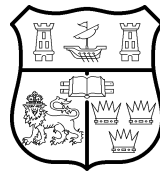


Title	Efficient adaptation of Large Language Models for digital media and government applications
Authors	Trust, Paul
Publication date	2024
Original Citation	Trust, P. 2024. Efficient adaptation of Large Language Models for digital media and government applications. PhD Thesis, University College Cork.
Type of publication	Doctoral thesis
Rights	© 2024, Paul Trust. - https://creativecommons.org/licenses/by/4.0/
Download date	2026-10-26 03:40:25
Item downloaded from	https://hdl.handle.net/10468/17932

Efficient Adaptation of Large Language Models for Digital Media and Government Applications

Paul Trust
120222601

**Thesis submitted for the degree of
Doctor of Philosophy**



NATIONAL UNIVERSITY OF IRELAND, CORK

COLLEGE OF SCIENCE, ENGINEERING AND FOOD SCIENCE

SCHOOL OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY
SFI CENTRE FOR RESEARCH TRAINING IN ADVANCED NETWORKS FOR
SUSTAINABLE SOCIETIES (ADVANCE CRT)

November 2024

Head of School: Professor Dirk Pesch

Supervisors: Dr Rosane Minghim
Dr Ahmed Zahran

Contents

List of Figures	iv
List of Tables	vi
Acknowledgements	x
Abstract	xi
1 Introduction	1
1.1 Motivation	1
1.2 Contributions	3
1.3 Publications and Submissions	4
1.4 Thesis Structure	5
2 Language Models: Background and Evolution	7
2.1 Traditional Language Modeling	9
2.2 Document Representation	10
2.2.1 One-hot Vector Encoder Representations	11
2.2.2 Count-based Word Representations	12
2.2.3 Static Dense Embeddings	13
2.2.3.1 Word2Vec	13
2.2.3.2 Word2Vec Improvements	14
2.3 Traditional Machine Learning Approaches	15
2.4 Feed-forward Neural Networks	17
2.5 Recurrent Neural Networks	18
2.6 Sequence to sequence Models	20
2.7 Transformers	22
2.8 Pre-training	26
2.8.1 Encoder-only Language Models	26
2.8.2 Decoder-only Language Models	27
2.8.3 Encoder-Decoder Language Models	29
2.9 Fine-tuning and Prompting	30
2.10 Language Model Alignment	31
2.10.1 Instruction Tuning	31
2.10.2 Preference Alignment	33
2.11 Retrieval Augmented Generation and Language Model Agents	35
2.12 Multimodality and General Purpose Language Models	36
3 Related Work	38
3.1 Weakly Supervised Learning for Predicting Economic Policy Uncertainty from News Articles	38
3.2 Citizen Feedback Analysis with Large Language Models	42
3.3 Large Language Models for Analysis of Public Sector Document Repositories	46
4 Weakly Supervised Learning for Predicting Economic Policy Uncertainty from News Articles	51
4.1 Methodology	51
4.1.1 Weakly supervised EPU classification	52
4.1.1.1 Weak Label Aggregation	53

4.1.1.2	Discriminative Model	55
4.1.1.3	Multi-label Classification for EPU Articles	56
4.1.1.4	Hierarchical Classification for EPU Articles	57
4.1.2	Data	58
4.1.3	Experimental Setup and Evaluation	59
4.2	Results and Discussion	60
4.3	Conclusion	67
5	Large Language Models for Citizen Feedback Data	68
5.1	Methodology	68
5.1.1	Data	69
5.1.2	Topic Modeling for User Feedback	70
5.1.3	Machine Learning for User Generated Feedback	72
5.1.3.1	Classification	73
5.1.3.2	Summarizing and Question Answering	75
5.1.3.3	Training and Implementation Details	77
5.1.4	Evaluation Setup and Metrics	78
5.2	Results and Discussion	78
5.2.1	Statistical and Descriptive Analysis	79
5.2.2	Topic modeling results	79
5.2.3	Classification of citizen feedback	84
5.2.3.1	Citizen Feedback Classification	85
5.2.3.2	Citizen Feedback Summarization	88
5.3	Conclusion	89
6	Large Language Models for Enhanced Interaction with Government Data Repositories	91
6.1	Methods	91
6.1.1	Tool Augmented Generation	95
6.1.2	Data	98
6.1.3	Evaluation of the Language Model Generation	100
6.1.4	Implementation and Language Models Used	103
6.2	Results and Discussion	104
6.3	Limitations	115
6.4	Conclusion	115
7	Hallucination Detection in Large Language Model outputs	117
7.1	Methodology	117
7.1.1	Hallucination and Faithfulness Scores for Hallucination Detection	118
7.1.2	Weakly Supervised Hallucination Detection	120
7.1.3	Data	121
7.1.4	Experimental Setup and Metrics	122
7.2	Results and Discussion	122
7.2.1	Hallucination and Faithfulness Scores for Hallucination Detection	123
7.2.2	Weakly Supervised Learning for Hallucination Detection	124
7.3	Conclusion	129
8	Conclusion and Future Work	130
8.1	Conclusions	130

8.2 Future Work	132
---------------------------	-----

List of Figures

2.1	Attention in RNNs [BCB15]	21
2.2	Overview of the components of the transformer architecture [VSP ⁺ 17]	23
2.3	Zero-shot prompt template for sentiment classification	30
2.4	Few-shot prompt template for sentiment classification	31
4.1	EPU label generation prompt	53
4.2	Receiver Operating Curves (ROC) curves for different models for binary EPU classification task. WS in the case means that labels generated by DP were used to train the respective model.	64
5.1	Overview of the citizen feedback analysis process, including classification, statistical and descriptive analysis, summarization, and question answering.	69
5.2	Template for classification with zero-shot prompting. The variables in brackets are replaced with actual values at inference	74
5.3	Template for classification with few-shot prompting. The variables in brackets are replaced with actual values at inference	75
5.4	A prompt template for instruction tuning language models in summarization tasks, where text in {brackets} is replaced with actual feedback, and summaries are substituted with expert-curated summaries	76
5.5	A prompt template for instruction tuning language models in question-answering tasks, where {question} is replaced with user queries and {answer} with model-generated or expert-provided responses.	76
5.6	Word plots that summarize key words in the textual feedback	79
5.7	Diagnostic values to determine the optimal number of topics	80
5.8	Distribution of words within topics	81
5.9	Topics associated with inquiries that are resolved at initiation versus those that are not.	82
5.10	Topics received from Large and Medium tax payers versus the rest of the tax payers	83
5.11	Topics handled by Kampala and Entebbe Stations versus the rest of the stations	84
5.12	Box plots showing different strategies to handle class imbalance for category classification	86
5.13	Line graphs showing how number of demonstration examples varies with GM	87
6.1	An overview pipeline that combines RAG and LLM agents	92
6.2	RAG Prompt template where the values in brackets (context and question) are replaced with actual values at inference.	95
6.3	Tool Augmented Generation Prompt for the LLM agent	97
6.4	Data Generation Prompt template, the context in brackets is replaced by actual document chunks at inference.	99
6.5	Answer Evaluation Prompt, the values in brackets, response is replaced with the actual answer generated by the model and reference_answer is replaced with the ground truth for that question	101

6.6	A bar graph showing the influence of an embedding model on RAG accuracy	107
6.7	A line graph showing the influence of the number documents in a context on generation accuracy	108
6.8	A line graph showing the influence of number of re-rank documents on generation accuracy	109
6.9	A bar graph showing the influence of retrieval method on generation accuracy	110
6.10	Line graphs showing the influence of Chunk Size and Overlap on generation accuracy	111
6.11	Confusion matrix for different modes of tool selection. The values in the confusion matrix and on the scale represent the number of documents.	113
7.1	Hallucination Score Prompt. Values in curly brackets are replaced with actual values at inference.	118
7.2	Faithfulness Score Prompt. Values in curly brackets are replaced with actual values at inference.	119
7.3	Hallucination Label Generation Prompt. Values in curly brackets are replaced with actual values at inference.	121
7.4	Box plots illustrating the accuracy of fine-tuning and prompting adaptation methods.	127
7.5	Accuracy for Model fine-tuned on Data generated by different Hallucination Label Generators	128

