicate a probabilistic pattern in the dataset generation for Summarization and QA. Therefore, methods that learn from the probabilities may achieve high results due to this pattern, leading to incorrect interpretations of the method's generalization across all cases. However, the HaluEval dataset can still be valuable for evaluating unsupervised methods, as these do not rely on learning from the sequence probabilities.

Although choosing specific LLM_E models led to notable improvements in certain tasks, other LLM_E models still delivered competitive outcomes, with some even exceeding state-of-the-art benchmarks. For instance, when paired with the SNN

classifier, *OPT* achieved a 79% F_1 score in the QA task, and *GPT-2* reached an 82% F_1 score.

Table 7.4. Results of the approach and previous methods in the HELM benchmark measured in PR_{AUC}.

Baselines	Falcon	GPT-J	LLB -7B	LLC -13B	LLC -7B	OPT
PE-max	0.648	0.750	0.685	0.444	0.493	0.726
SCG-NLI	0.685	0.868	0.764	0.583	0.657	0.810
SAPLMA	0.513	0.699	0.578	0.305	0.407	0.621
MIND	**0.790**	**0.877**	**0.788**	0.604	**0.676**	**0.884**
Ours						
GPT-2	0.683	0.847	0.759	0.618	0.616	0.850
BART	0.710	0.828	0.695	0.569	0.568	0.825
LED	0.683	0.809	0.722	0.527	0.548	0.829
OPT	0.719	0.839	0.773	0.634	0.637	0.864
GPT-J	0.702	0.808	0.751	**0.642**	0.588	0.834
LLC-7b	0.727	0.855	0.785	0.563	0.644	0.842
Gemma	0.738	0.850	0.786	0.601	0.651	0.843

7.2.4.2 HELM. Table 7.4 presents the results of the method applied on the HELM benchmark Su et al. (2024). Overall, the approach did not outperform MIND Su et al. (2024) except for LLC-13B sentences. Nevertheless, despite using only four features, the method achieves better results than other approaches such as SAPLMA Azaria and Mitchell (2023b) and, in some instances, SelfCheckGPT with Natural Language Inference (SCG-NLI) Manakul et al. (2023) and other methods reported in Su et al. (2024).

Unlike MIND, which uses unsupervised training data without annotations, the method is trained with the annotated data provided in the HELM benchmark.

7.2.5 HELM results without condition-text

In this section, Table 7.5 displays the results of the method on the HELM benchmark using only the token probabilities from the *generated-text*. The impact on

Table 7.5. Results of the approach and previous methods in the HELM benchmark measured in PR_{AUC} without *condition-text*.

Baselines	**Falcon**	**GPT-J**	**LLB-7B**	**LLC-13B**	**LLC-7B**	**OPT**
PE-max	0.6479	0.7497	0.6851	0.4439	0.4931	0.7263
SCG-NLI	0.6846	0.8680	0.7644	0.5834	0.6565	0.8103
SAPLMA	0.5128	0.6987	0.5777	0.3047	0.4066	0.6212
MIND	**0.7895**	**0.8774**	**0.7876**	0.6043	**0.6755**	**0.8835**
Ours						
GPT-2	0.7110	0.8097	0.7384	0.5194	0.6085	0.7994
BART	0.6853	0.8129	0.7139	0.5624	0.5686	0.8258
LED	0.7017	0.8424	0.6931	0.5194	0.5494	0.8204
OPT	0.7163	0.7748	0.6695	0.5608	0.6751	0.7773
GPT-J	0.7051	0.7873	0.6984	**0.6121**	0.5856	0.7989
LLC-7b	0.6968	0.8403	0.7423	0.5464	0.6370	0.8395
Gemma	0.6872	0.8256	0.7319	0.5788	0.6428	0.8265

performance when comparing these results to those in Table 7.4 can be seen. This comparison shows the importance of the *condition-text*. Further, it suggests that the anomaly observed in the HaluEval dataset for Summarization and QA tasks might be due to a probabilistic pattern in the *generated-text*.

7.2.5.1 True-False. The results achieved with any of the selected LLMs on this dataset fell short of the baseline set by Azaria and Mitchell (2023b) and were considerably lower than their approach. This indicates a significant limitation in the methodology and highlights the need to utilize hidden layers as features. Future methods should be evaluated on this challenging dataset to address this gap.

Table 7.6 presents the specific numerical results of the method on the True-False dataset. Despite demonstrating competitive performance on many HaluEval and HELM tasks using only four features, the approach performed poorly on this dataset. Therefore, future evaluations of hallucination using supervised methods could be conducted on datasets like this one or similar datasets where rapid methods like

Table 7.6. Results of the approach and previous methods in the True-False dataset measured in accuracy. The SALPMA results shown are using the 16th hidden layer with LLC-7b.

Model	**Cities**	**Invent.**	**Elem.**	**Anim.**	**Comp**	**Facts**
BERT-5-shot	0.5416	0.4799	0.5676	0.5643	0.5540	0.5148
SAPLMA	**0.9223**	**0.8938**	**0.6939**	**0.7774**	**0.8658**	**0.8254**
Ours						
GPT-2	0.4312	0.5353	0.4924	0.4920	0.5041	0.5049
BART	0.3846	0.5365	0.5172	0.4920	0.4550	0.4607
LED	0.4985	0.4954	0.5182	0.5357	0.5191	0.4787
OPT	0.4950	0.5479	0.5118	0.4573	0.5050	0.5392
GPT-J	0.5023	0.5308	0.5268	0.4871	0.5283	0.5408
LLC-7b	0.5182	0.5216	0.5267	0.5287	0.5208	0.5669
Gemma	0.5091	0.5205	0.4870	0.4692	0.4983	0.4705

the one developed in this chapter have been tested to avoid any potential probabilistic patterns.

7.2.5.2 Overall Conclusions from Results. Overall, the results show that the supervised learning approach, using only four features, performs competitively with current methods and even surpasses state-of-the-art techniques across various tasks and datasets.

In the HaluEval benchmark, which involves generations from ChatGPT using GPT-3.5, some of the top-performing models were *GPT-2* and *GPT-J*, closely related to the LLM-Generator tested. Interestingly, other models not based on the LLM-Generator, such as *OPT*, *BART*, and *LED*, achieved comparable or even superior results in some tasks. Notably, smaller models like *BART* outperformed all other LLM evaluators in the QA task, indicating that model size alone does not determine performance, but rather training data and architectural differences play a significant role.

Finally, the True-False dataset highlighted the weaknesses of the method, revealing that the features used are insufficient for effectively detecting hallucinations in this type of data.

7.2.6 Feature Importance Analysis - Ablation

Experiments using individual numerical features were done to assess their impact on results. Table 7.7 displays how accuracy for three tasks in HaluEval and three LLM_E models varied depending on which features were included. In most cases, using all features yielded the best results. When evaluating each feature individually, it was found that *mtp* and *avgtp* were the most significant, particularly for Summarization and QA. However, in the KGD task, the newly introduced feature *mpd* proved more critical for larger models like *GPT-J* and *LLC-7b*.

Table 7.7. Feature importance based on accuracy for three tasks in the HaluEval benchmark given three LLM_E.

Features				Summarization			Question Answering			KGD		
mtp	avgtp	mpd	mps	GPT_J	BART	LLC_{7b}	GPT_J	BART	LLC_{7b}	GPT_J	BART	LLC_{7b}
✓	✓	✓	✓	**0.98**	0.77	**0.69**	**0.76**	**0.95**	**0.74**	**0.66**	0.65	0.60
✓				0.50	**0.79**	0.50	0.64	0.92	0.50	0.56	0.60	0.50
	✓			**0.98**	0.64	0.57	0.69	**0.95**	0.51	0.62	**0.66**	0.50
		✓		0.50	0.61	0.54	0.72	0.90	0.73	0.62	0.58	**0.61**
			✓	0.51	0.62	0.60	0.62	0.64	0.57	0.53	0.53	0.52

7.3 Discussion

The proposed method's competitive performance compared to current state-of-the-art techniques highlights its potential for improving the reliability of LLM-based systems in legal compliance audits. This approach can help ensure that automated audits yield accurate and trustworthy results by effectively identifying hallucination instances where LLMs produce incorrect or misleading information. This is crucial in legal contexts where the correctness of the information directly impacts compliance assessments and regulatory decisions.

The observation that different LLMs, including smaller models like BART, can outperform or match the performance of larger models in specific tasks highlights the importance of evaluating hallucination detection across diverse LLM architectures. This suggests that incorporating a range of LLMs, regardless of size, can enhance the robustness of detection methods. For legal audits, this means that the choice of LLM for generating and evaluating compliance-related information should be carefully considered to optimize performance. The same principle applies to any domain, not only legal.

The finding that using different LLMs as evaluators often yield similar or superior results compared to using the LLM-Generator itself indicates a potential advantage in employing a diverse set of evaluators. This approach can improve the accuracy and reliability of hallucination detection by leveraging the strengths of various models. It also suggests that employing multiple evaluators can provide a more comprehensive compliance assessment, reducing the risk of overlooking critical legal issues in a software system. Once again, the same principle applies to any domain, not only legal.

When applied to the True-False dataset, the method's weakness reveals an area for improvement. The lack of significance in certain features for detecting hallucinations in this dataset suggests that further refinement is needed to enhance the method's applicability across different data types. Addressing these limitations will make the approach more versatile and effective in real world auditing scenarios.

7.4 Limitations

One limitation concerns the numerical features and models chosen as LLM_E. While the current approach has proven effective for certain tasks, it may only partially capture the depth and complexity of some types of textual content. Specifically, the derived features may be less meaningful for tasks such as Knowledge-Grounded Dialogue (KGD), which involve nuanced context and real-time interactions.

Although the method showed superior performance in tasks like Summarization and Question Answering in HaluEval, it delivered competitive yet not leading results in KGD and General User Queries. This might suggest that the approach could be overly specialized or that task-specific feature engineering is required. The limitations may also stem from the inherent constraints of the LLMs used as LLM_E.

Additionally, due to the limitations on context length for some LLMs, the *condition-text* had to be shortened or supplementary knowledge, which might result in losing the crucial context needed for accurate token probability classification. Some LLMs require more context length to fully utilize the information without omitting details.

Another key limitation is that the effectiveness and results of the approach may be affected by the datasets' characteristics. If the datasets contain inherent biases or lack diversity, it could skew the model's performance. For instance, the patterns found in the HaluEval benchmark may make these four numerical features effective for detecting particular hallucinations. Nevertheless, this does not change the fact that advanced state-of-the-art methods have yet to demonstrate comparable performance under similar conditions.

Although the method showed competitive results in the HELM benchmark, it's noteworthy that, except for SALPMA, most other methods used unsupervised techniques. Since the approach relies on supervised learning, it depends on well-curated and annotated data, which presents a limitation. Additionally, the performance issues observed with the True-False dataset indicate that the approach might need further refinement with additional features to be effective in a broader range of scenarios.

Finally, this method is based on binary classification. In practical situations, hallucinations can be more complex and vary in severity, which the current approach may not fully address. Additionally, there is a lack of interpretability; while numerical features can provide some insight, they do not explain the specific erroneous or

fabricated information being introduced. This could be mitigated by using a more flexible evaluation framework, such as analyzing the Receiver Operating Characteristic (ROC) curve across varying positive-class thresholds. This would allow for a deeper examination of how accuracy, precision, and recall fluctuate with different thresholds, providing more nuanced insights into model performance. By exploring different threshold values, we could assess how the model's behavior changes and identify the optimal balance between precision and recall, ultimately improving the understanding of hallucination detection. A study focusing on ROC curves would also clarify how well the model generalizes across cases with different degrees of hallucinations, offering a more comprehensive evaluation of its robustness and effectiveness. In this chapter, we presented results from the Precision-Recall Curve AUC on the HaluEval dataset but did not investigate the behavior of this curve or the ROC curve on any of the datasets, which is recommended for future methods based on this type of approach.

7.5 Conclusions

This chapter presents a supervised learning approach for identifying hallucinations in conditional text generated by LLMs. By leveraging just four features derived from conditional token probabilities, this approach demonstrated competitive performance across various tasks and datasets, offering a promising direction for enhancing the reliability of LLM outputs. The results revealed that excluding the conditional text or prompt reduced performance, showing the critical importance of context in the evaluation process. Furthermore, extensive evaluation across three datasets provided valuable insights into the method's effectiveness, mainly when different LLMs were used as evaluators. This exploration emphasized the benefits of employing diverse models for evaluation purposes, with results indicating that alternative models can yield comparable or even superior outcomes compared to the LLM-Generator itself.

The implications of this chapter extend far beyond the technical field, particularly in domains that rely heavily on LLMs, such as medical, legal, educational,

and financial sectors. By enhancing the reliability of LLM outputs, this work contributes to the ethical use of these models in sensitive applications where accuracy and trustworthiness are vital. In the context of automatic legal compliance for software systems, this chapter attempts to develop reliable methods to detect hallucinations in LLM-generated text, thereby improving the integrity and dependability of Legal Natural Language Processing (Legal NLP) applications. As LLMs become increasingly integrated into various aspects of software development and regulatory compliance, ensuring the accuracy and reliability of their outputs will be crucial in navigating complex legal landscapes and meeting extended compliance requirements. This chapter's findings represent a significant step toward achieving these goals, paving the way for more robust and trustworthy LLM-based systems across multiple domains.

7.6 Credit

In this section, we will provide each author's contributions to the work presented in this chapter. For this, we will use the system CRediT (`https://www.elsevier.com/researcher/author/policies-and-guidelines/credit-author-statement`) from Springer to make it easier.

Ernesto Quevedo Caballero: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data Curation, Writing - Original Draft, Writing-Review Editing, Visualization, Project administration.

Jorge Yero: Software, Formal analysis, Writing-review and editing.

Rachel Koerner: Software, Validation, Formal analysis, Writing-review and editing.

Pablo Rivas: Visualization, Writing-review and editing, Supervision.

Tomas Cerny: Visualization, Writing-review and editing, Supervision.

CHAPTER EIGHT

LLMs Hallucinations Mitigation Survey

As discussed in the last two chapters, addressing hallucinations extends beyond merely detecting and correcting errors after they have occurred. In many applications, particularly those involving high-stakes decision-making or the processing of sensitive information, the consequences of hallucinations can be severe. For instance, in medical applications, a hallucinated output could lead to incorrect diagnoses or treatment recommendations, potentially endangering lives. In legal systems, hallucinations could result in flawed advice or ruling, leading to significant life and economic losses.

Moreover, the challenge of mitigating hallucinations is not just about preventing errors but also about fostering trust and reliability in LLM-based systems. As these models become more integrated into critical infrastructures and decision-making processes, ensuring their outputs are grounded in reality and free from fabrication is essential for their adoption and effective use. Users must be able to trust that the information provided by LLMs is accurate and transparent in its derivation, allowing for informed decision-making.

The stakes are particularly high in the domain of automatic legal compliance in software systems. Legal compliance involves interpreting, applying, and enforcing complex regulatory frameworks that govern software development, data privacy, cybersecurity, and more. Hallucinations in this context can have far reaching implications. A fabricated legal interpretation or an incorrect compliance recommendation could lead to violations of regulations, legal disputes, and significant financial penalties. Furthermore, the complexity and specificity of legal texts make them particularly vulnerable to hallucinations, as LLMs might need help to correctly interpret nuanced legal language or resolve ambiguities in regulatory documents.

Mitigating hallucinations in LLMs used for legal compliance is not only about ensuring the accuracy of the model's outputs but also about maintaining the integrity of the entire compliance process. Inaccuracies in this domain can undermine the credibility of the compliance system, leading to a loss of trust among stakeholders and potentially jeopardizing the organization's legal standing. Therefore, developing robust methods to mitigate hallucinations is crucial for successfully deploying LLMs in legal compliance workflows. This includes detection and correction mechanisms and strategies to prevent hallucinations from occurring in the first place, such as improving the quality of training data, refining prompt engineering techniques, and leveraging ensemble approaches to enhance the robustness of the model's outputs.

Therefore, although this thesis does not experiment with or present results on a specific method for hallucination mitigation, it still provides a literature review of current state-of-the-art methods for readers who wish to explore this challenging problem further. Additionally, this chapter concludes with an in-depth discussion about the problem of hallucinations in LLMs, based on the findings from Chapters 6 and 7.