List of Tables

2.1	A high-level overview of techniques discussed in chapter 2.1	8
3.1	A table summarizing relevant studies on automated methods for EPU classification	39
3.2	Citizen feedback and customer support relevant works	43
3.3	Related work on applying LLMs in public sector and government applications	47
4.1	Distribution of subcategories for news articles describing EPU	59
4.2	Evaluation metrics (CK and MCC) for different label generators for the binary EPU classification task	61
4.3	Performance evaluation metrics (F1, accuracy and GM) for different label generators for the binary EPU classification task	62
4.4	Performance comparison of different models and datasets on EPU binary classification. Dataset WS in the case means that labels generated by DP were used to train the respective model	63
4.5	Performance metrics for different models on the multi-label text classification task	64
4.6	Performance evaluation of various models and hierarchical strategies on the binary EPU classification task	65
4.7	Performance evaluation of various models and hierarchical strategies on multi-label classification task	66
5.1	Class distribution of categories. Categories with less 40 observations were combined into "Rare categories"	70
5.2	Evaluation metrics: F1, GM, AUC, AUPRC of different Models for category prediction. FS in this table means Few-shot	85
5.3	ROUGE and BERTScore metrics for prompting and fine-tuning approaches across models for summarization.	88
5.4	ROUGE and BERTScore metrics for prompting and fine-tuning approaches across models for Question Answering	89
6.1	Overview of the prompts used to evaluate the groundedness, relevance and standalone of the generated questions.	102
6.2	Accuracy scores for different LLM data generators across relevance, groundedness, and standalone criteria.	104
6.3	Accuracy scores from human evaluations, LLM judge assessments, and similarity metrics: ROUGE and BLEU for different RAG systems on the IRISH and HSE datasets.	105
6.4	Accuracy scores for different multimodal LLMs evaluated by humans and different LLM judges.	114
6.5	Correlation and agreement metrics (MCC and CK) between different LLM judges and the human evaluator	114

7.1	Class distribution of test datasets labeled by human annotators for the two hallucination detection scenarios: scenario 1 (based on hallucination and faithfulness scores) and scenario 2 (based on weakly supervised learning).	121
7.3	Performance metrics of different LLM judges for hallucination detection on RAG responses	123
7.4	Comparison of different models for the hallucination detection task under zero-shot learning with prompting	125
7.5	Performance Measures for different Ensemble Techniques	125
7.6	Accuracy for different models across different generation tasks: MT, DM and PG	126

I, Paul Trust, certify that this thesis is my own work and has not been submitted for another degree at University College Cork or elsewhere.

Paul Trust

Ndabahwerize Joseph, Kyomuhangi Irene

Acknowledgements

I am profoundly grateful for the incredible support and guidance I received throughout my PhD journey, which began in Cork in January 2021. The path to completing my PhD was challenging and filled with cycles of trying, failing, and persevering. This journey has been enriched and made possible thanks to several individuals who have been instrumental in the last four years in Ireland. Firstly, I extend my heartfelt thanks to my supervisors, Rosane Minghim and Ahmed Zahran. Their exceptional mentorship from day one of my PhD journey has been invaluable. Beyond their impressive academic accomplishments, I have had the privilege of knowing them personally and witnessing their passion, dedication, and work ethics. Their guidance has been a source of inspiration, and it was an honor to work under their direction. I am fortunate to have had such amazing mentors by my side throughout this journey. I am also immensely grateful to Prof Evangelos Milos, professor at Dalhousie University and Director of Machine Learning and Networked Information Spaces (MALNIS). Participating in his group for all four years of my PhD significantly contributed to my growth and progress. The feedback from presenting my work within the group was particularly helpful. Prof Evangelos's support, including providing computing power from Compute Canada for my experiments, was crucial to my research.

Special thanks goes to Dr Omala Kizito, a lecturer at the School of Statistics and Planning at Makerere University. His mentorship since my undergraduate days and throughout my PhD has been unparalleled. Dr Kizito's encouragement, especially during challenging times, was a key factor in my persistence and progress. My colleagues at Advance CRT have also significantly contributed to my achievement. I am thankful for the collaboration and exploration I shared with my brilliant friend, Haseeb Younis. Additionally, the emotional support from my family back in Uganda has been a pillar of strength for me. Lastly, I extend my gratitude to all members and staff of the ADVANCE CRT and the School of Computer Science and Information Technology at University College Cork. This work has been supported by the Science Foundation Ireland (SFI) Center for Research Training in Advanced Networks and Future Communications (CRT ADVANCE) at University College Cork. Their collective support has been indispensable in my academic journey. This work has emanated from research conducted with the financial support of the Science Foundation Ireland under Grant number 18/CRT/6222.

Paul Trust
Cork, November 2024

Abstract

The digital transformation has greatly increased the amount of data, particularly text generated across various fields, including the public sector and digital media. Information such as political campaigns, media reports, citizen feedback, and press releases is now commonly shared on digital platforms. Politicians, government agencies, businesses, and citizens use these platforms to express their views, strategies, goals, and policies. Analyzing this data can provide valuable insights into public opinion, ongoing policy discussions, business strategies, and socio-political dynamics.

However, the sheer volume of data makes traditional manual analysis or early computational methods impractical, highlighting the need for more efficient automated approaches from Natural Language Processing (NLP), particularly Large Language Models (LLMs), to manage, analyze, and summarize this information. In the course of this work, document-based learning has seen a landmark advance with the progress of generative Artificial Intelligence (AI) and the availability of engines and models that are revolutionizing NLP.

Despite significant adoption and investment in the private sector, academia, and high-resource fields, LLMs are less utilized in low-resource fields and the public sector due to constraints such as lack of labeled data, and insufficient budget allocation for Machine Learning (ML) infrastructure and training. In this Thesis, which was developed during a transition time of fast development of LLMs, we study ways of adapting the most up to date models to novel scenarios in order to both achieve efficiency and understanding of how to adapt LLMs to applications.

To achieve this, this Thesis adapts LLMs to applications in the public sector and digital media. The main approaches developed include: applying weak supervision by leveraging synthetic labels generated by other LLMs to fine-tune models for classifying news articles related to Economic Policy Uncertainty (EPU); proposing LLM-based methods for classifying citizen feedback into different categories, as well as for summarization and question answering of citizen feedback; adapting LLMs for automating the handling and navigation of public documents by incorporating strategies such as Retrieval Augmented Generation (RAG), LLM agents; and developing techniques for hallucination detection in these domains.

Chapter 1

Introduction

1.1 Motivation

The digital transformation has led to a substantial increase in data generation across various fields, much of which is unstructured and largely textual [SHB⁺24]. Text is fundamental to human communication, and it is widely used by politicians and government bodies for activities such as policy discussions, campaigns, legislation, press releases, and peace negotiation, to name a few [JPV⁺21]. In the context of government, citizens frequently share their views or feedback on digital platforms, primarily through text, and many official government processes are documented in textual form [KEJM20]. Analyzing these texts can enhance our understanding of citizen feedback, track policy trends, assess government action impacts, and improve access to extensive government document repositories [DVCA23].

In economics and business, text is frequently used in product reviews, public company filings, economic reports, and business news articles, providing opportunities to gain insights into consumer behavior, asset price movements, and stock market trends [GKT19]. Similarly, in sociology, text analysis is an effective method for discovering communication patterns that reflect social relationships, power dynamics, and collective behaviors. By examining text in various communication forms, such as social media posts, interviews, or political speeches, researchers can better understand group dynamics, trends, public discourse, and the evolution of social roles [XCC⁺20].

The value of text data in understanding human behavior, society, and thought has in the past led to an extensive manual analysis by social scientists, businesses, academia, and governments. However, the digital revolution has caused the volume of text data to grow exponentially, far exceeding the capacity of traditional manual methods [GS13]. To address this, automated techniques from Natural Language Processing are now used

to manage the increasing data volume.

NLP has received increasing attention in recent years, particularly with the development of LLMs. The release of OpenAI's Chat Generative Pre-trained Transformer (ChatGPT) showcased their capabilities, driving widespread interest. These models excel in tasks such as question answering, summarization, text classification, and machine translation, achieving performance close to human levels [Ope23, TMS⁺23]. While the private sector has rapidly embraced and invested in NLP, its adoption in the public sector and low-resource settings remains limited. This is largely due to the need for fine-tuning or additional training with domain-specific data, which is often scarce and expensive to acquire in the areas of interest in this Thesis [DPHZ23].

This Thesis studies the employment of automated NLP methods, particularly LLMs, for analyzing digital media and unstructured data in the public sector, focusing on the efficient adaptation of LLMs in these contexts. In order to support applications with considerably less resources in terms of data and computing power than those of big companies, many questions on current LLMs need to be asked and answered, some of which this work aims to tackle. In this context, the main research questions approached in this Thesis are:

1. How can the latest advancements in LLMs be effectively leveraged in fields with constrained data availability and limited computational resources?
2. Can we use cost-effective labeling sources, like synthetic labels, to fine-tune LLMs for downstream tasks instead of relying on human labels?
3. How can LLMs enhance the analysis and use of unstructured citizen feedback in government organizations?
4. Can LLMs be used as language assistants to process and streamline questions from government documents and websites?
5. What are the main issues faced by AI professionals in adapting current models to particular applications reliably?

To address these questions, this Thesis focuses on three applications. The first application utilizes LLMs for economic monitoring by classifying news articles related to EPU, improving upon Baker et al.'s (2016) keyword-based approach, which often results in a high number of false positives [BBD16]. To enhance cost-effectiveness, this Thesis proposes a weakly supervised learning approach where training labels are generated by other LLMs instead of human annotators. Additionally, it proposes methods for identifying different types of EPU in news articles using multi-label and hierarchical classification.

The second application focuses on analyzing citizen feedback, much of which is unstructured text data that often remains unanalyzed despite its potential value. To address this, this Thesis proposes methods for analyzing real citizen interactions in a government organization, initially using topic models and later integrating automated approaches for classification, summarization, and question answering based on LLMs. These techniques improve the efficiency of analysis and responses to citizen concerns.

The third application involves adapting LLMs for summarizing and processing government documents, including press releases, policies, and procedural guidelines. Although these documents are publicly accessible, navigating them to extract relevant information can be challenging. This Thesis proposes leveraging LLMs to automate document retrieval and summarization through RAG and LLM agents, reducing hallucination rates to ensure accuracy in these applications. Additionally, it develops methods for detecting hallucinations in LLM-generated responses to prevent misinformation and implements intelligent chatbots using RAG and LLM agents to facilitate citizen inquiries about government documents. These applications are summarized in the next section and discussed in detail in the subsequent chapters of the Thesis.

1.2 Contributions

This Thesis focuses on the efficient and reliable adaptation of current LLMs for applications in digital media and the public sector. The contributions, which are detailed in the Thesis chapters, are summarized below.

- The first contribution of this Thesis is the development of a cost-effective and scalable approach for classifying news articles related to EPU using LLMs fine-tuned with weak supervision techniques. This method reduces reliance on costly and time-consuming human-labeled data by generating synthetic labels from other LLMs. By efficiently classifying news articles, it improves accuracy compared to baseline methods. Additionally, beyond binary classification, this Thesis proposes methods for identifying different types of EPU in news articles using multi-label and hierarchical classification. This contribution is discussed in detail in chapter 4.
- The second contribution of this Thesis is the development and evaluation of methods using LLMs to analyze unstructured citizen feedback data, demonstrated through a case study from the Uganda Revenue Authority (URA). It introduces a dataset of real citizen feedback, which is valuable for building and testing ML models in the public sector, where labeled datasets are often scarce. This work also adapts and evaluates LLMs for classifying citizen feedback into multiple cate-

gories. Additionally, this Thesis assesses both prompting and fine-tuning approaches using generative LLMs for question answering and summarization of citizen feedback. This contribution is discussed in detail in chapter 5.

- The third contribution of this Thesis is the development and evaluation of generative LLMs for the public sector, focusing on their use as language assistants to efficiently answer common questions from citizens and government officials who would otherwise spend considerable time searching through documents for information. This work involves developing and extensively evaluating RAG systems and their components for government document processing. Additionally, LLM agents are adapted to integrate real-time information from online government sources, ensuring responses are based on the most up-to-date data. This Thesis also assesses the effectiveness of multimodal LLMs in handling documents that contain both text and images. To support training and evaluation in this domain, multiple datasets are created, including text and vision instruction tuning datasets, and datasets designed for evaluating LLM agents and RAG systems. This contribution is detailed in chapter 6.
- The fourth contribution of this Thesis is the development and evaluation of methods for detecting hallucinations in LLM outputs. The first proposed approach utilizes hallucination and faithfulness scores to identify hallucinations in responses from RAG systems and LLM agents using prompt-based techniques. Additionally, this Thesis proposes a weakly supervised distillation learning approach that enhances prompt-based hallucination classifiers by fine-tuning smaller LLMs using labels generated by more capable LLMs. This method reduces inference costs and processing time while serving as a cost-effective alternative to human annotation, which can be expensive and difficult to obtain. This contribution is detailed in chapter 7.

1.3 Publications and Submissions

Certain aspects of the work presented in this Thesis are discussed in the following manuscripts and publications.

- **Paul Trust**, Ahmed Zahran, and Rosane Minghim: *Understanding the Influence of News on Society Decision Making: Application to Economic Policy Uncertainty*, Neural Computing and Applications, 35:14929–14945, 2023.

Paul Trust and Rosane Minghim: *Text Classification in the Age of Large Language Models*, Machine Learning and Knowledge Extraction, 6:2688-2721, 2024.

- **Paul Trust**, Rosane Minghim: *Weakly Supervised Distillation Learning for Hallucination Detection*, In preparation for submission.
- **Paul Trust**, Kizito Omala, and Rosane Minghim: *Leveraging Topic Modeling and Large Language Models to Enhance Analysis of Government Citizen Feedback*, In preparation for submission.
- **Paul Trust**, Kizito Omala, and Rosane Minghim: *Large Language Models for Enhanced Interaction with Government Data Repositories*, In preparation for submission.

The following publications originate from the research conducted during this PhD study but are not presented in this Thesis.

- **Paul Trust**, and Rosane Minghim: *Query-Focused Submodular Demonstration Selection for In-Context Learning in Large Language Models*, In Proceedings of the 2023 31st Irish Conference on Artificial Intelligence and Cognitive Science (AICS), pages 1-8, 2023.
- **Paul Trust**, Provia Kadusabe, Haseeb Younis, Rosane Minghim, and Evangelos Milios, Ahmed Zahran: *Unsupervised Natural Language Premise Selection in Mathematical Text using Sentence-MPNet*, In Proceedings of TextGraphs-16: Graph-based Methods for Natural Language Processing COLING, pages 119–123, 2022.
- **Paul Trust**, Rosane Minghim, Evangelos Milios, and Kadusabe Provia: *Gated Graph Neural Networks for Causal Event Classification from Social-Political News Articles*, In Proceedings of the 5th Workshop on Challenges and Applications of Automated Extraction of Socio-political Events from Text (CASE), EMNLP, pages 85–90, 2022.
- Haseeb Younis, **Paul Trust**, and Rosane Minghim: *Understanding High Dimensional Spaces through Visual Means Employing Multidimensional Projections*, International Journal on Engineering, Science and Technology (IJonEST), 4:170-179, 2022.

1.4 Thesis Structure

Chapter 1 presents the motivation for this Thesis, emphasizing the importance of NLP and LLM in managing large datasets, particularly in the public sector and in digital media. It discusses our contributions, highlighting the relevance of each method. This chapter also establishes a foundation for the Thesis by describing the research background, objectives, and the organization of the upcoming chapters.

Chapter 2 reviews the evolution of language models from simple n-grams to advanced systems like Generative Pre-trained Transformer-4 (GPT-4). This chapter covers key topics relevant to this Thesis, including traditional methods, word representations, deep learning architectures, transformers, instruction tuning and preference alignment.

Chapter 3 reviews the literature to identify gaps and outline tasks and proposed methods in this Thesis. It provides the rationale for our approaches and places our work within the context of the current State of The Art (SoTA).

Chapter 4 presents a cost-effective and scalable approach for classifying news articles concerning EPU. This method employs pre-trained LLMs that are fine-tuned using weak supervision techniques and also discusses multi-label and hierarchical classification to detect different types of EPU.

Chapter 5 presents and assesses advanced LLM-based methods for analyzing unstructured citizen feedback data. The proposed approaches involve adapting LLM techniques for classification, summarization, and question answering.

Chapter 6 presents and assesses conversational generative LLMs for public sector use, utilizing RAG and LLM agents to efficiently respond to inquiries. It also proposes and evaluates multimodal LLMs that combine text and images.

Chapter 7 introduces and assesses hallucination detection techniques, incorporating both hallucination and faithfulness scores, alongside a weakly supervised distillation learning method.