8.1 Taxonomy of Hallucination Mitigation Methods in LLMs

This section provides a detailed overview of current methods to mitigate hallucinations in LLMs. Building on the insights discussed in Chapter 5, these methods are classified according to the root causes of hallucinations. Figure 8.1 illustrates the summary of the methods used to mitigate hallucinations in LLMs found in this chapter.

8.1.1 Mitigating Misinformation and Biases

The most straightforward way to reduce misinformation and biases is to gather high-quality factual data to prevent the introduction of false information and perform data cleansing to remove biases.

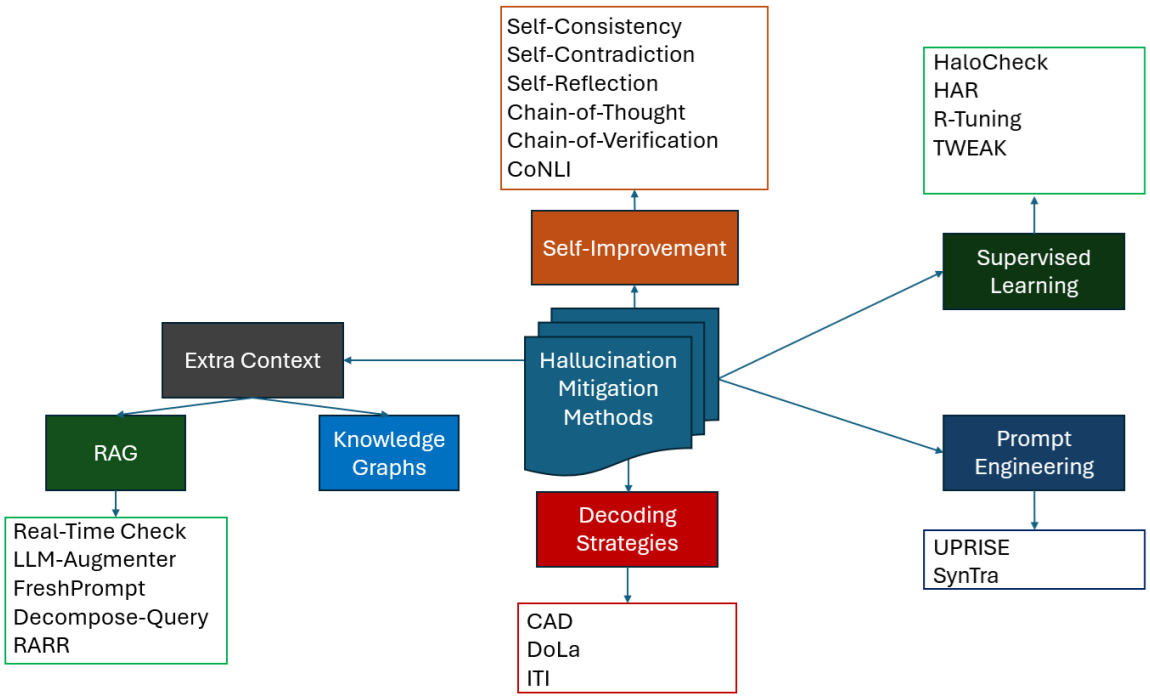

Figure 8.1. Summary of the review on Hallucination Mitigation in LLMs

Lin et al. (2022) proposes a benchmark designed to evaluate the accuracy of language models in providing truthful answers to questions. This benchmark includes 817 questions across 38 categories, such as health, law, finance, and politics. The questions were crafted to reflect answers some humans might give due to misconceptions or false beliefs. To achieve high performance, models must produce correct answers based on learned imitations of human text. Testing was conducted with GPT-3, GPT-Neo/J, GPT-2, and a T5-based model. The best-performing model was accurate in 58% of cases, whereas human accuracy was 94%. The models often produced incorrect answers that mirrored common misconceptions, with larger models generally needing more accuracy. This finding diverges from other Natural Language Processing (NLP) tasks where larger models typically perform better. The result is anticipated if inaccurate answers are learned from the training data. The authors suggest that improving truthfulness may require more than simply scaling up models and might benefit from fine-tuning with training objectives beyond merely imitating web text.

The research of Ladhak et al. (2023) investigates how sociocultural and other biases identified in pre-trained LLMs transfer to downstream tasks, explicitly focusing on name nationality bias. Their study traces this bias from the pre-training phase to its effects on summarization tasks, examining various summarization models. They demonstrate that these biases can lead to hallucinations, resulting in factually incorrect summaries. The study also reveals that the extent to which biases propagate depends on the summarization model used: more abstractive models tend to allow biases to transfer more directly as incorrect information. Additionally, the authors analyze how adjustments to adaptation methods and fine-tuning datasets impact name nationality biases. They find that while such modifications can reduce the overall occurrence of hallucinations, they do not alter the types of biases that emerge.

The work of Viswanath and Zhang (2023) presents a detailed quantitative evaluation of various biases related to race, gender, ethnicity, and age in widely-used pre-trained language models like BERT and GPT-2. They discuss how these biases, which models inherit from their training data, can be identified and measured using mathematical tools. Additionally, they introduce a toolkit designed to facilitate this process. The toolkit offers plug-and-play interfaces that connect these mathematical tools with popular language models, allowing users to evaluate biases in both existing and custom models. It also provides options for applying current debiasing techniques to mitigate these biases.

8.1.2 Retrieval Augmented Generation (RAG)

Retrieval Augmented Generation (RAG) is a hybrid approach designed to mitigate the problem of hallucinations by combining the generative capabilities of LLMs with a retrieval-based mechanism that pulls in relevant information from external sources. This approach enhances the factual accuracy of generated text by grounding it in real, retrievable data Gao et al. (2023).

RAG operates through two main components: the retriever and the generator. The retriever searches for and retrieves relevant information from a knowledge base or external documents, using embeddings or other similarity metrics to identify the most contextually appropriate passages. Once relevant information is retrieved, it is fused with the original query and fed into the generator, which is the LLM itself. The generator uses this external information as context, grounds its responses, and reduces the likelihood of hallucination. The process begins with a user query, which the retriever processes to find related data. This data is then integrated into the input for the LLM, which generates the final output, which is now anchored in verified information. This approach ensures that the output is coherent and factually accurate, thereby addressing one of the significant limitations of LLMs Gao et al. (2023).

Next, some of the most relevant works that use this methodology are summarized.

Kang et al. (2023) introduce a new method called Real-time Verification and Rectification (Ever). Rather than correcting hallucinations after the text has been fully generated, Ever employs a real-time, incremental approach to both generating text and rectifying hallucinations as they emerge. This strategy aims to identify and correct inaccuracies throughout the text generation process. Compared to existing retrieval-based and non-retrieval-based methods, Ever significantly enhances the reliability and factual accuracy of the generated content across various tasks, including short-form question answering, biography creation, and multi-hop reasoning.

The research of Peng et al. (2023) proposes a system called LLM-Augmenter. This system enhances a black-box LLM by integrating modular, plug-and-play components. These modules enable the LLM to generate responses grounded in external knowledge sources, such as task-specific databases. Additionally, LLM-Augmenter iteratively refines the LLM's prompts, using feedback from utility functions like the factuality score of the generated responses to improve accuracy. The authors

empirically validate the effectiveness of LLM-Augmenter in two use cases: task-oriented dialogue and open-domain question answering. In both scenarios, the system significantly reduces hallucinations in ChatGPT-generated responses. The framework also integrates Reinforcement Learning (RL) techniques, optimized using policy gradient, to further improve model performance.

The work of Vu et al. (2023) introduces FreshQA. This dynamic question-answering benchmark includes a wide range of question types, such as those requiring knowledge of rapidly changing information and those with false premises that need to be corrected. The study benchmarks various closed and open-source LLMs using a two-mode evaluation procedure designed to assess both the correctness of responses and the presence of hallucinations. Through human evaluations involving over 50,000 judgments, the study highlights the significant limitations of these models, particularly their difficulties with questions that involve fast-changing knowledge and false premises, regardless of model size. In response to these findings, the study introduces FreshPrompt. This straightforward few-shot prompting technique significantly enhances LLM performance on the FreshQA benchmark by incorporating up-to-date information retrieved from a search engine into the prompt. The experiments demonstrate that FreshPrompt outperforms other search engine-augmented prompting methods, such as Self-Ask Press et al. (2022) and commercial systems. Further analysis of FreshPrompt reveals that the quantity of retrieved evidence and the sequence in which it is presented play a crucial role in improving the accuracy of LLM-generated answers. Additionally, instructing the LLM to provide concise and direct responses reduces hallucinations compared to prompts encouraging more verbose answers.

The work of Varshney et al. (2023) their approach begins by identifying potential hallucination candidates using the model's logit output values. These candidates are then validated for accuracy through a specialized procedure, and any detected hallucinations are mitigated before the generation process continues.

Extensive experiments with GPT-3.5 (text-davinci-003) on the article generation task demonstrate the proposed techniques' effectiveness. The detection method achieves a recall rate of approximately 88%, and the mitigation technique successfully addresses 57.6% of the correctly identified hallucinations. Notably, the mitigation process does not introduce new hallucinations, even in cases of false positives (incorrectly detected hallucinations). The combined approach of active detection and mitigation reduces the hallucination rate in the GPT-3.5 model from 47.5% to 14.5% on average. The study also highlights the broad applicability and effectiveness of the approach through additional evaluations, including its performance on different types of questions, such as multi-hop and false premise questions, and with another LLM from a different model family, Vicuna. Overall, this work significantly enhances the reliability and trustworthiness of LLMs, an essential advancement toward their widespread adoption in real-world applications.

Cao et al. (2023) presents the "Decompose-and-Query" (D&Q) framework, designed to address the challenge of hallucinations in LLMs during question-answering (QA) tasks. LLMs often struggle with multi-hop relations in complex questions or lack the required knowledge to provide accurate answers, leading to hallucinated responses. D&Q aims to mitigate this by guiding the model to decompose complex questions and query external knowledge in a structured, reliable manner. Experimental results highlight the effectiveness of D&Q. On the ChitChatQA dataset, D&Q performed on par with ChatGPT in 67% of cases, and on the HotPotQA Yang et al. (2018) question-only setting, it achieved an F1 score of 59.6%. The research introduces ChitChatQA, a new dataset, and demonstrates how D&Q enhances LLM performance by improving factual accuracy and reducing hallucinations in multi-hop QA tasks.

The research of Gao et al. (2023) introduces RARR (Retrofit Attribution using Research and Revision), a system designed to address the problem of unsupported or misleading content generated by language models (LMs). While LMs excel at tasks

such as question answering, reasoning, and dialogue, they often produce outputs that need more attribution to external evidence, making it difficult for users to assess the trustworthiness of the content. RARR aims to bridge this gap by offering a solution that automatically finds and integrates external attribution for generated text while preserving the original output as much as possible. RARR operates in two key phases: first, it retrieves attribution for the LM-generated output, and second, it post-edits the output to correct any unsupported content without significantly altering its style or structure. The system can be applied to various state-of-the-art LMs and performs well across different generation tasks, showing significant improvements in attribution while maintaining the original qualities of the text. Additionally, RARR requires minimal training examples and relies on standard web searches to retrieve evidence, making it adaptable and efficient. The study highlights the limitations of existing LMs in retaining factoid knowledge and demonstrates how RARR addresses this issue by refining outputs to be more attributable to external sources. While RARR is a promising solution for improving factual reliability, the paper acknowledges that there is still substantial room for improvement in integrating attribution mechanisms into LMs.

The work of Rawte et al. (2023) thoroughly analyzes hallucinations by categorizing them based on degree, orientation, and type and by proposing corresponding mitigation strategies. They define two primary orientations of hallucinations: (i) factual mirage (FM) and (ii) silver lining (SL). They further divide these orientations into intrinsic and extrinsic categories, with three levels of severity: (i) mild, (ii) moderate, and (iii) alarming. Additionally, the authors classify hallucinations into six types: (i) acronym ambiguity, (ii) numeric nuisance, (iii) generated golem, (iv) virtual voice, (v) geographic erratum, and (vi) time wrap. The authors introduce the HallucInation eLiciTation (HILT) dataset, which comprises 75,000 samples generated by 15 contemporary LLMs and human annotations for the

identified categories to support their analysis. Furthermore, the authors propose the Hallucination Vulnerability Index (HVI), a method for quantifying and ranking LLMs based on their susceptibility to generating hallucinations. HVI could be a valuable tool for the broader NLP community and play a role in AI policy-making.

Lewis et al. (2020) explore integrating pre-trained models with a differentiable access mechanism to explicit non-parametric memory, which has been primarily investigated for extractive downstream tasks. The authors propose a general-purpose fine-tuning approach for Retrieval Augmented Generation (RAG) models, combining pre-trained parametric and non-parametric memory for language generation. In their approach, the parametric memory consists of a pre-trained seq2seq model. In contrast, non-parametric memory is a dense vector index of Wikipedia that can be accessed using a pre-trained neural retriever. The authors compare two RAG formulations: one that conditions the same retrieved passages throughout the generated sequence and another that utilizes different passages for each token. They fine-tune and evaluate these models across a broad range of knowledge-intensive NLP tasks. Their results set a new state-of-the-art on three open-domain QA tasks, surpassing both parametric seq2seq models and task-specific retrieve-and-extract architectures. The RAG models produce more specific, diverse, and factual language for language generation tasks compared to a leading parametric-only seq2seq baseline.

8.1.3 Self-Improvement through reasoning and feedback

Self-improvement through reasoning and feedback methods for hallucination mitigation in LLMs involves enabling models to refine their outputs by leveraging internal reasoning processes and external feedback. These techniques encourage LLMs to evaluate their responses critically, incorporate feedback to correct errors and adapt over time. An advantage of this approach is its capacity for continuous learning and refinement, potentially leading to more accurate and reliable outputs as the model evolves. However, challenges include the risk of the model overfitting to specific

feedback and potentially reinforcing incorrect patterns if the feedback is not well-calibrated. Additionally, the effectiveness of reasoning and feedback methods depends on the quality and consistency of the feedback provided, which can be variable and may require careful management to avoid unintended consequences.

The research of Mündler et al. (2023) notes that LLMs are prone to generating text with hallucinated content, a key issue being self-contradiction, where the model produces two contradictory statements within the same context. In this study, the authors extensively investigated self-contradiction across various instruction-tuned LMs, focusing on evaluation, detection, and mitigation. The primary evaluation involves open-domain text generation, with the approach also applicable to shorter question-answering tasks. The analysis highlights the significant occurrence of self-contradictions, such as 17.7% of all ChatGPT's sentences. The authors introduce a novel prompting-based framework that effectively detects and mitigates self-contradictions. Their detection method achieves high accuracy, for instance, with an F1 score of around 80% when applied to ChatGPT. The proposed mitigation algorithm iteratively refines the generated text to eliminate contradictory content while maintaining its fluency and informativeness. Crucially, the entire framework applies to black-box LMs and does not require the retrieval of external knowledge, offering a complementary approach to retrieval-based methods, particularly since a significant portion of self-contradictions (e.g., 35.2% for ChatGPT) cannot be verified using online text.

The work of Ji et al. (2023) highlight the potential of LLMs for generative and knowledge-intensive tasks, including question-answering (QA). Despite this promise, practical deployment faces significant challenges, particularly the issue of "hallucination," where models produce information that sounds plausible but is inaccurate or nonsensical. This problem is especially critical in the medical domain due to the specialized professional concepts and the potential social risks. In this

work, the authors analyze the phenomenon of hallucination in medical generative QA systems using widely adopted LLMs and datasets. Their investigation focuses on identifying and understanding common problematic answers, with a particular emphasis on hallucination. The authors introduce an interactive self-reflection methodology integrating knowledge acquisition and answer generation to address this challenge. Through this iterative feedback process, their approach steadily improves the factuality, consistency, and entailment of the generated answers. The method produces accurate responses by leveraging LLMs' interactivity and multitasking capabilities. Experimental results from both automatic and human evaluations demonstrate the superiority of this approach in reducing hallucinations compared to baseline methods.