Chapter 8 summarizes the key findings from the experiments and results, highlighting the conclusions drawn from this Thesis. Additionally, it identifies potential directions for future work, inspired by the studies and methodologies conducted throughout this Thesis.

In the next chapters, starting with chapter 2, the contents of this Thesis are discussed in detail.

Chapter 2

Language Models: Background and Evolution

This chapter reviews the evolution of language models, beginning with simple n-gram approaches and progressing to advanced systems such as GPT-4, which were rapidly evolving during the period in which the work conducted in this Thesis was done. The discussion of these methods is essential not only because several are applied throughout the Thesis, but also to provide context for the choices made in the design and implementation of the approaches used.

Section 2.1 covers traditional language modeling techniques, followed by section 2.2, which discusses different word representations. Traditional ML methods are presented in section 2.3, while sections 2.4 and 2.5 cover deep learning approaches based on feed-forward neural networks and Recurrent Neural Network (RNN) architectures and their variants. Section 2.7 discusses attention mechanisms and the transformer architecture that underlies many recent advances in NLP.

Section 2.8 discusses pre-training techniques used to equip LLMs with parametric knowledge, followed by section 2.9, which covers fine-tuning and prompting strategies. Section 2.10 focuses on how language models are adapted to reflect human preferences using post fine-tuning methods such as Reinforcement Learning from Human Feedback (RLHF) and instruction tuning. Section 2.11 presents (RAG) and the use of LLM agents, while section 2.12 discusses multimodal LLMs and general purpose AI. A summary of the techniques discussed in this chapter is provided in Table 2.1.

Table 2.1: A high-level overview of techniques discussed in chapter 2.1

Category	Technique	Main idea	Models
Representation	Count-based 2.2.2	Word frequency for document representation	BoW, TF-IDF [Aiz03]
	Static embeddings 2.2.3	Fixed dense word vectors capturing semantics	Word2Vec [MSC ⁺ 13], Doc2Vec [LM14], Glove [PSM14], Fasttext [BGJM17]
	Contextualized embeddings 2.8	Context-sensitive embeddings for words	BERT [DCLT19], RoBERTa [LOG ⁺ 19], SBERT [RG19]
Traditional NLP	Traditional ML 2.3	Classical ML methods	Logistic Regression [Bis06], SVM [CV95], Decision Trees [Qui86], Random Forests [Bre01]
	Deep Learning 2.5	Neural networks for NLP	RNN [GBC16], GRU [JZS15], LSTM [HS97]
LLMs	Transformer 2.7	Foundational architecture on which modern LLM are based.	Transformer [VSP ⁺ 17]
	Encoder-only 2.8.1	Masked language modeling pre-training	BERT [DCLT19], RoBERTa [LOG ⁺ 19], ELECTRA [CLLM20], XLNet [YDY ⁺ 19]
	Decoder-only 2.8.2	Next-token prediction pre-training	GPT-2, GPT-3 [BMR ⁺ 20], GPT-4 [Ope23], BLOOM [SFA ⁺ 22], OPT [ZRG ⁺ 22]
	Encoder-decoder 2.8.3	Encoder-decoder pre-training	T5 [RSR ⁺ 20], BART [LLG ⁺ 20]
	Multimodal LLMs 2.12	Multiple data modalities (text, image, video, audio)	GPT-4-Vision, LLAMA3-Vision, Gemini
LLM Adaptation	Prompting 2.9	Adaption via language instructions	Few-shot [DLD ⁺ 24], CoT [WWS ⁺ 22], ToT [YYZ ⁺ 24b]
	Fine-tuning 2.9	Domain-specific adaptation	Pre-trained LLMs [DCLT19]
	Instruction tuning 2.10.1	Instruction following	Falcon [AAA ⁺ 23], Mistral [JSM ⁺ 23], Dolly [CHM ⁺ 23]
	Preference Alignment 2.10.2	Human preference alignment	LLAMA [GTDS ⁺ 24], InstructGPT [ZSW ⁺ 19], Claude [Ant24]
	RAG 2.11	External knowledge at inference from vector databases	Embeddings, LLMs [XCG ⁺ 24]
	LLM Agents 2.11	External tools and structured reasoning	ReACT [YZY ⁺ 23]

2.1 Traditional Language Modeling

Language modeling is a fundamental task in NLP and involves predicting the probability of word sequences in a language. These models act as probability distributions over sequences of tokens, enabling them to differentiate between more and less likely sequences based on their likelihood of occurrence. This capability allows Language Models (LMs) to understand the grammatical and linguistic complexities of languages. An LM assigns a probability $p(x_1, x_2, \dots, x_n)$ to each sequence of tokens $x_1, x_2, \dots, x_n$ from a vocabulary V .

This probability serves as an indicator of the sequence's plausibility. The primary goal is to assess the likelihood of a sequence of n words, treated as a joint event, where each x_i is a random variable. The joint probability $p(x_1, x_2, \dots, x_n)$ can be decomposed into a series of conditional probabilities using chain rule shown in Equation 2.1.

$$p(x_1, x_2, \dots, x_n) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1}) \quad (2.1)$$

In this model, $p(x_i | x_1, \dots, x_{i-1})$ represents the probability of the word x_i appearing after the preceding sequence of words.

Earlier approaches in NLP calculated the probability $p(x_1, x_2, \dots, x_n)$ using simpler models known as n-gram LMs. An n-gram represents a sequence of n consecutive words. For instance, in the sentence: "Paul is going to school", the examples of n-grams include: the uni-gram ("Paul"), bi-gram ("Paul is"), and tri-gram ("Paul is going").

The goal of an n-gram LM is to calculate the probability of a word x_i given its preceding context. For example, the probability of the word: "school" following the phrase: "Paul is going to" is estimated using relative frequency counts shown in Equation 2.2. This involves analyzing a large corpus and dividing the number of occurrences of the phrase: "Paul is going to school" by the number of times: "Paul is going to" appears.

$$p(\text{school} | \text{Paul is going to}) = \frac{\text{Count}(\text{Paul is going to school})}{\text{Count}(\text{Paul is going to})} \quad (2.2)$$

As sequences become longer, counting the frequency of each word following every possible long sequence becomes increasingly challenging. This difficulty arises because language is inherently creative, and many specific contexts may have never occurred together before.

Instead of computing the probability of a word given the entire history, it can be approximated by considering only the last few words. For instance, the bi-gram model approximates the probability of a word given all previous words $p(x_n | x_{1:n-1})$ by using

only the conditional probability of the preceding word $p(x_n|x_{n-1})$. In other words, instead of calculating: $p(\text{school}|\text{Paul is going to})$ in our example, we approximate it with: $p(\text{school}|\text{to})$. By doing this, we are making a Markov assumption. Markov models are a type of probabilistic model that rely on the assumption that the likelihood of a future event can be determined based solely on recent events, without needing to account for distant past occurrences.

This can be extended from the bi-gram model, which considers one previous word, to a tri-gram model that considers two previous words, and further to an n-gram model that looks $n - 1$ words into the past. For an n-gram, the probability of a word given its entire context is approximated using Equation 2.3 where N_g denotes the n-gram size.

$$p(x_n|x_{1:n-1}) \approx p(x_n|x_{n-N_g+1:n-1}) \quad (2.3)$$

For the bi-gram assumption where $N_g = 2$, Equation 2.3 simplifies to Equation 2.4.

$$p(x_n|x_{1:n-1}) \approx p(x_n|x_{n-1}) \quad (2.4)$$

The probabilities of bi-grams or n-grams can be estimated by obtaining counts from the corpus and normalizing them to range between 0 and 1.

Although n-gram models are important in language modeling, they encounter significant difficulties, especially with data sparsity and memory usage. As the length of the n-grams increases, their occurrence in the training data diminishes, which impacts the model's ability to accurately predict infrequent sequences. Furthermore, n-gram models often require considerable memory to store the frequency counts of all observed sequences, and this need increases as the corpus expands. These challenges have led to the development of more advanced NLP methods designed to overcome these problems, which will be discussed next.

2.2 Document Representation

In language modeling and various other ML applications, a crucial pre-processing step involves converting text into a numerical format, as ML models cannot directly interpret raw text. The choice of word representation is crucial, as it significantly influences the model's effectiveness on downstream tasks. Word representations were initially predominantly sparse and count-based, focusing on word frequencies, sometimes with normalization. These count-based representations are high-dimensional and often fail to capture the semantic relationships between words and documents effectively. Recently, there has been a shift towards dense, real-valued word representations, which

offer improved semantic representation and understanding [JM09].

This shift towards dense, semantic, and contextualized word representations is rooted in linguistic theories. For instance, J.R. Firth (1957) emphasized the importance of context in understanding word meanings, encapsulated in his famous assertion: "You shall know a word by the company it keeps" [Fir57]. This perspective also aligns with Ludwig Wittgenstein's view that a word's meaning is tied to its usage in language [BM19] and Zellig Harris's emphasis on the distributional properties of language for thorough description [Har54]. These theories highlight the importance of context in defining meaning and have inspired the development of modern word representations that are both semantically rich and context-sensitive. The word representations derived from these approaches are known as embeddings and can be either static, assigning fixed vectors to each word, or contextualized, representing words based on the context they are used.

2.2.1 One-hot Vector Encoder Representations

One-hot vector encoding [JM09] is the most basic method for representing words and is the basis for more sophisticated word representations. In this approach, every word in the vocabulary is assigned a distinct vector in which one element is set to 1 to indicate the word's presence, while all other elements are set to 0. The vector's length matches the vocabulary size, so if there are n words in the vocabulary, each word is represented by a vector with n dimensions. For example, in a vocabulary consisting of the words: {novel, book, magazine}, the one-hot encoded vector for "novel", would be $[1, 0, 0]$, for "book", it would be $[0, 1, 0]$, and for "magazine", it would be $[0, 0, 1]$. Mathematically, if x_i is the i -th word in the vocabulary, its one-hot vector $\mathbf{v}_i$ is defined in Equation 2.5.

$$\mathbf{v}_i[j] = \begin{cases} 1 & \text{if } j = i, \\ 0 & \text{otherwise} \end{cases} \quad (2.5)$$

where j ranges from 1 to n words in the vocabulary. This representation, while straightforward and easy to implement, results in very sparse and high-dimensional vectors, especially in large vocabularies. It also does not capture any semantic relationships between words, as each word is equidistant from every other word in this space. For example, the cosine similarity between "novel" $[1, 0, 0]$ and "magazine" $[0, 0, 1]$ would be 0, indicating they are completely unrelated, which is inaccurate.

2.2.2 Count-based Word Representations

This section reviews improved representations over one-hot vectors that represent words by their frequency in a document. The basic example of these approaches is the Bag of words (BoW) representation [JM09]. In the BoW method, a text (such as a sentence or document) is represented as an unordered set of words, where word order and grammar are disregarded, but word multiplicity is preserved. Each document is transformed into a fixed-length vector, where the length corresponds to the vocabulary size, and each element reflects the Term Frequency (TF (x_i, d)), which is frequency of i -th word x_i in the document d .

While the BoW model improves representation over one-hot vector encoding, it tends to overemphasize common words that frequently appear across various texts (stop words) but does little to distinguish between different documents.

Term Frequency Inverse Document Frequency (TF-IDF) [Aiz03] is an improvement of BoW that addresses some of its limitations by assigning weights to words based on their distribution across a corpus. Words that appear in all documents are given lower weights, while words that appear more frequently in one document and less frequently in others are assigned higher weights. This weighting scheme helps to minimize the impact of commonly occurring words that appear in all documents from having a significant weight like in the BoW model.

TF-IDF accomplishes this by combining two measures: TF and Inverse Document Frequency (IDF). IDF is calculated as the logarithm of the number of documents in the corpus divided by the number of documents where the specific word appears shown in Equation 2.6.

$$\text{IDF}(x_i, N_d) = \log \left(\frac{\text{Total number of documents } (N_d)}{\text{Number of documents with word } x_i} \right) \quad (2.6)$$

The TF-IDF metric is then calculated as the product of the two terms as shown in Equation 2.7

$$\text{TF-IDF}(x_i, d, N_d) = \text{TF}(x_i, d) \times \text{IDF}(x_i, N_d) \quad (2.7)$$

Though TF-IDF reduces the prominence of common terms and highlights distinctive features of documents, it has some limitations. It cannot capture word context or semantic relationships, struggles with words that have multiple meanings (polysemy), and assumes that rarer words are more important. Moreover, the method often produces high-dimensional representations since the matrix size is equal to the vocabulary size. These limitations have prompted the development of more sophisticated models like Word to Vector (Word2Vec) [MCCD13], which offer dense, real-valued vector representations that capture deeper linguistic relationships. These representations are

discussed next.

2.2.3 Static Dense Embeddings

This section describes static dense embeddings starting with Word2Vec, which provide a more compact and efficient form of representation with dimensions typically ranging from 50 to 5000. Unlike sparse vectors that are predominantly filled with zeros, dense embeddings consist of real-valued numbers, including negative values, and though less interpretable, they have demonstrated superior performance in various NLP tasks.

2.2.3.1 Word2Vec

Word2Vec, developed by Mikolov et al. (2013) [MSC⁺13], uses large text datasets to generate word embeddings that capture both syntactic and semantic relationships. It represents words in a dense vector space, placing semantically related words and those that frequently appear together in similar contexts close to each other. One of the model's strengths is its ability to perform vector arithmetic to reflect semantic concepts. For example, the Equation: $vector("King") - vector("Man") + vector("Woman")$ typically results in a vector close to "Queen", demonstrating how Word2Vec captures complex linguistic relationships within a multi-dimensional space.

The Word2Vec model represents each word in a given vocabulary as a low-dimensional vector, much smaller than the vocabulary size. It learns these vector values by analyzing word distributions within a short context window, typically around 1 to 4 words. Word2Vec comes in two variants based on pre-training objectives: Continuous Bag of Words (CBOW) and Skip-Gram. The Skip-Gram model predicts the surrounding context words given a target (center) word within a defined context window. Conversely, the CBOW model predicts the target (center) word using the surrounding context words, essentially performing the inverse operation of Skip-Gram.

Since the mathematical formulations of both variants are highly similar, only the Skip-Gram model is described here. It maximizes the log-likelihood of accurately predicting context words given the target word for a sequence of words: $x_1, x_2, \dots, x_n$. The objective function for this process is shown in Equation 2.8.

$$\text{Objective} = \frac{1}{n} \sum_{t=1}^n \sum_{-C_w \leq j \leq C_w, j \neq 0} \log p(x_{t+j}|x_t) \quad (2.8)$$

In Equation 2.8, x_t denotes the target word, x_{t+j} are the context words within a C_w -word window around the target, and n represents the total number of words in the corpus. The probability $p(x_{t+j}|x_t)$ is calculated using the softmax function. This structure

allows the Skip-Gram model to effectively learn word embeddings by relating target words to their respective contexts in the corpus.

Both CBOW and Skip-Gram models use negative sampling to efficiently approximate softmax calculations, particularly when dealing with large vocabularies. Word2Vec has been successfully pre-trained on large datasets like Google News and Twitter, which contain billions of words. The resulting word embedding matrix vectors from this pre-training can be used as initial word representations for various NLP tasks, greatly enhancing the performance of those downstream applications.

2.2.3.2 Word2Vec Improvements

Word2Vec is effective in generating word vectors but often struggles to capture the full semantic meaning and structure of longer texts. To overcome this limitation, Document to Vector (Doc2Vec) [LM14] extends Word2Vec by creating vector representations for entire sentences or documents rather than just individual words. This allows it to generate fixed-length vectors from variable-length texts, such as sentences, paragraphs, and full documents. In the Doc2Vec model, both words and paragraphs are assigned unique vectors. The paragraph vectors are combined with word vectors and averaged to predict the next word in a given context, incorporating information from both individual words and the overall paragraph.

Beyond Doc2Vec, several improvements to Word2Vec have been introduced, including Global Vectors for Word Representation (GloVe) [PSM14] and FastText [BGJM17]. While Word2Vec captures only local context, GloVe enhances word embeddings by incorporating both global and local contexts using a global log-bilinear regression model that combines matrix factorization with local context window methods. However, both Word2Vec and GloVe struggle with word morphology, as they assign distinct vectors to each word, making them less effective for languages with large vocabularies and rare words. FastText [BGJM17] addresses this limitation by representing words as collections of character n-grams, allowing it to generate embeddings for unseen words and making it more efficient for large corpora.