Yan et al. (2024) emphasize the importance of comparative reasoning in text preference prediction, noting that LLMs often show inconsistencies. Although techniques like Chain-of-Thought have improved accuracy in various settings, they need help consistently identifying the similarities and differences in complex texts. In response, the authors introduce Structured Comparative (SC), a prompting approach designed to predict text preferences by generating structured intermediate comparisons. SC begins by proposing comparison aspects and then generates textual comparisons for each aspect. Consistent comparisons are selected using a pairwise consistency comparator, ensuring that each aspect's comparisons highlight the differences between texts, significantly reducing hallucination and improving consistency. The authors demonstrate that SC enables LLMs to achieve state-of-the-art performance in text preference prediction through comprehensive evaluations across various NLP tasks, including summarization, retrieval, and automatic rating.

The study of Liu et al. (2023) notes that these models' enormous scale and computational demands pose substantial challenges for their practical deployment in resource-constrained environments. Although techniques like chain-of-thought (CoT) distillation have shown promise in creating small language models (SLMs) from

LLMs, there is a concern that distilled SLMs might still inherit flawed reasoning and hallucinations from the original LLMs. To address these issues, the authors propose a twofold approach: First, they introduce a novel method for transferring the self-evaluation capability from LLMs to SLMs, aiming to reduce the negative effects of flawed reasoning and hallucinations. Second, they suggest distilling more comprehensive thinking by incorporating multiple distinct CoTs and self-evaluation outputs, ensuring a more thorough and robust knowledge transfer into SLMs. Experiments conducted on three NLP benchmarks: SVMAP Patel et al. (2021), CQA Talmor et al. (2019) and ANLI Nie et al. (2020); show that their method significantly enhances the performance of distilled SLMs, providing a new perspective on developing more effective and efficient SLMs for resource-constrained settings.

The research of Dhuliawala et al. (2023) investigates the potential for language models to reflect on their responses and correct their errors. To tackle this, they develop the Chain-of-Verification (CoVe) method. The CoVe method begins with the model drafting an initial response. The model then plans a set of verification questions to fact-check its draft. These questions are answered independently to prevent bias that might arise from the draft response or other answers. Finally, the model generates a revised, verified response based on this process. Experimental results show that CoVe effectively reduces hallucinations across various tasks, including list-based questions from Wikidata [1], closed-book MultiSpanQA Li et al. (2022), and long-form text generation.

Lei et al. (2023) propose a hierarchical framework to detect and mitigate ungrounded hallucinations. The framework employs Chain of Natural Language Inference (CoNLI) to detect hallucinations and utilizes post-editing to reduce them. The framework is designed as a plug-and-play solution, requiring no fine-tuning or domain-specific prompt engineering, making it easy to integrate into existing systems.

[1] `https://query.wikidata.org/`

The framework demonstrated state-of-the-art performance in hallucination detection through extensive experiments across various text-to-text datasets. Additionally, it significantly improved text quality through rewriting without compromising fluency or coherence. The approach incorporates sentence-level and entity-level judgments to detect hallucinations, offering interpretable outputs for hallucination reduction. This generalizable framework effectively improves the factual accuracy of LLM-generated content while preserving its overall quality.

8.1.4 Prompt Engineering

Prompt engineering methods for hallucination mitigation in LLMs involve designing and refining input prompts to guide the model toward producing more accurate and reliable outputs. By crafting prompts that provide clear context, constraints, or specific instructions, users can reduce the likelihood of generating misleading or incorrect information. The advantage of prompt engineering lies in its simplicity and flexibility; it allows users to adapt the model's behavior without altering the underlying architecture. However, the effectiveness of this approach can be limited by the model's inherent capabilities and the challenge of creating universally effective prompts. Additionally, over-reliance on prompt engineering might lead to inconsistent results if the prompts are not carefully tailored or if the model encounters novel scenarios not covered by the prompts Sahoo et al. (2024).

The research of Si et al. (2023) breaks down reliability into four key aspects that align with the established framework of machine learning safety: generalizability, social biases, calibration, and factuality. The authors' primary contribution is the development of simple and effective prompts designed to enhance GPT-3's reliability in these areas. Their approach focuses on: 1) improving generalization to out-of-distribution data, 2) addressing social biases by balancing demographic distribution and incorporating natural language instructions, 3) calibrating output probabilities to ensure more accurate predictions, and 4) updating the model's factual knowledge

and reasoning processes. With these tailored prompts, GPT-3 demonstrates greater reliability across all facets compared to smaller-scale supervised models. The authors publicly make their processed datasets, evaluation scripts, and model predictions available. Their systematic empirical study provides new insights into the reliability of prompting LLMs and offers practical strategies for practitioners to use LLMs like GPT-3 more reliably.

Cheng et al. (2023) present UPRISE, a method for tuning a lightweight and versatile retriever that automatically generates prompts for a given zero-shot task input. They demonstrate the universality of this approach through a cross-task and cross-model evaluation. Specifically, the retriever is tuned using diverse tasks and tested on unseen task types. The tuning is performed with a small, frozen LLM, GPT-Neo-2.7B, while the performance of the retriever is evaluated on much larger models, such as BLOOM-7.1B, OPT-66B, and GPT-3-175B. The results indicate that UPRISE effectively mitigates the hallucination problem, as evidenced by experiments with ChatGPT. This suggests that UPRISE can potentially enhance the performance of even the most advanced LLMs.

The work of Jones et al. (2023) introduces SynTra, a method that uses a synthetic task to reduce hallucination. SynTra begins by designing a synthetic task where hallucinations are easily elicited and measured. It then optimizes the LLM's system message through prefix-tuning on this synthetic task. This optimized system message is subsequently transferred to more complex, real-world tasks that are difficult to optimize. The authors demonstrate that SynTra effectively reduces hallucination in two 13B-parameter LLMs across three real-world abstractive summarization tasks, using only the synthetic retrieval task for supervision. They also find that focusing on optimizing the system message, rather than fine-tuning the entire model, is crucial, as fine-tuning the model on the synthetic task can unexpectedly increase hallucination.

SynTra illustrates that leveraging synthetic data can provide valuable flexibility to mitigate undesired behaviors in practical applications.

8.1.5 Decoding Strategies

Decoding strategies for hallucination mitigation in LLMs involve adjusting the algorithms used to generate text to minimize inaccuracies and misleading information. Techniques such as beam search, sampling, and temperature control guide the model's output towards more reliable responses. The advantage of decoding strategies is that they can enhance the quality and coherence of the generated text by controlling various aspects of the output process, potentially reducing the likelihood of hallucinations. However, these methods often involve trade-offs between creativity and accuracy, and fine-tuning the decoding parameters can take time and effort. Additionally, while decoding strategies can help reduce certain types of hallucinations, they may need to fully address the root causes of inaccuracies, particularly those stemming from the model's training data or inherent limitations Shi et al. (2024).

Shi et al. (2023) introduce context-aware decoding (CAD). This method employs a contrastive output distribution to enhance the difference between output probabilities when a model is used with and without context. Their experiments demonstrate that CAD, which does not require additional training, significantly improves the faithfulness of various LLMs families, including OPT, GPT, LLaMA, and FLAN-T5, on summarization tasks. For instance, CAD achieves a 14.3% gain in factuality metrics for LLaMA. Moreover, CAD proves particularly effective in overriding a model's prior knowledge when it contradicts the provided context, resulting in substantial improvements on tasks where resolving such knowledge conflicts is crucial.

The research of Chuang et al. (2023) develops a method called Decoding by Contrasting Layers (DoLa), which involves obtaining the next token distribution by contrasting the differences in logits projected from later versus earlier layers to the vocabulary space. This approach leverages the observation that factual knowledge

in LLMs is often localized to specific transformer layers. The results show that DoLa effectively enhances surface-level factual knowledge and reduces the generation of incorrect facts. The method consistently improves truthfulness across various tasks, including multiple-choice and open-ended generation tasks. For instance, it improves the performance of LLaMA models on TruthfulQA Lin et al. (2022) by 12-17 absolute percentage points, demonstrating its potential to make LLMs more reliable in generating truthful information.

The study of Li et al. (2024) introduces Inference-Time Intervention (ITI), a technique aimed at enhancing the "truthfulness" of LLMs. ITI works by adjusting model activations during inference through specific directions applied to a limited number of attention heads. This intervention notably improves the performance of LLaMA models on the TruthfulQA Lin et al. (2022) benchmark. For instance, ITI increases the truthfulness of an instruction finetuned LLaMA model, Alpaca, from 32.5% to 65.1%. The authors also identify a tradeoff between truthfulness and helpfulness and demonstrate how to balance this by tuning the strength of the intervention. ITI is minimally invasive and computationally efficient, requiring only a few hundred examples to identify truthful directions, unlike methods such as Reinforcement Learning with Human Feedback (RLHF), which demand extensive annotations. The findings suggest that LLMs may have an internal representation of truthfulness, even though they sometimes produce incorrect information.

8.1.6 Using Knowledge Graphs

Knowledge graph-based methods for hallucination mitigation in LLMs utilize structured representations of knowledge to enhance the accuracy and reliability of generated content. These methods provide contextual information and factual verification by integrating knowledge graph databases that represent relationships between entities and concepts into the model's decision-making process. The advantage

of using knowledge graphs is their ability to offer a rich, organized source of real-world information that can help correct or prevent erroneous outputs. However, challenges include the need for comprehensive and up-to-date knowledge graphs and the complexity of integrating these graphs into the model's inference process. Additionally, the effectiveness of this approach depends on the quality and coverage of the knowledge graph, which might only encompass some relevant domains or emerging information Pan et al. (2024).

Ji et al. (2023) present RHO ρ, which utilizes representations of linked entities and relation predicates from a knowledge graph (KG). Their approach includes two key strategies: (1) local knowledge grounding, which combines textual embeddings with corresponding KG embeddings, and (2) global knowledge grounding, which enhances RHO with multi-hop reasoning abilities through an attention mechanism. Additionally, they introduce a response re-ranking technique that leverages walks over KG sub-graphs to improve conversational reasoning. Experimental results on the OpenDialKG dataset Moon et al. (2019) demonstrate that their approach significantly outperforms state-of-the-art methods, particularly in reducing hallucinations. According to automatic and human evaluations, their method achieves a 17.54% improvement in FeQA Durmus et al. (2020).

The research of Bayat et al. (2023) introduces FLEEK, a prototype tool designed to automatically detect factual errors in textual information generated by LLMs or humans. Recognizing the challenge of manually verifying factual accuracy, FLEEK extracts factual claims from text, gathers evidence from external knowledge sources, evaluates each claim's factuality, and suggests revisions for any identified errors based on the collected evidence. The tool aims to reduce the labor involved in error detection and correction. Initial empirical evaluations of FLEEK show promising results, with F1 scores between 77-85% for detecting factual errors. Despite these encouraging results, the authors acknowledge several limitations. First, FLEEK relies

on initial responses from LLMs, which can introduce inaccuracies. Second, the current evaluation is conducted on small-scale, manually annotated datasets.

8.1.7 Faithfulness loss functions

This section covers research that introduced a faithfulness-based loss function Chrysostomou and Aletras (2021). The focus is ensuring that a model's outputs align closely with the input data or ground truth, minimizing errors, omissions, and distortions. By creating a metric to gauge how faithfully a model reflects its input, this approach aims to enhance the reliability of model outputs, particularly in high-stakes fields like legal compliance, where accuracy is critical.

Yoon et al. (2022) addresses the challenge of Video-grounded Dialogue (VGD), where the goal is to generate an answer to a question based on a video and the surrounding dialogue. Although recent advancements in multi-modal reasoning have improved answer generation, many dialogue systems still suffer from text hallucination where irrelevant text is copied from input sources without fully understanding the question. This issue arises because the models tend to learn superficial patterns, as answers in the dataset often contain words from the input. Consequently, VGD systems rely too much on copying text, hoping those words match the correct answers. To tackle this, the authors introduce the Text Hallucination Mitigating (THAM) framework, which uses a Text Hallucination Regularization (THR) loss based on an information-theoretic approach to measure hallucinations. When applied to current dialogue systems, THAM improves performance on VGD benchmarks like AVSD@DSTC7 and AVSD@DSTC8 Alamri et al. (2019), enhancing the system's interpretability.

The research of Qiu et al. (2023) introduces a new metric, mFACT, which assesses the faithfulness of non-English summaries by leveraging translation-based transfer from various English faithfulness metrics. They also propose a straightforward but effective method to reduce hallucinations in cross-lingual transfer by weighting the loss of each training example based on its faithfulness score. Through extensive

experiments in multiple languages, they show that mFACT is the best metric for detecting hallucinations. Additionally, their loss weighting method significantly improves performance and faithfulness, as confirmed by automatic and human evaluations, compared to strong baselines like MAD-X Pfeiffer et al. (2020).

8.1.8 Supervised Fine-tuning

Supervised Fine-Tuning (SFT) is a crucial process in developing LLMs, aimed at optimizing their performance for specific downstream tasks by leveraging labeled datasets. This technique involves adjusting the LLM's weights through gradient descent, guided by a task-specific loss function that quantifies the discrepancy between the model's predictions and the true labels. The effectiveness of SFT largely hinges on the quality of the training data, which directly impacts the model's ability to generate accurate and reliable outputs Xu et al. (2023); Touvron et al. (2023). By refining the model's parameters through this targeted approach, SFT enhances its adaptability, enabling it to perform more effectively on new or previously unseen tasks Jin et al. (2024); Chung et al. (2024); Wang et al. (2022).

Elaraby et al. (2023) focuses on evaluating and mitigating hallucinations in BLOOM 7B Le Scao et al. (2023), a notable example of these smaller, publicly available LLMs used for research and commercial purposes. The authors introduce HaloCheck, a lightweight, BlackBox, knowledge-free framework that quantifies the extent of hallucinations in such models. They also investigate methods like knowledge injection and teacher-student training to reduce hallucinations in low-parameter LLMs. They reduce reliance on expensive instructions from larger models by fine-tuning the models with domain-specific knowledge and leveraging more powerful models like GPT-4 to generate detailed question answers. The experimental results show significantly decreased hallucinations, especially in challenging domains for these models.

The research of Köksal et al. (2023) addresses the concept of attribution in LLMs, which is crucial for managing information sources and enhancing the factual accuracy of the models. Existing methods often rely on open-book question answering (QA) to improve attribution. Still, factual datasets may cause models to recall information from their pretraining data rather than truly improving attribution. In contrast, the authors suggest that counterfactual open-book QA datasets, where answers must be strictly based on the provided text, enhance attribution more effectively. To create these counterfactual datasets, they introduce hallucination-augmented recitations (HAR), a method that leverages hallucinations in LLMs. Using open book QA as a case study, they show that models fine-tuned with their counterfactual datasets improve text grounding and achieve up to an 8% increase in F1 score. Their counterfactual datasets outperform human-annotated factual datasets, even when using models and datasets that are four times smaller. The improvements hold across different model sizes and datasets, including multi-hop, biomedical, and adversarial QA tasks.

The study of Tian et al. (2023) fine-tuned LLMs to enhance factual accuracy without relying on human labeling, focusing on more open-ended generation tasks than previous work. The authors build on two key innovations in NLP: (1) recent methods for assessing factuality by comparing outputs to external knowledge bases or using a model's confidence scores, and (2) direct preference optimization, a technique that fine-tunes models based on preference rankings rather than supervised imitation. The authors use these automatically generated factuality preference rankings—created either from existing retrieval systems or a novel retrieval-free approach to significantly improve the factual accuracy of Llama-2 on held-out topics. Compared to Llama-2-chat, at the 7B scale, they achieve a 58% reduction in factual errors when generating biographies and a 40% reduction when answering medical questions, outperforming methods like RLHF and decoding strategies aimed at improving factuality.

Zhang et al. (2024) propose a novel approach called Refusal-Aware Instruction Tuning (R-Tuning). This method identifies the mismatch between the knowledge stored in the pre-trained parameters and the instruction-tuning data. Based on this, they create refusal-aware data that teaches LLMs to avoid answering questions beyond their parametric knowledge. Experiments show that R-Tuning effectively improves models' ability to accurately answer known questions and refuse when faced with unknown ones. Notably, this refusal ability generalizes well to out-of-domain datasets, functioning as a meta-skill applicable to other tasks. Moreover, their analysis reveals that learning uncertainty through R-Tuning enhances a model's calibration and ability to estimate uncertainty better than traditional uncertainty-based testing methods.

The work of Qiu et al. (2023) introduces TWEAK (Think While Effectively Articulating Knowledge), a decoding-only method that can be applied to any LLM generator without retraining. TWEAK treats each generated sequence and its future possibilities as hypotheses and ranks them based on how well they align with the input facts using a Hypothesis Verification Model (HVM). Initially, the authors demonstrate the method's effectiveness using a Natural Language Inference (NLI) model as the HVM, showing that it improves faithfulness with minimal impact on output quality. They then enhance the HVM by training it on a novel dataset created by the authors, called FATE (Fact-Aligned Textual Entailment), which pairs input facts with accurate and altered descriptions. Testing TWEAK on two different generators, the authors report that the best TWEAK variants increase faithfulness by an average of 2.24/7.17 points (FactKB) in in/out-of-distribution settings, with only a minor 0.14/0.32-point decrease in quality (BERTScore) Zhang et al. (2019).

8.2 Discussion

Since the thesis already provides a detailed literature review in Chapter 6 and discusses the developed methodology for hallucination detection in Chapter 7, this section will also include an overarching discussion that integrates the content of

Chapters 6, 7, and 8. The focus will be on hallucinations in LLMs, the reasons they occur, and finally, how to mitigate them.

8.2.1 Hallucinations in LLMs

The occurrence of hallucinations in Artificial Intelligence (AI) systems, particularly within LLMs, presents significant risks, mainly when applied to critical domains such as healthcare, finance, and public safety. In healthcare, for instance, a hallucinated response by an AI model that incorrectly identifies a malignant lesion as benign could have life-threatening consequences for a patient. Conversely, falsely identifying a benign lesion as malignant might lead to unnecessary and potentially harmful medical interventions. In the financial sector, the implications of hallucinated outputs could result in misguided investment decisions, potentially leading to severe economic losses.

Even in applications deemed less critical, the deceptive nature of hallucinations poses a substantial barrier to the broader adoption of AI. Suppose the model occasionally fails to extract an essential item or introduces a spurious one. In that case, it becomes virtually impossible for users to detect these errors without manually reviewing the entire transcript. This undermines the utility of AI in such applications, as the need for constant human oversight nullifies the supposed efficiency gains.

In the context of automatic legal compliance for software systems, the risks associated with hallucinations are further amplified. Legal compliance often requires adherence to regulations, where any deviation can result in significant legal and financial repercussions. Hallucinated outputs in this domain could lead to incorrect interpretations of legal requirements, potentially exposing organizations to legal liabilities and regulatory penalties. The challenge is compounded by the difficulty in effectively communicating the limitations of LLMs to end-users, who may need more expertise to assess the AI's outputs critically. As a result, errors introduced by hallucinations can propagate through an organization's decision-making processes,

leading to flawed decisions that appear entirely plausible due to the AI's seemingly coherent outputs.

Addressing the causes of hallucinations in LLMs requires a commitment to responsible AI development practices. This includes prioritizing transparency and accountability in AI systems, implementing robust testing and validation protocols before deployment, and establishing ongoing monitoring mechanisms to detect and mitigate hallucinations as they occur. Ethical considerations must be at the forefront of AI development, particularly in applications with significant consequences for public safety, financial stability, and legal compliance.

Moreover, it is crucial to recognize that current generative AI systems cannot guarantee the absence of hallucinations. This inherent unreliability requires developers, users, and stakeholders to remain vigilant, understanding that hallucinations are not anomalies but rather intrinsic to how these models function under the current paradigm. Unlike traditional software, where failures are typically categorized as bugs, hallucinations are a byproduct of the stochastic nature of generative models. Consequently, stakeholders must implement guardrails and contingency plans to manage the output of these systems, acknowledging that hallucinations, while insidious, are an expected feature rather than an exception.

This is also supported by the research of Xu et al. (2024), which formalizes the problem and demonstrates that eliminating hallucinations in LLMs is impossible. The authors define a formal world where hallucination is identified as inconsistencies between a computable LLM and a computable ground truth function. Using learning theory, they show that LLMs cannot learn all computable functions, meaning hallucinations are inevitable. Since this formal world is a simplified representation of the natural world, hallucinations are unavoidable in real-world LLMs.

8.2.2 Why Hallucinations Happen

After a deep dive into chapters 6 and 7 and this chapter on hallucinations in LLMs, let's discuss why they happen.

Various surface-level factors contribute to hallucinations, such as issues with data, modeling, and prompting. However, the fundamental reason behind all hallucinations in LLMs lies in the very nature of the current language modeling paradigm. This paradigm, by its very design, inherently produces hallucinations.

An essential contributor to hallucinations in LLMs is their inherent randomness, a byproduct of the generative process they follow. When these models generate text, they don't deterministically produce the same response each time. Instead, they rely on probabilities assigned to various word sequences based on patterns learned during training. This probabilistic nature means that even when presented with the same prompt, the model might generate slightly different outputs on different occasions.

Without this stochastic element, LLMs would produce repetitive and overly rigid text, limiting their ability to generate diverse or contextually nuanced responses. However, this randomness comes with a trade-off: while it allows for more varied and adaptive outputs, it also increases the risk of hallucinations.

In situations where the model has insufficient data or conflicting information, the randomness in selecting probable next words can lead to the generation of sentences that sound plausible but are not factually accurate. The model is essentially making an educated guess based on statistical patterns rather than a grounded understanding of reality.

Additionally, generative AI models, including LLMs, create outputs by identifying and reproducing statistical patterns in their training data. Instead of storing explicit factual information, LLMs encode relationships between words and phrases based on these patterns. This approach means that the models don't have

a concrete understanding of what is true or false; they produce text that sounds plausible.

This approach generally works because generating plausible text often results in something true, especially if the model is trained on predominantly accurate data. However, LLMs are trained on vast amounts of text from the Internet, including correct information and inaccuracies, biases, and fabrications. Consequently, while these models have learned to generate many accurate sentences by recognizing patterns associated with truth, they have also been exposed to numerous variations of sentences that are partially or entirely incorrect.

One of the main reasons hallucinations occur in LLMs is their need for more grounding in authoritative sources of knowledge. With a firm base in verified, factual information, these models find it easier to differentiate between truth and falsehood, generating hallucinated content. However, this issue goes beyond just the absence of factual grounding. Even if a model were trained exclusively on accurate data, which presumes the availability of such high-quality information, the statistical nature of language models still makes them prone to hallucinations.

Consider a scenario where a model has only encountered truthful sentences and has learned the relationships between the words in these sentences. If there are two nearly identical sentences, both factually correct but differing by just a few words like a date and a name, for instance, "Legal event A happened in year X" and "Legal event B happened in year Y" the model might generate a mixed-up sentence such as "Legal event B happened in year X." The probability of this error is only marginally lower than producing either of the original, correct sentences, which highlights the inherent risk of hallucination in these systems.

The underlying issue is that statistical language models assume that minor variations in the input (i.e., the sequence of words) will result in slight variations in the output (i.e., the probability of generating a particular sentence). Technically

speaking, these models assume a smooth distribution, which is essential because the amount of data they need to encode vastly exceeds the model's memory capacity (the number of parameters). As a result, these models must compress the training data, which inevitably leads to some loss of information.

Statistical language models assume that sentences similar to those in the training data are also plausible. They create a smooth representation of language, which works well as long as plausibility isn't confused with factual accuracy. The critical issue is that these models were not designed initially with factual accuracy. They were built for tasks like translation, where coherence and plausibility are the primary concerns. The problem arises when these models are repurposed for tasks that require factual accuracy, such as answering questions, where the distinction between plausible and factual becomes critical.

The issue is that facts are not smooth or variable, and a statement is either true or false. LLMs, however, need to be equipped to draw a clear line between factual and non-factual sentences. Their inherent design creates blurred boundaries, making it impossible to establish a definitive threshold where a sentence can be deemed false based on its perplexity value. Even if such a threshold could be determined, it would vary across different sentences.

You might wonder why using this smooth representation cannot be avoided. The reason is that generating new sentences that weren't explicitly in the training data requires the model to infer that some unseen sentences are also plausible. This inference necessitates certain assumptions, and the smoothness hypothesis is reasonable and computationally efficient. These models are trained using gradient descent, which depends on smoothness in the loss function. This approach works well if factual accuracy is not the primary concern. With this smooth, loose compression of the training data, the model would retain the ability to generate novel sentences altogether.

This is why the current generative AI paradigm is inherently prone to hallucinations, regardless of how good the training data is or how sophisticated the training methods and safeguards are. The statistical language modeling approach generates plausible-sounding sentences by recombining words that have appeared together in similar contexts in the training data. It lacks an inherent understanding of whether a sentence is true or false; it only recognizes patterns that resemble sentences in the training set.

Providing the model with vast amounts of high-quality data could reduce the likelihood of generating false sentences to an almost negligible level. Unfortunately, this isn't the case. Recent research Wei et al. (2024); Yu et al. (2024); Zhang et al. (2024) indicates that if a sentence can be generated at all, no matter how low its initial probability, there exists a prompt that can elicit that sentence with near certainty. This means that even with the best safeguards, the system remains vulnerable to manipulation, and users can never be entirely confident that it won't be "jailbroken" Wei et al. (2024); Yu et al. (2024) by a malicious actor.

8.2.3 Hallucination Mitigation

Hallucinations in LLMs remain an inherent challenge due to their reliance on statistical patterns rather than a true understanding of factuality. Despite this, several strategies have been developed to mitigate the impact of these inaccuracies in practice.

One prominent approach is the incorporation of external knowledge bases and fact-checking systems. By grounding LLMs in authoritative sources, the risk of generating inaccurate or fabricated outputs is significantly reduced. This strategy involves integrating models with databases that provide verified information, allowing cross-referencing and generating content validation. However, while effective, this approach requires constant updates and maintenance of knowledge bases to ensure relevance and accuracy.

Another key strategy involves enhancing model architectures and training paradigms. Researchers are exploring ways to develop more robust models that are less susceptible to hallucinations. This includes increasing model complexity, integrating explicit reasoning capabilities, and using specialized training data and loss functions to minimize errors. These advancements aim to improve the model's ability to distinguish between accurate and fabricated information, yet they also introduce challenges related to model scalability and computational resources.

Transparency and interpretability of AI models are crucial for addressing hallucinations. Making models' decision-making processes more transparent makes diagnosing and correcting the sources of inaccuracies easier. Techniques such as interpretability tools and visualization methods can help understand how models generate outputs and identify potential areas where hallucinations might occur. However, balancing model complexity and interpretability remains a significant challenge.

Standardized benchmarks and test sets for hallucination assessment are essential for quantifying the prevalence and severity of hallucinations. These benchmarks enable researchers and developers to measure and compare different models' performance systematically. By establishing clear metrics and evaluation criteria, it becomes possible to identify which models perform better in reducing hallucinations and make informed decisions about their deployment. Nevertheless, creating comprehensive and representative benchmarks is an ongoing effort that requires continuous refinement.

Interdisciplinary collaboration is also critical for advancing the understanding and mitigation of hallucinations. Engaging with experts from scientific reasoning, legal argumentation, and other relevant disciplines fosters a broader perspective on the problem. This collaboration can lead to the development of more nuanced approaches and solutions that address the complexities of hallucinations in various contexts.

Looking ahead, the development of hybrid models that integrate multiple mitigation strategies presents a promising direction. These models combine various techniques, such as external knowledge integration, improved training paradigms, and advanced verification methods, to provide a more robust defense against hallucinations. Additionally, exploring unsupervised or weakly supervised learning techniques could enhance scalability and flexibility in mitigating inaccuracies.

Addressing hallucination mitigation strategies' ethical implications and societal effects is also crucial. It is a significant concern to ensure these strategies promote user trust and do not introduce new biases or inaccuracies. Researchers, practitioners, and ethicists must collaborate to establish standards and guidelines prioritizing user comprehension and authenticity.

8.2.4 Proposals

Based on these arguments, the following ideas in the scope of legal compliance are proposed.

First, grounding LLMs in authoritative legal texts, regulations, and case law can significantly enhance the accuracy of their outputs. By integrating comprehensive and up-to-date legal knowledge bases, LLMs can better align their responses with current laws and regulatory standards. This approach reduces the risk of generating incorrect legal advice and ensures the system complies with evolving legal frameworks. For example, an LLM designed to assess software compliance with GDPR could be integrated with a knowledge base containing the latest GDPR amendments and case law, improving its accuracy and relevance.

Second, in the legal domain, where accountability and transparency are vital, enhancing the interpretability of LLMs is crucial. By making the decision-making processes of these models more transparent, legal professionals can better understand and trust the model's outputs. This transparency facilitates the identification of errors or biases in the model's reasoning, which is essential for ensuring that automated

compliance assessments are accurate and justifiable. For example, a model that explains its compliance decisions can help legal teams understand the rationale behind its recommendations and verify their correctness.

Third, establishing standardized benchmarks for hallucination assessment in the context of legal compliance enables consistent evaluation of model performance. By developing test sets that include a variety of legal compliance scenarios, researchers and developers can measure how well LLMs perform in identifying compliance issues and generating accurate legal interpretations. These benchmarks can guide the development of more reliable compliance tools and help compare the effectiveness of different models in practical legal contexts.

Fourth, applying LLMs for legal compliance benefits greatly from interdisciplinary collaboration. Engaging with legal experts, compliance officers, and regulatory authorities ensures that the models are aligned with legal standards and practices. This collaboration can lead to developing more sophisticated models that address specific legal requirements and adapt to changes in the regulatory landscape. By working together, AI researchers and legal professionals can refine mitigation strategies and enhance the overall reliability of automated compliance systems.

8.3 Limitations

The criteria used in this chapter for the review may have inadvertently left out relevant research, potentially leading to an incomplete or biased view of hallucination detection in LLMs. Additionally, with the rapid advancements in LLM technology, the findings could quickly become outdated as new approaches and innovations arise. Due to the fast-paced nature of this field, regular updates to the review are necessary to incorporate emerging developments and provide a more thorough understanding of hallucination mitigation methods.

Another limitation lies in the scope of the models reviewed. Most research centers around widely known LLM architectures like GPT-based and LLaMa-based

models. However, many other models, notably smaller or less commercially visible, employ different mitigation techniques that have not been sufficiently covered here. As a result, this review may not fully capture the broader landscape of hallucination mitigation methods, especially those that may be more efficient or tailored for specific applications.

Additionally, this review does not address the ethical implications of hallucination mitigation strategies. While many methods discussed focus on improving factual accuracy and reducing bias, broader ethical concerns remain unexplored, such as transparency in automated decision-making and the unintended consequences of mitigation strategies, like transforming a non-factual statement into one that contains bias or toxicity. These issues are crucial, particularly as LLMs become more integrated into high-stakes decision-making processes in healthcare, finance, and legal compliance.

The conclusions drawn here are specific to hallucination detection and may not apply broadly to other areas where LLMs are used. While some challenges discussed may be relevant across LLM research, this review focuses on detecting hallucinations. Moreover, the lack of cross-validation of the studies in this review means that any limitations within those works may also impact the conclusions drawn here.

8.4 Conclusions

This thesis chapter provides a comprehensive review of current methods for mitigating hallucinations in LLMs and explores their relevance in the context of automatic legal compliance for software systems. Hallucinations in which LLMs generate plausible but factually incorrect outputs present a significant challenge, especially in high-stakes domains like legal compliance.

The review highlights strategies to address hallucinations, including integrating external knowledge bases, advancements in model architectures, improvements in transparency and interpretability, and the development of standardized benchmarks. Techniques such as mFACT, contextual frameworks, and self-contradiction detection

have been examined for their roles in identifying and mitigating hallucinations. Collectively, these methods represent the forefront of current efforts to enhance the reliability of LLMs.

However, the chapter also identifies inherent limitations and challenges within these approaches. For instance, while external knowledge bases are valuable, they require ongoing updates and maintenance to remain relevant. Similarly, while promising, advanced model architectures and training paradigms introduce complexities related to scalability and resource requirements. Although crucial, Transparency and interpretability efforts often need help balancing complexity with comprehensibility.

The chapter proposes several avenues for future research and development in light of these challenges. The potential for hybrid models that combine multiple mitigation strategies presents a promising direction, offering a more robust solution to the problem of hallucinations. Additionally, exploring unsupervised or weakly supervised learning techniques could enhance the scalability and flexibility of mitigation methods. Integrating interdisciplinary insights drawing from legal expertise, compliance knowledge, and advanced AI techniques can further refine and improve the effectiveness of hallucination mitigation in legal compliance.