Although models like Word2Vec, GloVe, and FastText have their advantages, they share a significant limitation: they produce the same vector for a word, regardless of its meaning in different contexts. For example, a word "fly" in the sentences: "The fly bit me" and "Peter and John fly the airplane" would have identical vectors, missing the context-specific meanings. This limitation has prompted the development of contextualized word embeddings, which create dynamic embeddings based on the word's usage context, that are often extracted from pre-trained LLMs described in section 2.7.

2.3 Traditional Machine Learning Approaches

Before the widespread adoption of neural networks in NLP, most NLP tasks primarily used statistical methods and traditional ML techniques. These approaches heavily rely on feature-based learning to perform various NLP tasks. The development of traditional ML methods in NLP spans from simpler linear models to more advanced tree-based and ensemble methods, which are discussed next [Bis06, MRS08, JM09].

Logistic Regression: Logistic regression [Bis06] is one of the earliest ML models used in NLP, particularly for binary and multiclass classification tasks. Unlike linear regression, which predicts continuous values, logistic regression predicts the probability of a discrete class label y given an input feature vector $\mathbf{x}$. For binary classification, it applies the sigmoid function, which maps any real-valued input to a probability between 0 and 1, as defined in Equation 2.9.

$$P(y|\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w} \cdot \mathbf{x} + b)}} \quad (2.9)$$

Here, $\mathbf{w}$ is the weight vector, $\mathbf{x}$ represents the input feature vector, and b is the bias term. For multiclass classification, logistic regression applies the softmax function to produce a probability distribution over multiple classes. Logistic regression is computationally efficient, interpretable, and works well when there is a linear decision boundary between classes.

Support Vector Machines (SVM): SVM [CV95] is an ML algorithm designed for classification by identifying an optimal hyperplane that maximizes the margin between different classes. This margin represents the distance between the closest data points from each class, known as support vectors. SVMs are particularly effective in high-dimensional feature spaces and can be applied to both linear and non-linear classification problems. For a dataset with feature vectors $\mathbf{x}$ and corresponding labels y , the objective of an SVM is to determine a hyperplane that best separates the classes by maximizing the margin. The optimization problem is formulated as shown in Equation 2.10.

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{subject to } y_i(\mathbf{w}^T \cdot \mathbf{x}_i + b) \geq 1, \forall i \quad (2.10)$$

where $\mathbf{w}$ is the weight vector and b is the bias term. To handle non-linearly separable data, SVMs utilize kernel functions like Radial Basis Function (RBF) kernel [Joa98], which transform input data into a higher-dimensional space where a linear separation is possible.

Decision Trees: Decision trees [Qui86] perform classification by constructing a hierarchical structure where each internal node represents a feature, branches define decision

rules, and leaf nodes correspond to class labels. The classification process begins at the root node, where the input data is evaluated against a feature-based condition. Based on the outcome, the data follows a specific branch to the next node, where another decision is made. This process continues until the data reaches a leaf node, which assigns the final class label. To determine the best feature for splitting at each node, decision trees use criteria such as Gini impurity, which measures how well a split separates the classes. The Gini impurity is defined by Equation 2.11.

$$\text{Gini Impurity} = 1 - \sum_{i=1}^{N_c} p_i^2 \quad (2.11)$$

where p_i represents the proportion of samples belonging to class i , and N_c is the total number of classes. A lower Gini impurity value indicates that a node contains more homogeneous class distributions, leading to better classification accuracy.

Random Forests: Random forests [Bre01] perform classification by combining multiple decision trees to improve generalization and reduce overfitting. Instead of relying on a single decision tree, a random forest builds an ensemble of trees, each trained on a randomly sampled subset of the training data with replacement (bootstrapping). During classification, each tree independently predicts a class label, and the final decision is made through majority voting defined in Equation 2.12.

$$\hat{y} = \text{mode}(\{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_{N_{tr}}\}) \quad (2.12)$$

where $\hat{y}_i$ denotes the predicted class label from the i -th tree, and N_{tr} represents the total number of trees in the forest.

Gradient Boosting Machines (GBM): GBM [Fri01] performs classification by iteratively building models that correct the errors of previous ones. Unlike random forests, which train models independently, GBM constructs an ensemble of weak learners in a sequential manner, optimizing a loss function using gradient descent. At each iteration i , the model is updated using Equation 2.13.

$$F_i(\mathbf{x}) = F_{i-1}(\mathbf{x}) + \gamma h_i(\mathbf{x}) \quad (2.13)$$

Here, $\mathbf{x}$ denotes the input feature vector, $F_i(\mathbf{x})$ is the prediction of the model at iteration i , and $F_{i-1}(\mathbf{x})$ is the prediction from the previous iteration. The term $h_i(\mathbf{x})$ represents the weak learner trained to approximate the residual errors, typically by fitting the negative gradient of the loss function, and γ is the learning rate that controls the contribution of the new learner to the overall model. This iterative process enables

the GBM model to progressively reduce classification error. Variants such as XGBoost, LightGBM, and CatBoost are widely used in NLP tasks.

Traditional ML models discussed so far contributed significantly to early NLP developments but are limited by their heavy reliance on manual feature engineering. Deep learning architectures, including feed-forward networks, RNNs, and transformers, perform joint feature engineering with learning, reducing the reliance on manually crafted features by learning text representations directly from raw data [VSP⁺17]. These approaches are discussed next.

2.4 Feed-forward Neural Networks

Feed-forward neural networks [GBC16] are fundamental building blocks to the development of advanced neural architectures. Originally inspired by a simplified model of human neurons, they became essential in developing NLP systems in the early days. These networks are structured hierarchically with an input layer, several hidden layers, and an output layer with information flowing uni-directionally without loops. Neurons within these networks process real-valued numerical inputs by performing a weighted sum of the inputs plus a bias term, a computation that can be represented mathematically in Equation 2.14.

$$z = \sum_i (w_i x_i) + b \quad (2.14)$$

$$z = \mathbf{w} \cdot \mathbf{x} + b \quad (2.15)$$

Where x_i represents the i -th input, w_i denotes the corresponding weight for the i -th input, b is the bias, $\mathbf{x}$ is the input vector, and $\mathbf{w}$ is the weight vector.

In neural networks, the linear output z is not used directly. Instead, it is passed through a non-linear function σ known as the activation function. Early neural networks commonly used the sigmoid function as their non-linear activation function given by Equation 2.16.

$$f(z) = \sigma(z) = \frac{1}{1 + e^{-z}} \quad (2.16)$$

In some respects, the $\tanh$ function, defined in Equation 2.17, is preferred to the sigmoid function due to its output range from -1 to $+1$, smooth differentiability, and improved ability to normalize extreme values toward the mean [GBC16].

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (2.17)$$

Despite its advantages, the $\tanh$ function, like the sigmoid, can lead to saturation with

large input values, causing outputs to approach 1 and derivatives to near 0. This leads to the vanishing gradient problem during training, where the error signal becomes significantly reduced as small gradients are propagated through the network. To address this, many networks use the Rectified Linear Unit (ReLU) and its variants, which mitigate or eliminate these issues shown in Equation 2.18.

$$f(z) = \text{ReLU}(z) = \max(0, z) \quad (2.18)$$

The ReLU function sets negative inputs to zero while maintaining a constant derivative of 1 for positive values, thereby avoiding the saturation problem typical of sigmoid and $\tanh$ functions. This characteristic significantly enhances the convergence speed of learning algorithms and prevents the gradients from vanishing.

Once the activation function is selected, the next step is training the neural network for a specific task. Back propagation is a commonly used algorithm for training feed-forward neural networks, utilizing the chain rule from calculus to efficiently compute gradients and adjust the network's weights and biases. This process minimizes the difference between the network's predicted outputs and the actual target values.

Feed-forward neural networks have limitations in NLP applications due to the absence of parameter sharing and their inability to consider the sequential nature of text. Since each node learns parameters independently, the model requires a large number of parameters to be trained. These challenges have driven the development of more advanced architectures, such as Recurrent Neural Network (RNN) and transformers, which are designed to better capture the structure and dependencies in text, and are discussed next.

2.5 Recurrent Neural Networks

RNNs are designed to process sequential data by utilizing cycles within their connections to maintain a form of memory. These networks save the output from one layer and reuse it as input for the next, creating a loop that allows for the dynamic handling of temporal sequences. This cyclical nature means that the current state depends on both the previous state and the current input, enabling the network to remember past information. RNNs are particularly effective for modeling text because each word or character in a sequence can influence the subsequent ones. By sharing parameters across hidden layers, RNNs manage to learn efficiently with fewer parameters [GBC16].