Tailoring these mitigation strategies to the specific requirements of automatic legal compliance in software systems is essential. Accurate and reliable legal assessments are critical to ensure regulation compliance and avoid legal repercussions. By applying and adapting the reviewed techniques to this context, researchers and practitioners can develop more effective tools for automated compliance checks, thereby enhancing the trustworthiness and efficacy of these systems.

This chapter illuminates the current landscape of hallucination mitigation in LLMs and provides a structured overview of existing techniques and their limitations. The insights gained offer a foundation for future research and innovation in developing

more reliable and accurate language models, particularly in the domain of software systems' legal compliance. As the field progresses, continued exploration of hybrid approaches, scalability solutions, and interdisciplinary collaboration will be key to advancing the state of knowledge and practice in hallucination mitigation.

CHAPTER NINE

Conclusion

This concluding chapter summarizes the findings presented in this dissertation. The key elements are carefully highlighted as significant contributions and possible directions for future research.

In an era of rapid advancements in software development, ensuring compliance with legal regulations has become increasingly critical. The intricate separation between legal expertise and software engineering necessitates robust, automated methods for compliance and auditing. This thesis focused on leveraging Large Language Models (LLMs) to bridge this gap, particularly emphasizing legal document question answering and classification and understanding complex software systems based on microservices architectures. By evaluating the capabilities of LLMs in these domains, the dissertation contributes essential insights into their potential role as auditors of legal compliance in software systems. Although only a portion of the broader vision of full-spectrum compliance auditing is covered, this thesis lays the groundwork for a comprehensive approach by addressing the applications of LLMs in understanding legal documents and software systems separately.

A core component of this thesis is the evaluation of state-of-the-art LLMs such as BERT, ALBERT, RoBERTa, DistilBERT, and Legal-BERT on established datasets like SQuAD V2.0 and PolicyQA. This evaluation provides critical insights into how these models can be effectively employed in the legal domain. The introduction of the CSIAC-DoDIN V1.0 dataset, which centers on cybersecurity policies and organizational responsibilities, further strengthens the thesis by offering a new benchmark for future work in Legal Natural Language Processing (Legal NLP) tailored to cybersecurity. This dataset is a valuable resource for legal document analysis and establishes baselines

for evaluating LLM performance in legally sensitive contexts, laying the foundation for further advancements.

In the context of software systems, the thesis explores the application of LLMs in understanding complex architectures, particularly microservices. Utilizing Persistence Operation-aware Component Call Graphs (PO-CCG) as a contextual knowledge base, the dissertation demonstrates how structured data representations can enhance LLM performance in analyzing software systems. This approach highlights the limitations of relying solely on raw source code and emphasizes the critical role of prompt engineering in improving LLM performance. The findings suggest that integrating structured knowledge with LLMs can lead to more effective tools for compliance checking and software system analysis, thus advancing the reliability and accuracy of these models in practical applications.

This thesis addresses a significant challenge: hallucinations in LLMs are a critical barrier to their deployment in legal compliance tasks. Hallucinations, where LLMs generate incorrect or misleading information, pose a severe risk when applying these models to legal document analysis and software system understanding. The dissertation identifies and analyzes this problem and develops a novel method for detecting hallucinations in LLM outputs. The thesis challenges the assumption that larger models are inherently superior by rigorously evaluating this method across different LLM architectures, including large and smaller models like BART. Instead, it demonstrates that smaller, well tuned models can perform competitively, broadening the scope of applicable models for hallucination detection.

Moreover, the thesis introduces the innovative concept of using multiple LLMs as evaluators, demonstrating that a multi-model approach can yield more accurate and comprehensive results than relying on a single LLM. This insight paves the way for new approaches in compliance auditing, where leveraging the strengths of various models can enhance the system's overall reliability. By addressing the nuances of

hallucination detection, this dissertation contributes to the broader field of AI safety and robustness, offering valuable strategies for improving LLM performance in critical tasks related to legal compliance.

While primarily focusing on the application of LLMs in legal document question answering and software system analysis, the thesis also lays the groundwork for a more comprehensive approach to automatic legal compliance auditing. The insights gained provide a robust foundation for future work that integrates LLMs across the entire spectrum of compliance tasks, from understanding legal documents to evaluating complex software systems. Although the dissertation covers only a portion of this broader vision, the contributions are significant.

9.1 Publications

This research has produced notable outcomes, including publications highlighting its broad applications and theoretical contributions. Below, we'd like to present the publications that have resulted from this work and explore possible directions for future research. Next, the publications directly related to this dissertation are listed:

- "Legal Natural Language Processing from 2015-2022: A Comprehensive Systematic Mapping Study of Advances and Applications" IEEE Journal Impact Factor 3.9 Q1
- "Study of Question Answering on Legal Software Document using BERT based models" NAACL LXAI workshop 2022 (2nd Best paper Award)
- "Creation and Analysis of a Natural Language Understanding Dataset for DoD Cybersecurity Policies (CSIAC-DoDIN V1. 0)" CSCE 2023
- "Evaluating ChatGPT's Proficiency in Understanding and Answering Microservice Architecture Queries Using Source Code Insights" Springer Journal.

- "Detecting Hallucinations in Large Language Model Generation: A Token Probability Approach" CSCE 2024 (Best paper award)

Moreover, some works were published that are indirectly related to this dissertation but strongly related to the application of LLMs in the understanding of code in a software system:

- "BERT Goes to SQL School: Improving Automatic Grading of SQL Statements" CSCE 2023

- "A Dataset of Microservices-based Open-Source Projects" 21st International Conference on Mining Software Repositories (Core A)

9.2 Contributions

This thesis significantly advances the automatic legal compliance analysis field for software systems by exploring the application and limitations of state-of-the-art LLMs in this domain. Key contributions include the evaluation of LLMs like ChatGPT in interpreting complex software projects and legal texts, the introduction of the CSIAC-DoDIN V1.0 dataset for cybersecurity policy analysis, and the application of the PO-CCG intermediate representation for enhancing LLM performance in microservice architecture contexts. It also addresses critical challenges, such as hallucination detection and prompt engineering, offering new methodologies and insights that improve the accuracy and reliability of LLMs in general and potentially in their application for automated legal compliance audits, thereby reducing the risk of non-compliance and supporting more informed decision-making processes in software development.

9.2.1 Study of State-of-the-Art Approaches in Legal NLP for Software Development.

This thesis contributes significantly to the automatic legal compliance analysis of software systems by thoroughly investigating the performance and applicability of state-of-the-art Legal NLP models in the context of legal documents related to

software development. The study explores the capabilities of several prominent models, including BERT, LEGAL-BERT, ALBERT, DistilBERT, and RoBERTa, across the SQuAD V2.0 and PolicyQA datasets. One key finding is the underperformance of domain-specific models like LEGAL-BERT on the PolicyQA dataset compared to general-purpose models like ALBERT and BERT, revealing the importance of domain specificity and the need for models tailored to particular subdomains within Legal NLP. This insight is crucial for developing effective compliance tools, as it suggests that even within the broader legal domain, significant variations in subdomains, such as software development and privacy in applications, can markedly impact model performance.

To address the scarcity of resources in Legal NLP, particularly in software-based policies like cybersecurity, CSIAC-DoDIN V1.0 dataset was introduced. This new curated dataset, focused on cybersecurity policies, responsibilities, and procedures outlined by the DoD Deputy CIO for Cybersecurity, provides a vital resource for advancing Legal NLP applications in software compliance. Baseline performance was established on this dataset using classic transformer-based language models like BERT, RoBERTa, Legal-BERT, and PrivBERT across tasks such as multiclass classification and text co-occurrence. The dataset and the code for training and evaluation have been made openly accessible to support further research and the development of new models.

The experimental results from these studies have profound implications for automating legal compliance checks in software systems. The findings suggest that organizations must carefully select and fine-tune NLP models to ensure they align with the specific legal subdomains pertinent to their operations. This may involve training models on datasets to represent the legal issues inherent in software development more accurately and employing ensemble methods to enhance model performance. Furthermore, as legal frameworks around cybersecurity and software systems evolve,

there is a clear need for continuous model updates and retraining to maintain the effectiveness of compliance tools over time. This ongoing adaptation is essential for ensuring that automated compliance checks remain robust and reliable in the face of changing legal standards and software complexities.

9.2.2 Assessing ChatGPT's Ability to Comprehend and Respond to Microservice Architecture Questions Using Source Code Insights.

One of the key contributions of this thesis is developing a comprehensive set of system design questions aimed at assessing service and interaction perspectives within microservice architectures. These questions were crucial for benchmarking ChatGPT's effectiveness in generating accurate responses based on different representations, revealing that while ChatGPT could handle service and interaction views to some extent, its performance varied significantly depending on the representation used and the complexity of the questions, highlighting the importance of carefully designed queries that align with the strengths and limitations of the language model.

The dissertation also provides a comparative analysis to determine whether ChatGPT performs better when provided with raw source code or the more abstracted PO-CCG representation. The findings indicated that PO-CCG was generally more effective, particularly in scenarios involving multiple microservices where token limits were a concern. However, the raw source code proved more precise for questions requiring detailed implementation specifics. This shows the importance of selecting the appropriate level of abstraction in legal compliance audits, as the choice of knowledge representation can significantly impact the accuracy and reliability of the analysis.

Additionally, the thesis included a statistical evaluation of ChatGPT's performance across different categories of questions and emphasized the critical role of prompt engineering. The findings show that refining prompts by explicitly defining technical terms significantly improved the model's ability to generate correct and

contextually relevant answers. This aspect of the research demonstrates the necessity of precise and carefully crafted prompts in legal compliance, where ambiguity or misinterpretation could lead to significant errors.

9.2.3 A Token Probability Method for Detecting Hallucinations in Large Language Model Outputs.

This thesis develops a supervised learning approach for detecting hallucinations in LLMs outputs. The core of this contribution is the proposal of a token probability method that utilizes four carefully selected features to identify instances where LLMs generate incorrect or misleading information, also known as hallucinations. The research demonstrates this approach's efficacy by successfully applying two classifiers: Logistic Regression and a Simple Neural Network. These models were rigorously evaluated across three datasets, revealing their competitive performance compared to existing state-of-the-art methods.

The analysis of the proposed method extends beyond its application to a single model as the comparative effectiveness of using different LLMs as evaluators for hallucination detection is performed. The findings show the significance of model diversity, showing that smaller LLMs, such as BART, can often match or even outperform larger models like LLaMa-Chat-7b in specific tasks. This insight is precious for legal compliance audits, where the choice of LLM for generating and evaluating compliance-related information directly influences the accuracy and reliability of the assessment. By incorporating a range of LLMs, the method enhances the robustness of hallucination detection, thereby improving the overall trustworthiness of automated legal audits.

Furthermore, the thesis highlights the importance of using different LLMs as evaluators rather than relying solely on the LLM that generated the text. This approach improves detection accuracy and provides a more comprehensive compliance assessment by leveraging the unique strengths of various models. This methodology

has broader implications, suggesting that a multi-model approach could benefit other domains where LLMs are employed for critical decision-making tasks.

However, areas for improvement are also identified, particularly when applying the method to the True-False dataset, where certain features proved less significant in detecting hallucinations. This finding points to further refinement to enhance the method's applicability across diverse data types, making it more versatile and effective in real-world auditing scenarios. The proposed approach can be further optimized by addressing these limitations to support reliable and accurate legal compliance analysis in increasingly complex software systems.

9.3 Future Works

As this thesis has demonstrated, integrating LLMs into the domain of automatic legal compliance in software systems presents significant opportunities and challenges. The findings and limitations identified across the various chapters of this work open up multiple avenues for future research and development.

9.3.1 Training Domain-Specific LLMs for Legal Compliance

The findings from this thesis suggest that training domain-specific LLMs from scratch, using datasets exclusively related to software development and legal compliance, could yield superior performance compared to general-purpose models. Future research should explore the development of specialized models, such as an ALBERT variant or other high-performance model found in the literature that is trained solely on legal texts and compliance documents. This approach could lead to more accurate and reliable LLMs tailored to software systems' specific legal compliance needs, offering organizations a powerful tool for navigating the increasingly complex regulatory landscape.

9.3.2 Enhancing Context Management in LLMs

One of the key challenges identified in this thesis is the context length constraint imposed by LLMs, particularly in complex environments like large software system analysis or even extensive legal documents. Future research could focus on developing sophisticated information extraction and retrieval algorithms that selectively prioritize and inject the most relevant context for a given query. This approach would optimize the use of limited context space, potentially improving the precision of LLM responses in legal and regulatory settings. Moreover, investigating the potential of Chain-Of-thought prompting, Self-Reflection, and Self-Consistency techniques could further mitigate issues related to context management and prompt brittleness, leading to more robust and contextually aware LLM outputs.

Besides the context length limitation, another promising avenue for future work is the exploration of role-specific query handling by LLMs. Given that different stakeholders, such as project managers, architects, and developers, pose distinct queries relevant to their roles, it would be valuable to assess the ability of LLMs to tailor their responses accordingly. This could involve training models to recognize the context and intent behind a query based on the user's role, thereby enhancing the relevance and accuracy of the information provided. Such advancements could significantly improve the usability of LLMs in complex legal compliance environments, where precision and contextual relevance are essential.

9.3.3 Improving Detection of Hallucinations in LLMs

A critical area for future exploration involves refining the methods for detecting hallucinations in LLM-generated text, particularly in the context of legal documents and compliance-related tasks. While the current supervised learning approach, leveraging token probabilities, has shown promise, there is substantial potential in hybrid methods that combine In-Context Learning with probabilistic-based techniques, including supervised classifiers. Additionally, future research could explore

incorporating advanced ensemble learning techniques, which combine predictions from multiple LLM evaluators, to enhance the accuracy and reliability of hallucination detection systems.

Additionally, recent research such as Chen et al. (2024) has shown that normalizing probabilities not only by the size of the text but also by the number of similar responses obtained through Self-Consistency significantly improves results. This highlights the importance of hybrid methods and the value of combining the strengths of multiple approaches. Similar to Chen et al. (2024), the methodology developed in Chapter 7 can be extended with Self-Consistency.

Moreover, it is worth exploring the combination of not only token-level probabilities and Self-Consistency, but also techniques such as Self-Reflection, ensemble methods, and Mixture of Experts. The main challenge with large-scale combinations is the computational cost, which suggests that future directions must also focus on increasing the efficiency of these methods to make them usable in real-time while maintaining good latency for the user.

These directions could significantly improve the trustworthiness of LLMs in high-stakes domains such as legal compliance.

9.3.4 Ethical and Regulatory Implications

Finally, as LLMs become increasingly integrated into the legal compliance process, addressing their use's ethical and regulatory implications is essential. Future work should focus on developing guidelines and best practices for the ethical deployment of LLMs in legal settings, ensuring that these models are used responsibly and transparently. Beyond improving performance, it is equally important to identify scenarios where LLMs could potentially cause harm (bias, toxicity, wrong reasoning, etc...) in legal contexts or fail to comply with software regulations. As more problematic scenarios are discovered, further research can focus on mitigating these issues. However, if perfect accuracy is unattainable, it will be necessary to recognize the ongoing need

for human intervention, with future research aiming to minimize the frequency and extent of such interventions.

For this reason, more effort must be devoted to improving the explainability of LLM outputs. Currently, significant progress has been made in detecting and mitigating hallucinations, a major issue with LLMs. An interesting aspect of hallucinations is that methods like the one developed in Chapter 7 and similar approaches based on token probabilities Azaria and Mitchell (2023a); Chen et al. (2024) provide insight into why hallucinations occur by identifying which tokens are most likely causing the issue. The same effort should be made in research and ideas to achieve broader interpretability of LLM outputs in a generalized context.

9.4 Acknowledgements

This work was partially supported by the National Science Foundation under Grant Nos. 2039678, 2136961, and 2210091. The views expressed herein are solely those of the author and do not necessarily reflect those of the National Science Foundation.

Bibliography

Abdelfattah, A., M. Schiewe, J. Curtis, T. Cerny, and E. Song (2023). Towards security-aware microservices: On extracting endpoint data access operations to determine access rights. In *13th International Conference on Cloud Computing and Services Science (CLOSER 2023)*.

Abdelfattah, A. S. (2022). Microservices-based systems visualization: student research abstract. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, pp. 1460–1464.

Abdelfattah, A. S. (2024). Fostering microservice maintainability assurance through a comprehensive framework. *arXiv preprint arXiv:2407.16873*.