In an RNN, the update of the hidden state at a given time step t represented by h_t , relies on the previous hidden state h_{t-1} and the current input x_t . This relationship is

mathematically represented by the Equation 2.19.

$$h_t = \sigma(W_{hh}h_{t-1} + W_{hx}x_t) \quad (2.19)$$

The matrices W_{hh} and W_{hx} represent the weights applied to the previous hidden state and current input respectively, while σ is a non-linear activation function like tanh or ReLU. The predicted output at each time step, $\hat{y}_t$, is computed using Equation 2.20.

$$\hat{y}_t = \text{softmax}(Uh_t) \quad (2.20)$$

The hidden state h_t is multiplied by the weight matrix U and then passed through a softmax function, transforming the weighted hidden state into a probability distribution over the possible outputs.

Although RNNs are designed to use information from all previous hidden states, they often struggle to capture long-term dependencies due to exploding and vanishing gradients. Methods like gradient clipping [Mik12] can mitigate exploding gradients, but vanishing gradients remain a significant challenge. Gated RNNs such as Long Short Term Memory (LSTM) [HS97], address this issue by using gates and cells to decide which information to retain or forget when processing long sequences.

LSTM consists of several components that regulate the flow of information. The forget gate determines which past information should be discarded, allowing the model to remove irrelevant details. The input gate decides which new information should be added to the memory cell, ensuring that only relevant updates are stored. The cell state acts as a long-term memory, preserving useful information across time steps. The output gate controls what information from the updated cell state should be passed to the next hidden state, influencing the model's final decision. These mechanisms enable LSTMs to effectively handle long-term dependencies and reduce the vanishing gradient problem.

Several alternatives and extensions to LSTMs have been proposed, with Gated Recurrent Unit (GRU) [JZS15] being a popular choice. GRUs simplify the more complex LSTM units by using two gates instead of three and combining the cell state and hidden state into one. This streamlined design reduces computational load and simplifies training, making GRUs faster and less resource-intensive. Despite their simplicity, GRUs are generally reported to perform as well as LSTMs in most practical tasks.

2.6 Sequence to sequence Models

Many applications in NLP involve transforming an input sequence into a different output sequence. For instance, in machine translation, the task is to translate text from a source language into a target language. Similarly, in summarization, the goal is to produce a concise version of a longer text. These problems are typically addressed using models known as sequence-to-sequence or encoder-decoder models. In these models, one neural network, typically an RNN, encodes the input sequence, and a second network decodes this information to generate the output sequence as shown in Figure 2.1.

Early approaches represent the entire input sequence using a single context vector $c = h_{T_x}$, where h_{T_x} denotes the final hidden state produced by the encoder RNN, which is a function of the previous hidden state and the current input. The decoder then uses this fixed-length context vector c , along with previously generated outputs, to predict the target sequence, as shown in Equation 2.21.

$$p(y_i \mid y_{1:i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c) \quad (2.21)$$

where y_i is the i -th word in the output sequence, $\mathbf{x}$ is the input sequence, $y_{1:i-1}$ represents all previously predicted words, s_i is the decoder's hidden state at step i , and g is a nonlinear function like a feed-forward neural network that outputs the probability of the output.

A major limitation of using a fixed-length context vector c shown in Figure 2.1 (Information bottleneck) is that it must encompass all essential information from the source text, as the decoder relies solely on it for interpretation. In longer sentences, important details from earlier parts may not be fully captured by this context vector, leading to information loss and decreased model effectiveness [BCB15].

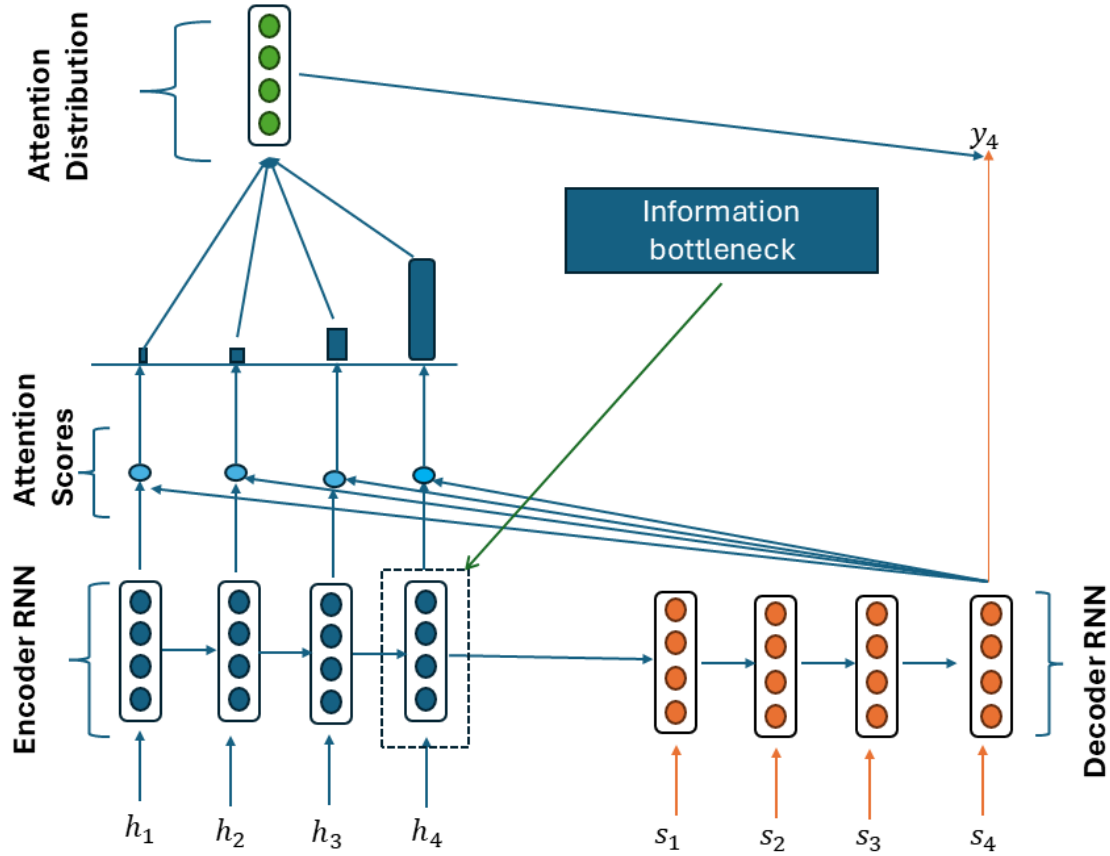


Figure 2.1: Attention in RNNs [BCB15]

The attention mechanism addresses the bottleneck problem by enabling the decoder to access all the encoder's hidden states, rather than just the final one. It improves the encoder-decoder framework by calculating a dynamic context vector c_i at each decoding step, integrating information from the full set of encoder hidden states, as shown in Equation 2.22 and Figure 2.1.

$$p(y_i \mid y_{1:i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i) \quad (2.22)$$

The context vector at each step is calculated as a weighted sum of the encoder hidden states of length T_x , with a strong emphasis on the parts that surround or influence the i -th output word, as shown in Equation 2.23.

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \quad (2.23)$$

These attention weights α_{ij} are then normalized with a softmax function, as shown in Equation 2.24.

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \quad (2.24)$$

where $e_{ij} = a(s_{i-1}, h_j)$ denotes an alignment model that evaluates how well the encoder hidden state at position j corresponds to the decoder state at position i . By allowing the decoder to focus on different parts of the input for each output word, the attention mechanism effectively reduces the bottleneck caused by the fixed-length context vector.

2.7 Transformers

Attention significantly improves the performance of RNNs in various applications. But the inherent sequential nature of RNNs even with attention remains a major limitation. Outputs at a future time step can only be computed after all the previous outputs have been computed, restricting the potential for parallel processing. This sequential dependency slows down both training and inference, a challenge that becomes even more pronounced when LLMs are trained with modern NLP datasets that contain billions of tokens. To address the limitations of recurrence, a transformer [VSP⁺17] exclusively utilizes an attention mechanism to capture global dependencies between inputs and outputs.