Abdelfattah, A. S. and T. Cerny (2023). Roadmap to reasoning in microservice systems: A rapid review. *Applied Sciences 13*(3).

Abdelfattah, A. S., A. Rodriguez, A. Walker, and T. Cerny (2023). Detecting semantic clones in microservices using components. *SN Computer Science 4*(5), 470.

Abualhaija, S., C. Arora, A. Sleimi, and L. C. Briand (2022). Automated question answering for improved understanding of compliance requirements: A multi-document study. In *2022 IEEE 30th International Requirements Engineering Conference (RE)*, pp. 39–50. IEEE.

Achiam, J., S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Adlakha, V., P. BehnamGhader, X. H. Lu, N. Meade, and S. Reddy (2023). Evaluating correctness and faithfulness of instruction-following models for question answering. *arXiv preprint arXiv:2307.16877*.

Agrawal, A., M. Suzgun, L. Mackey, and A. T. Kalai (2023). Do language models know when they're hallucinating references? *arXiv preprint arXiv:2305.18248*.

Ahmad, W. U., J. Chi, Y. Tian, and K.-W. Chang (2020). Policyqa: A reading comprehension dataset for privacy policies. *arXiv preprint arXiv:2010.02557*.

Ahmed, T. and P. Devanbu (2022). Few-shot training llms for project-specific code-summarization. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pp. 1–5.

Akça, O., G. Bayrak, A. M. Issifu, and M. C. Ganiz (2022). Traditional machine learning and deep learning-based text classification for turkish law documents

using transformers and domain adaptation. In *2022 International Conference on INnovations in Intelligent Systems and Applications (INISTA)*, pp. 1–6. IEEE.

Alamri, H., V. Cartillier, A. Das, J. Wang, A. Cherian, I. Essa, D. Batra, T. K. Marks, C. Hori, P. Anderson, et al. (2019). Audio visual scene-aware dialog. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7558–7567.

Aletras, N., D. Tsarapatsanis, D. Preoţiuc-Pietro, and V. Lampos (2016). Predicting judicial decisions of the european court of human rights: A natural language processing perspective. *PeerJ Computer Science 2*, e93.

Anand, D. and R. Wagh (2019). Effective deep learning approaches for summarization of legal texts. *Journal of King Saud University-Computer and Information Sciences*.

Azaria, A. and T. Mitchell (2023a, December). The internal state of an LLM knows when it's lying. In H. Bouamor, J. Pino, and K. Bali (Eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, Singapore, pp. 967–976. Association for Computational Linguistics.

Azaria, A. and T. Mitchell (2023b, December). The internal state of an LLM knows when it's lying. In H. Bouamor, J. Pino, and K. Bali (Eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, Singapore, pp. 967–976. Association for Computational Linguistics.

Bayat, F. F., K. Qian, B. Han, Y. Sang, A. Belyi, S. Khorshidi, F. Wu, I. F. Ilyas, and Y. Li (2023). Fleek: Factual error detection and correction with evidence retrieved from external knowledge. *arXiv preprint arXiv:2310.17119*.

Beltagy, I., M. E. Peters, and A. Cohan (2020). Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Bommasani, R., D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Brokos, G.-I., P. Malakasiotis, and I. Androutsopoulos (2016). Using centroids of word embeddings and word mover's distance for biomedical document retrieval in question answering. *arXiv preprint arXiv:1608.03905*.

Bushong, V., D. Das, A. Al Maruf, and T. Cerny (2021). Using static analysis to address microservice architecture reconstruction. In *ASE '21: Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering*, pp. Accepted to be published. IEEE/ACM.

Cao, H., Z. An, J. Feng, K. Xu, L. Chen, and D. Zhao (2023). A step closer to comprehensive answers: Constrained multi-stage question decomposition with large language models. *arXiv preprint arXiv:2311.07491*.

Cavusoglu, H., B. Mishra, and S. Raghunathan (2004). The effect of internet security breach announcements on market value: Capital market reactions for breached firms and internet security developers. *International Journal of Electronic Commerce 9*(1), 70–104.

Cerny, T., J. Svacina, D. Das, V. Bushong, M. Bures, P. Tisnovsky, K. Frajtak, D. Shin, and J. Huang (2020). On code analysis opportunities and challenges for enterprise systems and microservices. *IEEE Access*, 1–22.

Chalkidis, I., I. Androutsopoulos, and N. Aletras (2019). Neural legal judgment prediction in english. *arXiv preprint arXiv:1906.02059*.

Chalkidis, I., I. Androutsopoulos, and A. Michos (2017). Extracting contract elements. In *Proceedings of the 16th edition of the International Conference on Articial Intelligence and Law*, pp. 19–28.

Chalkidis, I., M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos (2020). Legal-bert: The muppets straight out of law school. *arXiv preprint arXiv:2010.02559*.

Chalkidis, I., M. Fergadiotis, P. Malakasiotis, and I. Androutsopoulos (2019). Large-scale multi-label text classification on eu legislation. *arXiv preprint arXiv:1906.02192*.

Chalkidis, I., M. Fergadiotis, N. Manginas, E. Katakalou, and P. Malakasiotis (2021). Regulatory compliance through doc2doc information retrieval: A case study in eu/uk legislation where text similarity has limitations. *arXiv preprint arXiv:2101.10726*.

Chalkidis, I., A. Jana, D. Hartung, M. Bommarito, I. Androutsopoulos, D. M. Katz, and N. Aletras (2021). Lexglue: A benchmark dataset for legal language understanding in english. *arXiv preprint arXiv:2110.00976*.

Chalkidis, I. and D. Kampas (2019). Deep learning in law: early adaptation and legal word embeddings trained on large corpora. *Artificial Intelligence and Law 27*(2), 171–198.

Chen, C., K. Liu, Z. Chen, Y. Gu, Y. Wu, M. Tao, Z. Fu, and J. Ye (2024). Inside: Llms' internal states retain the power of hallucination detection. *arXiv preprint arXiv:2402.03744*.

Chen, J., G. Kim, A. Sriram, G. Durrett, and E. Choi (2023). Complex claim verification with evidence retrieved in the wild. *arXiv preprint arXiv:2305.11859*.

Chen, J., A. Sriram, E. Choi, and G. Durrett (2022, December). Generating literal and implied subquestions to fact-check complex claims. In Y. Goldberg, Z. Kozareva, and Y. Zhang (Eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Abu Dhabi, United Arab Emirates, pp. 3495–3516. Association for Computational Linguistics.

Chen, M., J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba (2021). Evaluating large language models trained on code.

Chen, M., J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Cheng, D., S. Huang, J. Bi, Y. Zhan, J. Liu, Y. Wang, H. Sun, F. Wei, D. Deng, and Q. Zhang (2023). Uprise: Universal prompt retrieval for improving zero-shot evaluation. *arXiv preprint arXiv:2303.08518*.

Chern, I., S. Chern, S. Chen, W. Yuan, K. Feng, C. Zhou, J. He, G. Neubig, P. Liu, et al. (2023). Factool: Factuality detection in generative ai–a tool augmented framework for multi-task and multi-domain scenarios. *arXiv preprint arXiv:2307.13528*.

Chiang, C.-H. and H.-y. Lee (2023). Can large language models be an alternative to human evaluations? *arXiv preprint arXiv:2305.01937*.

Chrysostomou, G. and N. Aletras (2021). Enjoy the salience: Towards better transformer-based faithful explanations with word salience. *arXiv preprint arXiv:2108.13759*.

Chuang, Y.-S., Y. Xie, H. Luo, Y. Kim, J. Glass, and P. He (2023). Dola: Decoding by contrasting layers improves factuality in large language models. *arXiv preprint arXiv:2309.03883*.

Chung, H. W., L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, et al. (2024). Scaling instruction-finetuned language models. *Journal of Machine Learning Research 25*(70), 1–53.

Cohen, R., M. Hamri, M. Geva, and A. Globerson (2023). Lm vs lm: Detecting factual errors via cross examination. *arXiv preprint arXiv:2305.13281*.

Conover, W. J. and R. L. Iman (1981). Rank transformations as a bridge between parametric and nonparametric statistics. *The American Statistician 35*(3), 124–129.

Cram, W. A., J. D'arcy, and J. G. Proudfoot (2019). Seeing the forest and the trees: a meta-analysis of the antecedents to information security policy compliance. *MIS quarterly 43*(2), 525–554.

Cram, W. A., J. G. Proudfoot, and J. D'arcy (2017). Organizational information security policies: a review and research framework. *European Journal of Information Systems 26*, 605–641.

Cui, J., Z. Li, Y. Yan, B. Chen, and L. Yuan (2023). Chatlaw: Open-source legal large language model with integrated external knowledge bases. *arXiv preprint arXiv:2306.16092*.

Curran, J. R., S. Clark, and J. Bos (2007). Linguistically motivated large-scale nlp with c&c and boxer. In *Proceedings of the 45th annual meeting of the Association for Computational Linguistics Companion volume proceedings of the demo and poster sessions*, pp. 33–36.

D'Arcy, J., T. Herath, and M. K. Shoss (2014). Understanding employee responses to stressful information security requirements: A coping perspective. *Journal of management information systems 31*(2), 285–318.

Das, D., A. Walker, V. Bushong, J. Svacina, T. Cerny, and V. Matyas (2021). On automated rbac assessment by constructing a centralized perspective for microservice mesh. *PeerJ Computer Science 7*, e376.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dhuliawala, S., M. Komeili, J. Xu, R. Raileanu, X. Li, A. Celikyilmaz, and J. Weston (2023). Chain-of-verification reduces hallucination in large language models. *arXiv preprint arXiv:2309.11495*.

Do, P.-K., H.-T. Nguyen, C.-X. Tran, M.-T. Nguyen, and M.-L. Nguyen (2017). Legal question answering using ranking svm and deep convolutional neural network. *arXiv preprint arXiv:1703.05320*.

Dong, Y., X. Jiang, Z. Jin, and G. Li (2023). Self-collaboration code generation via chatgpt. *arXiv preprint arXiv:2304.07590*.

Dragoni, M., S. Villata, W. Rizzi, and G. Governatori (2016). Combining nlp approaches for rule extraction from legal documents. In *1st Workshop on MIning and REasoning with Legal texts (MIREL 2016)*.

Duan, X., B. Wang, Z. Wang, W. Ma, Y. Cui, D. Wu, S. Wang, T. Liu, T. Huo, Z. Hu, et al. (2019). Cjrc: A reliable human-annotated benchmark dataset for chinese judicial reading comprehension. In *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18*, pp. 439–451. Springer.

Duan, X., Y. Zhang, L. Yuan, X. Zhou, X. Liu, T. Wang, R. Wang, Q. Zhang, C. Sun, and F. Wu (2019). Legal summarization for multi-role debate dialogue via

controversy focus mining and multi-task learning. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pp. 1361–1370.

Durmus, E., H. He, and M. Diab (2020). Feqa: A question answering evaluation framework for faithfulness assessment in abstractive summarization. *arXiv preprint arXiv:2005.03754*.

Dziri, N., A. Madotto, O. Zaïane, and A. J. Bose (2021). Neural path hunter: Reducing hallucination in dialogue systems via path grounding. *arXiv preprint arXiv:2104.08455*.

Dziri, N., H. Rashkin, T. Linzen, and D. Reitter (2021). Evaluating groundedness in dialogue systems: The begin benchmark. *arXiv preprint arXiv:2105.00071 4*.

Elaraby, M., M. Lu, J. Dunn, X. Zhang, Y. Wang, S. Liu, P. Tian, Y. Wang, and Y. Wang (2023). Halo: Estimation and reduction of hallucinations in open-source weak large language models. *arXiv preprint arXiv:2308.11764*.

Falke, T., L. F. Ribeiro, P. A. Utama, I. Dagan, and I. Gurevych (2019). Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 2214–2220.

Fan, T., Y. Kang, G. Ma, W. Chen, W. Wei, L. Fan, and Q. Yang (2023). Fate-llm: A industrial grade federated learning framework for large language models. *arXiv preprint arXiv:2310.10049*.

Fang, X. and X. Zhao (2018). Nonlinear dimensionality reduction with judicial document learning. In *2018 IEEE International Conference on Big Knowledge (ICBK)*, pp. 448–455. IEEE.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association 32*(200), 675–701.

Galitsky, B. A. (2023). Truth-o-meter: Collaborating with llm in fighting its hallucinations.(2023).

Gao, L., Z. Dai, P. Pasupat, A. Chen, A. T. Chaganty, Y. Fan, V. Y. Zhao, N. Lao, H. Lee, D.-C. Juan, and K. Guu (2023). Rarr: Researching and revising what language models say, using language models.

Gao, M., J. Ruan, R. Sun, X. Yin, S. Yang, and X. Wan (2023). Human-like summarization evaluation with chatgpt. *arXiv preprint arXiv:2304.02554*.

Gao, Y., Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang (2023). Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

García-Constantino, M., K. Atkinson, D. Bollegala, K. Chapman, F. Coenen, C. Roberts, and K. Robson (2017). Cliel: context-based information extraction from commercial law documents. In *Proceedings of the 16th edition of the International Conference on Artificial Intelligence and Law*, pp. 79–87.

Ge, J., Y. Huang, X. Shen, C. Li, and W. Hu (2021). Learning fine-grained fact-article correspondence in legal cases. *IEEE/ACM Transactions on Audio, Speech, and Language Processing 29*, 3694–3706.

Goode, S., H. Hoehle, V. Venkatesh, and S. A. Brown (2017). User compensation as a data breach recovery action. *MIS Quarterly 41*(3), 703–A16.

Goodrich, B., V. Rao, P. J. Liu, and M. Saleh (2019). Assessing the factual accuracy of generated text. In *proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 166–175.

Goyal, T. and G. Durrett (2020). Evaluating factuality in generation with dependency-level entailment. *arXiv preprint arXiv:2010.05478*.

Hadi, M. U., R. Qureshi, A. Shah, M. Irfan, A. Zafar, M. B. Shaikh, N. Akhtar, J. Wu, S. Mirjalili, et al. (2023). A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*.

Haislip, J., J.-H. Lim, and R. Pinsker (2021). The impact of executives' it expertise on reported data security breaches. *Information Systems Research 32*(2), 318–334.

Harkous, H., K. Fawaz, R. Lebret, F. Schaub, K. G. Shin, and K. Aberer (2018). Polisis: Automated analysis and presentation of privacy policies using deep learning. In *27th USENIX Security Symposium (USENIX Security 18)*, pp. 531–548.

Hey, T., J. Keim, A. Koziolek, and W. F. Tichy (2020). Norbert: Transfer learning for requirements classification. In *2020 IEEE 28th international requirements engineering conference (RE)*, pp. 169–179. IEEE.

Hong, S., X. Zheng, J. Chen, Y. Cheng, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, et al. (2023). Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*.

Hörnemalm, A. (2023). ChatGPT as a Software Development Tool: The Future of Development.

Hu, E. J., Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Hu, X., K. Kuang, J. Sun, H. Yang, and F. Wu (2024). Leveraging print debugging to improve code generation in large language models. *arXiv preprint arXiv:2401.05319*.

Hua, X., A. Sreevatsa, and L. Wang (2021). Dyploc: Dynamic planning of content using mixed language models for text generation. *arXiv preprint arXiv:2106.00791*.

Huang, L., W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, et al. (2023). A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*.

Huang, W., X. Liao, Z. Xie, J. Qian, B. Zhuang, S. Wang, and J. Xiao (2021). Generating reasonable legal text through the combination of language modeling and question answering. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pp. 3687–3693.

Huo, S., N. Arabzadeh, and C. L. Clarke (2023). Retrieving supporting evidence for llms generated answers. *arXiv preprint arXiv:2306.13781*.

Husain, H., H.-H. Wu, T. Gazit, M. Allamanis, and M. Brockschmidt (2019). Codesearchnet challenge: Evaluating the state of semantic code search. *arXiv preprint arXiv:1909.09436*.

Jain, S., V. Keshava, S. M. Sathyendra, P. Fernandes, P. Liu, G. Neubig, and C. Zhou (2023). Multi-dimensional evaluation of text summarization with in-context learning. *arXiv preprint arXiv:2306.01200*.

Jalil, S., S. Rafi, T. D. LaToza, K. Moran, and W. Lam (2023). Chatgpt and software testing education: Promises & perils. In *2023 IEEE international conference on software testing, verification and validation workshops (ICSTW)*, pp. 4130–4137. IEEE.

Jayasinghe, S., L. Rambukkanage, A. Silva, N. de Silva, and A. S. Perera (2021). Critical sentence identification in legal cases using multi-class classification. In *2021 IEEE 16th International Conference on Industrial and Information Systems (ICIIS)*, pp. 146–151. IEEE.

Ji, D., P. Tao, H. Fei, and Y. Ren (2020). An end-to-end joint model for evidence information extraction from court record document. *Information Processing & Management 57*(6), 102305.

Ji, Z., N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys 55*(12), 1–38.

Ji, Z., Z. Liu, N. Lee, T. Yu, B. Wilie, M. Zeng, and P. Fung (2023, July). RHO: Reducing hallucination in open-domain dialogues with knowledge grounding. In A. Rogers, J. Boyd-Graber, and N. Okazaki (Eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, Toronto, Canada, pp. 4504–4522. Association for Computational Linguistics.

Ji, Z., T. Yu, Y. Xu, N. Lee, E. Ishii, and P. Fung (2023). Towards mitigating hallucination in large language models via self-reflection. *arXiv preprint arXiv:2310.06271*.

Jiang, J., F. Wang, J. Shen, S. Kim, and S. Kim (2024). A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*.

Jiang, T., D. Wang, L. Sun, H. Yang, Z. Zhao, and F. Zhuang (2021). Lightxml: Transformer with dynamic negative sampling for high-performance extreme multi-label text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Volume 35, pp. 7987–7994.

Jin, H., L. Huang, H. Cai, J. Yan, B. Li, and H. Chen (2024). From llms to llm-based agents for software engineering: A survey of current, challenges and future. *arXiv preprint arXiv:2408.02479*.

John, A. K., L. Di Caro, and G. Boella (2017). Solving bar exam questions with deep neural networks. In *Proceedings of the Second Workshop on Automated Semantic Analysis of Information in Legal Texts: co-located with the 16th International Conference on Artificial Intelligence and Law*.

Jones, E., H. Palangi, C. Simões, V. Chandrasekaran, S. Mukherjee, A. Mitra, A. Awadallah, and E. Kamar (2023). Teaching language models to hallucinate less with synthetic tasks. *arXiv preprint arXiv:2310.06827*.

Joshi, M., E. Choi, D. S. Weld, and L. Zettlemoyer (2017). Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Jurafsky, D. and J. H. Martin (2022). Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.

Kang, H., J. Ni, and H. Yao (2023). Ever: Mitigating hallucination in large language models through real-time verification and rectification. *arXiv preprint arXiv:2311.09114*.

Kapitsaki, G. M. and D. Paschalides (2017). Identifying terms in open source software license texts. In *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*, pp. 540–545. IEEE.

Kien, P. M., H.-T. Nguyen, N. X. Bach, V. Tran, M. Le Nguyen, and T. M. Phuong (2020). Answering legal questions by learning neural attentive text representation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 988–998.

Kingma, D. P. and J. Ba (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Köksal, A., R. Aksitov, and C.-C. Chang (2023). Hallucination augmented recitations for language models. *arXiv preprint arXiv:2311.07424*.

Kornilova, A. and V. Eidelman (2019). Billsum: A corpus for automatic summarization of us legislation. *arXiv preprint arXiv:1910.00523*.

Krishna, M., B. Gaur, A. Verma, and P. Jalote (2024). Using llms in software requirements specifications: An empirical evaluation. *arXiv preprint arXiv:2404.17842*.

Kumar, J. and S. Chimalakonda (2024). Code summarization without direct access to code-towards exploring federated llms for software engineering. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, pp. 100–109.

Kwiatkowski, T., J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, et al. (2019). Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics 7*, 453–466.

Laban, P., W. Kryściński, D. Agarwal, A. R. Fabbri, C. Xiong, S. Joty, and C.-S. Wu (2023). Llms as factual reasoners: Insights from existing benchmarks and beyond. *arXiv preprint arXiv:2305.14540*.

Ladhak, F., E. Durmus, M. Suzgun, T. Zhang, D. Jurafsky, K. McKeown, and T. Hashimoto (2023, May). When do pre-training biases propagate to downstream tasks? a case study in text summarization. In A. Vlachos and I. Augenstein (Eds.), *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, Dubrovnik, Croatia, pp. 3206–3219. Association for Computational Linguistics.

Landthaler, J., B. Waltl, P. Holl, and F. Matthes (2016). Extending full text search for legal document collections using word embeddings. In *JURIX*, pp. 73–82.

Le, Q. and T. Mikolov (2014). Distributed representations of sentences and documents. In *International conference on machine learning*, pp. 1188–1196. PMLR.

Le Scao, T., A. Fan, C. Akiki, E. Pavlick, S. Ilić, D. Hesslow, R. Castagné, A. S. Luccioni, F. Yvon, M. Gallé, et al. (2023). Bloom: A 176b-parameter open-access multilingual language model.

Lee, M. (2023). A mathematical investigation of hallucination and creativity in gpt models. *Mathematics 11*(10), 2320.

Lei, D., Y. Li, M. Wang, V. Yun, E. Ching, E. Kamal, et al. (2023). Chain of natural language inference for reducing large language model ungrounded hallucinations. *arXiv preprint arXiv:2310.03951*.

Lewis, M., Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer (2020, July). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, pp. 7871–7880. Association for Computational Linguistics.

Lewis, P., E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems 33*, 9459–9474.

Li, H., M. Tomko, M. Vasardani, and T. Baldwin (2022). Multispanqa: A dataset for multi-span question answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1250–1260.

Li, J., J. Chen, R. Ren, X. Cheng, W. X. Zhao, J.-Y. Nie, and J.-R. Wen (2024). The dawn after the dark: An empirical study on factuality hallucination in large language models. *arXiv preprint arXiv:2401.03205*.

Li, J., X. Cheng, X. Zhao, J.-Y. Nie, and J.-R. Wen (2023, December). HaluEval: A large-scale hallucination evaluation benchmark for large language models. In H. Bouamor, J. Pino, and K. Bali (Eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Singapore, pp. 6449–6464. Association for Computational Linguistics.

Li, K., O. Patel, F. Viégas, H. Pfister, and M. Wattenberg (2024). Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems 36*.

Liga, D. and L. Robaldo (2023). Fine-tuning gpt-3 for legal rule classification. *Computer Law & Security Review 51*, 105864.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81.

Lin, S., J. Hilton, and O. Evans (2022, May). TruthfulQA: Measuring how models mimic human falsehoods. In S. Muresan, P. Nakov, and A. Villavicencio (Eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Dublin, Ireland, pp. 3214–3252. Association for Computational Linguistics.

Lippi, M., P. Pałka, G. Contissa, F. Lagioia, H.-W. Micklitz, G. Sartor, and P. Torroni (2019). Claudette: an automated detector of potentially unfair clauses in online terms of service. *Artificial Intelligence and Law 27*, 117–139.

Liu, W., G. Li, K. Zhang, B. Du, Q. Chen, X. Hu, H. Xu, J. Chen, and J. Wu (2023). Mind's mirror: Distilling self-evaluation capability and comprehensive thinking from large language models. *arXiv preprint arXiv:2311.09214*.

Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Locke, D., G. Zuccon, and H. Scells (2017). Automatic query generation from legal texts for case law retrieval. In *Information Retrieval Technology: 13th Asia Information Retrieval Societies Conference, AIRS 2017, Jeju Island, South Korea, November 22-24, 2017, Proceedings 13*, pp. 181–193. Springer.

Luo, B., Y. Feng, J. Xu, X. Zhang, and D. Zhao (2017). Learning to predict charges for criminal cases with legal basis. *arXiv preprint arXiv:1707.09168*.

Luo, X., Y. Xue, Z. Xing, and J. Sun (2022). Prcbert: Prompt learning for requirement classification using bert-based pretrained language models. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pp. 1–13.

Luo, Z., Q. Xie, and S. Ananiadou (2023). Chatgpt as a factual inconsistency evaluator for text summarization. *arXiv preprint arXiv:2303.15621*.

Lyu, Y., Z. Wang, Z. Ren, P. Ren, Z. Chen, X. Liu, Y. Li, H. Li, and H. Song (2022). Improving legal judgment prediction through reinforced criminal element extraction. *Information Processing & Management 59*(1), 102780.

Ma, Y., Z. Lan, L. Zong, and K. Huang (2021). Global-aware beam search for neural abstractive summarization. *Advances in Neural Information Processing Systems 34*, 16545–16557.

Ma, Y., Y. Shao, Y. Wu, Y. Liu, R. Zhang, M. Zhang, and S. Ma (2021). Lecard: a legal case retrieval dataset for chinese law system. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pp. 2342–2348.

Manakul, P., A. Liusie, and M. Gales (2023, December). SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In H. Bouamor, J. Pino, and K. Bali (Eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Singapore, pp. 9004–9017. Association for Computational Linguistics.

Martins, A. and R. Astudillo (2016). From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International conference on machine learning*, pp. 1614–1623. PMLR.

Maynez, J., S. Narayan, B. Bohnet, and R. McDonald (2020). On faithfulness and factuality in abstractive summarization. *arXiv preprint arXiv:2005.00661*.

Merchant, K. and Y. Pande (2018). Nlp based latent semantic analysis for legal text summarization. In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1803–1807. IEEE.

Merity, S., C. Xiong, J. Bradbury, and R. Socher (2016). Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Mihalcea, R. and P. Tarau (2004). Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pp. 404–411.

Mikolov, T., K. Chen, G. Corrado, and J. Dean (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Milne, G. R. and M. J. Culnan (2004). Strategies for reducing online privacy risks: Why consumers read (or don't read) online privacy notices. *Journal of interactive marketing 18*(3), 15–29.

Min, S., K. Krishna, X. Lyu, M. Lewis, W.-t. Yih, P. W. Koh, M. Iyyer, L. Zettlemoyer, and H. Hajishirzi (2023). Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. *arXiv preprint arXiv:2305.14251*.

Mishra, A., D. Patel, A. Vijayakumar, X. L. Li, P. Kapanipathi, and K. Talamadupula (2021). Looking beyond sentence-level natural language inference for question answering and text summarization. In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 1322–1336.

Moon, S., P. Shah, A. Kumar, and R. Subba (2019). Opendialkg: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 845–854.

Mudalige, C. R., D. Karunarathna, I. Rajapaksha, N. de Silva, G. Ratnayaka, A. S. Perera, and R. Pathirana (2020). Sigmalaw-absa: dataset for aspect-based sentiment analysis in legal opinion texts. In *2020 IEEE 15th international conference on industrial and information systems (ICIIS)*, pp. 488–493. IEEE.

Mullenbach, J., S. Wiegreffe, J. Duke, J. Sun, and J. Eisenstein (2018). Explainable prediction of medical codes from clinical text. *arXiv preprint arXiv:1802.05695*.

Mündler, N., J. He, S. Jenko, and M. Vechev (2023). Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation. *arXiv preprint arXiv:2305.15852*.

Nan, F., R. Nallapati, Z. Wang, C. N. d. Santos, H. Zhu, D. Zhang, K. McKeown, and B. Xiang (2021). Entity-level factual consistency of abstractive text summarization. *arXiv preprint arXiv:2102.09130*.

Narayan, S., S. B. Cohen, and M. Lapata (2018, October-November). Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii (Eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, pp. 1797–1807. Association for Computational Linguistics.

Nemenyi, P. B. (1963). *Distribution-free multiple comparisons*. Princeton University.

Nenkova, A. and R. Passonneau (2004, May 2 - May 7). Evaluating content selection in summarization: The pyramid method. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, Boston, Massachusetts, USA, pp. 145–152. Association for Computational Linguistics.

Ni, A., P. Yin, Y. Zhao, M. Riddell, T. Feng, R. Shen, S. Yin, Y. Liu, S. Yavuz, C. Xiong, et al. (2023). L2ceval: Evaluating language-to-code generation capabilities of large language models. *arXiv preprint arXiv:2309.17446*.

Nie, Y., A. Williams, E. Dinan, M. Bansal, J. Weston, and D. Kiela (2020, July). Adversarial NLI: A new benchmark for natural language understanding. In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, pp. 4885–4901. Association for Computational Linguistics.

Noguti, M. Y., E. Vellasques, and L. S. Oliveira (2020). Legal document classification: An application to law area prediction of petitions to public prosecution service. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE.

Nokhbeh Zaeem, R. and K. S. Barber (2021). A large publicly available corpus of website privacy policies based on dmoz. In *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, pp. 143–148.

Pan, S., L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu (2024). Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*.

Parker, G., S. Kim, A. Al Maruf, T. Cerny, K. Frajtak, P. Tisnovsky, and D. Taibi (2023). Visualizing anti-patterns in microservices at runtime: A systematic mapping study. *IEEE Access*.

Patel, A., S. Bhattamishra, and N. Goyal (2021, June). Are NLP models really able to solve simple math word problems? In K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou (Eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online, pp. 2080–2094. Association for Computational Linguistics.

Paul, S., P. Goyal, and S. Ghosh (2022). Lesicin: A heterogeneous graph-based approach for automatic legal statute identification from indian legal documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Volume 36, pp. 11139–11146.

Peng, B., M. Galley, P. He, H. Cheng, Y. Xie, Y. Hu, Q. Huang, L. Liden, Z. Yu, W. Chen, et al. (2023). Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*.

Peng, S., E. Kalliamvakou, P. Cihon, and M. Demirer (2023). The impact of ai on developer productivity: Evidence from github copilot. *arXiv preprint arXiv:2302.06590*.

Pfeiffer, J., I. Vulić, I. Gurevych, and S. Ruder (2020, November). MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In B. Webber, T. Cohn, Y. He, and Y. Liu (Eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, pp. 7654–7673. Association for Computational Linguistics.

Pillai, V. G. and L. R. Chandran (2020). Verdict prediction for indian courts using bag of words and convolutional neural network. In *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pp. 676–683. IEEE.

Polsley, S., P. Jhunjhunwala, and R. Huang (2016). Casesummarizer: a system for automated summarization of legal texts. In *Proceedings of COLING 2016, the 26th international conference on Computational Linguistics: System Demonstrations*, pp. 258–262.

Ponte, J. M. and W. B. Croft (2017). A language modeling approach to information retrieval. In *ACM SIGIR Forum*, Volume 51, pp. 202–208. ACM New York, NY, USA.

Pothong, S. and N. Facundes (2021). Coreference resolution and meaning representation in a legislative corpus. In *2021 16th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)*, pp. 1–6. IEEE.

Press, O., M. Zhang, S. Min, L. Schmidt, N. A. Smith, and M. Lewis (2022). Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*.

Qian, C., W. Liu, H. Liu, N. Chen, Y. Dang, J. Li, C. Yang, W. Chen, Y. Su, X. Cong, et al. (2024). Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15174–15186.

Qin, R., M. Huang, and Y. Luo (2022). A comparison study of pre-trained language models for chinese legal document classification. In *2022 5th International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pp. 444–449. IEEE.

Qiu, Y., V. Embar, S. B. Cohen, and B. Han (2023). Think while you write: Hypothesis verification promotes faithful knowledge-to-text generation. *arXiv preprint arXiv:2311.09467*.

Qiu, Y., Y. Ziser, A. Korhonen, E. M. Ponti, and S. B. Cohen (2023). Detecting and mitigating hallucinations in multilingual summarisation. *arXiv preprint arXiv:2305.13632*.

Quevedo, E., T. Cerny, A. Rodriguez, P. Rivas, J. Yero, K. Sooksatra, A. Zhakubayev, and D. Taibi (2023). Legal natural language processing from 2015-2022: A comprehensive systematic mapping study of advances and applications. *IEEE Access*.

Rabelo, J., M.-Y. Kim, and R. Goebel (2019). Combining similarity and transformer methods for case law entailment. In *Proceedings of the seventeenth international conference on artificial intelligence and law*, pp. 290–296.

Radford, A., K. Narasimhan, T. Salimans, I. Sutskever, et al. (2018). Improving language understanding by generative pre-training.

Radford, A., J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog 1*(8), 9.

Rajpurkar, P., R. Jia, and P. Liang (2018). Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Rajpurkar, P., J. Zhang, K. Lopyrev, and P. Liang (2016). Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Rawte, V., S. Chakraborty, A. Pathak, A. Sarkar, S. Tonmoy, A. Chadha, A. P. Sheth, and A. Das (2023). The troubling emergence of hallucination in large language models–an extensive definition, quantification, and prescriptive remediations. *arXiv preprint arXiv:2310.04988*.

Ray, P. P. (2023). Chatgpt: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems*.

Reddy, S., D. Chen, and C. D. Manning (2019). Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics 7*, 249–266.

Reimers, N. and I. Gurevych (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Robertson, S. E., S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, et al. (1995). Okapi at trec-3. *Nist Special Publication Sp 109*, 109.

Ronanki, K., B. Cabrero-Daniel, and C. Berger (2022). Chatgpt as a tool for user story quality evaluation: Trustworthy out of the box? In *International Conference on Agile Software Development*, pp. 173–181. Springer.

Rosili, N. A. K., N. H. Zakaria, R. Hassan, S. Kasim, F. Z. C. Rose, and T. Sutikno (2021). A systematic literature review of machine learning methods in predicting court decisions. *IAES International Journal of Artificial Intelligence 10*(4), 1091.

Roziere, B., J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, R. Sauvestre, T. Remez, et al. (2023). Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Sahoo, P., A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha (2024). A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*.

Sanh, V., L. Debut, J. Chaumond, and T. Wolf (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Sannier, N., M. Adedjouma, M. Sabetzadeh, and L. Briand (2017). An automated framework for detection and resolution of cross references in legal texts. *Requirements Engineering 22*(2), 215–237.

Sansone, C. and G. Sperli (2022). Legal information retrieval systems: State-of-the-art and open issues. *Information Systems 106*, 101967.

Santhanam, S., B. Hedayatnia, S. Gella, A. Padmakumar, S. Kim, Y. Liu, and D. Hakkani-Tur (2021). Rome was built in 1776: A case study on factual correctness in knowledge-grounded response generation. *arXiv preprint arXiv:2110.05456*.

Saputro, D. R. S. and P. Widyaningsih (2017). Limited memory broyden-fletcher-goldfarb-shanno (l-bfgs) method for the parameter estimation on geographically weighted ordinal logistic regression model (gwolr). *AIP Conference Proceedings*.

Saxena, A. and P. Bhattacharyya (2024). Hallucination detection in machine generated text: A survey.

Schlackl, F., N. Link, and H. Hoehle (2022). Antecedents and consequences of data breaches: A systematic review. *Information & Management*, 103638.

Schuetz, S. W., P. Benjamin Lowry, D. A. Pienta, and J. Bennett Thatcher (2020). The effectiveness of abstract versus concrete fear appeals in information security. *Journal of Management Information Systems 37*(3), 723–757.

Shaheen, Z., G. Wohlgenannt, and D. Mouromtsev (2021). Zero-shot cross-lingual transfer in legal domain using transformer models. In *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 450–456. IEEE.

Shankar, A., A. Waldis, C. Bless, M. Andueza Rodriguez, and L. Mazzola (2023). Privacyglue: A benchmark dataset for general language understanding in privacy policies. *Applied Sciences 13*(6), 3701.

Shao, Y., J. Mao, Y. Liu, W. Ma, K. Satoh, M. Zhang, and S. Ma (2020). Bert-pli: Modeling paragraph-level interactions for legal case retrieval. In *IJCAI*, pp. 3501–3507.

Shavrina, T. and V. Malykh (2021). How not to lie with a benchmark: Rearranging nlp learderboards.

Shi, C., H. Yang, D. Cai, Z. Zhang, Y. Wang, Y. Yang, and W. Lam (2024). A thorough examination of decoding methods in the era of llms. *arXiv preprint arXiv:2402.06925*.

Shi, W., X. Han, M. Lewis, Y. Tsvetkov, L. Zettlemoyer, and S. W.-t. Yih (2023). Trusting your evidence: Hallucinate less with context-aware decoding. *arXiv preprint arXiv:2305.14739*.

Shulayeva, O., A. Siddharthan, and A. Wyner (2017). Recognizing cited facts and principles in legal judgements. *Artificial Intelligence and Law 25*(1), 107–126.

Shuster, K., S. Poff, M. Chen, D. Kiela, and J. Weston (2021). Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*.

Si, C., Z. Gan, Z. Yang, S. Wang, J. Wang, J. Boyd-Graber, and L. Wang (2023). Prompting gpt-3 to be reliable.

Song, D., S. Gao, B. He, and F. Schilder (2022). On the effectiveness of pre-trained language models for legal natural language processing: An empirical study. *IEEE Access 10*, 75835–75858.

Song, D., A. Vold, K. Madan, and F. Schilder (2022). Multi-label legal document classification: A deep learning-based approach with label-attention and domain-specific pre-training. *Information Systems 106*, 101718.

Sridhara, G., S. Mazumdar, et al. (2023). Chatgpt: A study on its utility for ubiquitous software engineering tasks. *arXiv preprint arXiv:2305.16837*.

Srinath, M., S. Wilson, and C. L. Giles (2020). Privacy at scale: Introducing the privaseer corpus of web privacy policies. *arXiv preprint arXiv:2004.11131*.

Steidl, D., B. Hummel, and E. Juergens (2013). Quality analysis of source code comments. In *2013 21st international conference on program comprehension (icpc)*, pp. 83–92. Ieee.

Su, W., C. Wang, Q. Ai, Y. Hu, Z. Wu, Y. Zhou, and Y. Liu (2024). Unsupervised real-time hallucination detection based on the internal states of large language models. *arXiv preprint arXiv:2403.06448*.

Sun, R., S. Ö. Arik, A. Muzio, L. Miculicich, S. Gundabathula, P. Yin, H. Dai, H. Nakhost, R. Sinha, Z. Wang, et al. (2023). Sql-palm: Improved large language model adaptation for text-to-sql (extended). *arXiv preprint arXiv:2306.00739*.

Sun, W., C. Fang, Y. You, Y. Chen, Y. Liu, C. Wang, J. Zhang, Q. Zhang, H. Qian, W. Zhao, et al. (2023). A prompt learning framework for source code summarization. *arXiv preprint arXiv:2312.16066*.

Sun, W., Y. Miao, Y. Li, H. Zhang, C. Fang, Y. Liu, G. Deng, Y. Liu, and Z. Chen (2024). Source code summarization in the era of large language models. *arXiv preprint arXiv:2407.07959*.

Suri, S., S. N. Das, K. Singi, K. Dey, V. S. Sharma, and V. Kaulgud (2023). Software engineering using autonomous agents: Are we there yet? In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 1855–1857. IEEE.

Taibi, D., V. Lenarduzzi, C. Pahl, and A. Janes (2017). Microservices in agile software development: a workshop-based study into issues, advantages, and disadvantages. In *Proceedings of the XP2017 Scientific Workshops*, pp. 1–5.

Talmor, A., J. Herzig, N. Lourie, and J. Berant (2019, June). CommonsenseQA: A question answering challenge targeting commonsense knowledge. In J. Burstein, C. Doran, and T. Solorio (Eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, pp. 4149–4158. Association for Computational Linguistics.

Team, G., T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, et al. (2024). Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Tian, K., E. Mitchell, H. Yao, C. D. Manning, and C. Finn (2023). Fine-tuning language models for factuality. *arXiv preprint arXiv:2311.08401*.

Tieu, T.-T., C.-N. Chau, T.-S. Nguyen, L.-M. Nguyen, et al. (2021). Apply bert-based models and domain knowledge for automated legal question answering tasks at alqac 2021. In *2021 13th International Conference on Knowledge and Systems Engineering (KSE)*, pp. 1–6. IEEE.

Touvron, H., L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Tran, V., M. Le Nguyen, S. Tojo, and K. Satoh (2020). Encoded summarization: summarizing documents into continuous vector space for legal case retrieval. *Artificial Intelligence and Law 28*, 441–467.

Tran, V., M. L. Nguyen, and K. Satoh (2019). Building legal case retrieval systems with lexical matching and summarization using a pre-trained phrase scoring model. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law*, pp. 275–282.

Trinh, T. H. and Q. V. Le (2018). A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

Undavia, S., A. Meyers, and J. E. Ortega (2018). A comparative study of classifying legal documents with neural networks. In *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 515–522. IEEE.

Vallecillos Ruiz, F. (2024). Agent-driven automatic software improvement. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, pp. 470–475.

Varshney, N., W. Yao, H. Zhang, J. Chen, and D. Yu (2023). A stitch in time saves nine: Detecting and mitigating hallucinations of llms by validating low-confidence generation. *arXiv preprint arXiv:2307.03987*.

Viswanath, H. and T. Zhang (2023). Fairpy: A toolkit for evaluation of social biases and their mitigation in large language models. *arXiv preprint arXiv:2302.05508*.

Vold, A. and J. G. Conrad (2021). Using transformers to improve answer retrieval for legal questions. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*, pp. 245–249.

Vu, T., M. Iyyer, X. Wang, N. Constant, J. Wei, J. Wei, C. Tar, Y.-H. Sung, D. Zhou, Q. Le, et al. (2023). Freshllms: Refreshing large language models with search engine augmentation. *arXiv preprint arXiv:2310.03214*.

Walker, A., D. Das, and T. Cerny (2020). Automated code-smell detection in microservices through static analysis: A case study. *Applied Sciences 10*(21).

Walker, A., I. Laird, and T. Cerny (2021, December). On automatic software architecture reconstruction of microservice applications. In *Information Science and Applications*, pp. (in print). Springer Singapore.

Wan, L. J., Y. Huang, Y. Li, H. Ye, J. Wang, X. Zhang, and D. Chen (2024). Software/hardware co-design for llm and its application for design verification. In *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 435–441. IEEE.

Wang, B. and A. Komatsuzaki (2021). Gpt-j-6b: A 6 billion parameter autoregressive language model.

Wang, T., P. Liang, and M. Lu (2018). What aspects do non-functional requirements in app user reviews describe? an exploratory and comparative study. In *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*, pp. 494–503. IEEE.

Wang, X., J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou (2022). Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Wang, Z., X. Wang, B. An, D. Yu, and C. Chen (2020). Towards faithful neural table-to-text generation with content-matching constraints. *arXiv preprint arXiv:2005.00969*.

Weber, I. (2024). Large language models as software components: A taxonomy for llm-integrated applications. *arXiv preprint arXiv:2406.10300*.

Wei, A., N. Haghtalab, and J. Steinhardt (2024). Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems 36*.

Weston, J., S. Chopra, and A. Bordes (2014). Memory networks. *arXiv preprint arXiv:1410.3916*.

White, J., S. Hays, Q. Fu, J. Spencer-Smith, and D. C. Schmidt (2024). Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. In *Generative AI for Effective Software Development*, pp. 71–108. Springer.

Wilson, S., F. Schaub, A. A. Dara, F. Liu, S. Cherivirala, P. G. Leon, M. S. Andersen, S. Zimmeck, K. M. Sathyendra, N. C. Russell, et al. (2016). The creation and analysis of a website privacy policy corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1330–1340.

Wyner, A. Z., B. J. Fawei, and J. Z. Pan (2016). Passing a usa national bar exam: a first corpus for experimentation. In *LREC 2016, Tenth International Conference on Language Resources and Evaluation*. LREC.

Xia, X., L. Bao, D. Lo, Z. Xing, A. E. Hassan, and S. Li (2017). Measuring program comprehension: A large-scale field study with professionals. *IEEE Transactions on Software Engineering 44*(10), 951–976.

Xiao, C., X. Hu, Z. Liu, C. Tu, and M. Sun (2021). Lawformer: A pre-trained language model for chinese legal long documents. *AI Open 2*, 79–84.

Xu, F., W. Shi, and E. Choi (2023). Recomp: Improving retrieval-augmented lms with compression and selective augmentation. *arXiv preprint arXiv:2310.04408*.

Xu, K., J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pp. 2048–2057. PMLR.

Xu, Z., S. Jain, and M. Kankanhalli (2024). Hallucination is inevitable: An innate limitation of large language models. *arXiv preprint arXiv:2401.11817*.

Yan, J. N., T. Liu, J. Chiu, J. Shen, Z. Qin, Y. Yu, C. Lakshmanan, Y. Kurzion, A. M. Rush, J. Liu, et al. (2024). Predicting text preference via structured comparative reasoning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10040–10060.

Yang, H., S. Yue, and Y. He (2023). Auto-gpt for online decision making: Benchmarks and additional opinions. *arXiv preprint arXiv:2306.02224*.

Yang, Z., Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems 32*.

Yang, Z., P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning (2018, October-November). HotpotQA: A dataset for diverse, explainable multi-hop question answering. In E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii (Eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, pp. 2369–2380. Association for Computational Linguistics.

Yang, Z., D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy (2016). Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 1480–1489.

Yoon, S., E. Yoon, H. S. Yoon, J. Kim, and C. Yoo (2022, December). Information-theoretic text hallucination reduction for video-grounded dialogue. In Y. Goldberg, Z. Kozareva, and Y. Zhang (Eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Abu Dhabi, United Arab Emirates, pp. 4182–4193. Association for Computational Linguistics.

Yoshioka, M., Y. Aoki, and Y. Suzuki (2021). Bert-based ensemble methods with data augmentation for legal textual entailment in coliee statute law task. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*, pp. 278–284.

Yu, Z., X. Liu, S. Liang, Z. Cameron, C. Xiao, and N. Zhang (2024). Don't listen to me: Understanding and exploring jailbreak prompts of large language models. *arXiv preprint arXiv:2403.17336*.

Zaheer, M., G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, et al. (2020). Big bird: Transformers for longer sequences. *Advances in neural information processing systems 33*, 17283–17297.

Zhang, C., J. Wang, Q. Zhou, T. Xu, K. Tang, H. Gui, and F. Liu (2022). A survey of automatic source code summarization. *Symmetry 14*(3), 471.

Zhang, H., S. Diao, Y. Lin, Y. Fung, Q. Lian, X. Wang, Y. Chen, H. Ji, and T. Zhang (2024, June). R-tuning: Instructing large language models to say 'I don't know'. In K. Duh, H. Gomez, and S. Bethard (Eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, Mexico City, Mexico, pp. 7113–7139. Association for Computational Linguistics.

Zhang, H., H. Song, S. Li, M. Zhou, and D. Song (2023, oct). A survey of controllable text generation using transformer-based pre-trained language models. *ACM Comput. Surv. 56*(3).

Zhang, H., C. Zhu, X. Wang, Z. Zhou, S. Hu, and L. Y. Zhang (2024). Badrobot: Jailbreaking llm-based embodied ai in the physical world. *arXiv preprint arXiv:2407.20242*.

Zhang, J., Y. Chen, C. Liu, N. Niu, and Y. Wang (2023). Empirical evaluation of chatgpt on requirements information retrieval under zero-shot setting. In *2023 International Conference on Intelligent Computing and Next Generation Networks (ICNGN)*, pp. 1–6. IEEE.

Zhang, S., S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, et al. (2022). Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zhang, S., J. Wang, G. Dong, J. Sun, Y. Zhang, and G. Pu (2024). Experimenting a new programming practice with llms. *arXiv preprint arXiv:2401.01062*.

Zhang, T., V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi (2019). Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Zheng, L., N. Guha, B. R. Anderson, P. Henderson, and D. E. Ho (2021). When does pretraining help? assessing self-supervised learning for law and the casehold dataset of 53,000+ legal holdings. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*, pp. 159–168.

Zhong, H., Z. Guo, C. Tu, C. Xiao, Z. Liu, and M. Sun (2018). Legal judgment prediction via topological learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3540–3549.

Zhong, H., C. Xiao, C. Tu, T. Zhang, Z. Liu, and M. Sun (2020a). How does nlp benefit legal system: A summary of legal artificial intelligence. *arXiv preprint arXiv:2004.12158*.

Zhong, H., C. Xiao, C. Tu, T. Zhang, Z. Liu, and M. Sun (2020b). Jec-qa: A legal-domain question answering dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Volume 34, pp. 9701–9708.

Zhou, C., G. Neubig, J. Gu, M. Diab, P. Guzman, L. Zettlemoyer, and M. Ghazvininejad (2020). Detecting hallucinated content in conditional neural sequence generation. *arXiv preprint arXiv:2011.02593*.

Zhou, X., X. Peng, T. Xie, J. Sun, C. Xu, C. Ji, and W. Zhao (2018). Benchmarking microservice systems for software engineering research. In M. Chaudron, I. Crnkovic, M. Chechik, and M. Harman (Eds.), *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, pp. 323–324. ACM.

Zhu, Y., R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler (2015, December). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*.