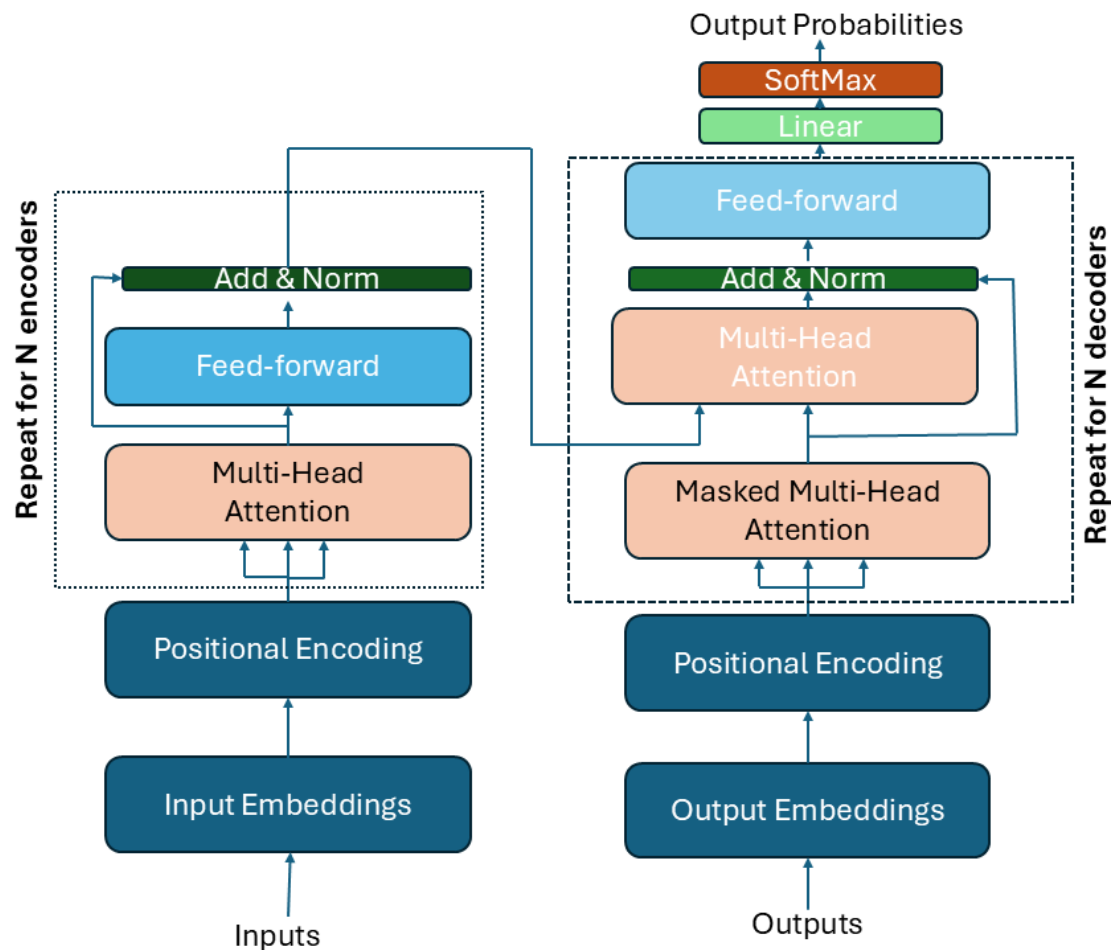


Figure 2.2: Overview of the components of the transformer architecture [VSP⁺17]

The original transformer model shown in Figure 2.2 is based on an encoder-decoder architecture. The encoder converts an input sequence representation into a sequence of continuous representations. The decoder then generates an output sequence from these representations. The encoder consists of N identical layers stacked on top of each other. Each layer has sub-layers: a multi-head self-attention mechanism and a position-wise feed-forward network. Residual connections are used around each sub-layer, followed by layer normalization.

Similarly, the decoder consists of N identical layers, each containing sub-layers like the encoder, along with an extra sub-layer that performs multi-head attention over the encoder's output. In the decoder, the self-attention sub-layer is modified with masking to prevent positions from attending to future positions.

Self-attention is an essential component in transformers for representing how words in a sequence influence the current input. It works by taking a query and looking up information in a key-value store, selecting the values of the keys that are most similar to the query. Each input token in self-attention has three roles: as a "query" focusing on the current input, as a "key" for comparison, and as a "value" for computing the output.

Transformers use weight matrices Q , K , and V to project an input token x_i into these roles. For a token x_i in the sequence, the query q_i is defined as: $q_i = Qx_i$, then key is defined as: $k_i = Kx_i$ and the value is defined: $v_i = Vx_i$.

The contextual representation of each x_i is obtained as a weighted sum of the sequence values defined by Equation 2.25, for a sequence of n tokens.

$$h_i = \sum_{j=1}^n \alpha_{ij} v_j \quad (2.25)$$

where α_{ij} are attention weights that represent the strength of contribution of each v_j . These attention weights are determined by the affinities between the keys and the query, calculated using Equation 2.26.

$$\alpha_{ij} = \frac{\exp(q_i^T k_j)}{\sum_{j'=1}^n \exp(q_i^T k_{j'})} \quad (2.26)$$

This process evaluates the significance of various tokens in representing x_i within a given context, employing different perspectives of x_i for the key, query, and value roles.

Instead of computing a single output at a time step i where the output y_i is computed independently, the entire process is parallelized using efficient matrix multiplications. This is done by packing the input embedding matrices of n tokens and model dimension d from the input sequences into a single matrix $X \in \mathbb{R}^{n \times d}$ and simultaneously computing the necessary query-key comparisons through matrix multiplication. Each row of X represents the embedding of one token of the input, with transformers for the most recent LLMs supporting input lengths of even up to millions of tokens.

In attention calculations, the dot product can result in large values, causing the softmax function to have very small gradients. To mitigate this, the original transformer scales the dot product by $1/\sqrt{d_k}$, where d_k is the dimension of the keys and K^T is the transpose of K .

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2.27)$$

To optimize GPU parallel processing, QK^T computes scores for all keys and values, including future tokens. Masked attention is then applied to zero out the attention weights for future tokens, ensuring each token attends only to previous or current tokens, preserving the autoregressive nature and preventing information leakage, as shown in Equation 2.28.

$$\alpha_{ij, \text{masked}} = \begin{cases} \alpha_{ij} & \text{if } j \leq i \\ 0 & \text{otherwise} \end{cases} \quad (2.28)$$

Words in a sentence can have multiple simultaneous relationships, so transformers use a multi-head self-attention where different heads of the same depth with each having its own set of parameters learn different aspects of input relationships as shown in Equation 2.29.

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) [VSP^+17] \quad (2.29)$$

The matrix W_i^Q is the query projection matrix that transforms input embeddings into query representations. The matrix W_i^K is the key projection matrix that encodes input features for relevance computation. The matrix W_i^V is the value projection matrix that represents the actual content to be aggregated. The matrix W^O is the output projection matrix that combines the outputs of multiple attention heads into a single representation.

These heads' outputs are concatenated (Concat) for the final multi-head attention output calculated in Equation 2.30.

$$MultiHead(Q, K, V) = Concat(head_i, \dots, head_h)W^O \quad (2.30)$$

To completely eliminate recurrence, transformers had to solve the problem of capturing input positions. Unlike RNNs, which inherently account for token order due to their sequential nature, transformers do not naturally understand the relative or absolute positions of tokens, as their attention mechanism does not consider this. To address this, transformers incorporate positional encodings into word embeddings, providing the model with information about token positions.

The attention mechanism in a transformer computes a weighted sum of value vectors without introducing explicit non-linearities. To enable the model to learn complex patterns, transformers apply a feed-forward network independently to each position after the attention mechanism, introducing the necessary element-wise non-linear transformations.

For efficient learning, transformers also use layer normalization and residual connections [BKH16]. Layer normalization stabilizes the distribution of activations within a layer, providing more uniform inputs for subsequent layers. This is done by computing and normalizing the mean and variance of a layer's activations. Residual connection simply adds the input of a layer to the output of that layer which allows information to pass directly from a lower layer to a higher layer, bypassing intermediate layers, which makes it easier to train deeper networks.

2.8 Pre-training

The introduction of transformers in NLP has established a new learning paradigm called pre-train and fine-tune, where models are not built from scratch but instead undergo extensive pre-training on large text corpora containing billions of tokens. This process allows models to learn complex linguistic structures and knowledge, which can then be transferred to specific tasks during fine-tuning. By leveraging pre-trained weights, these models start with a strong foundation in language, reducing the amount of data needed for task-specific adaptation. Pre-trained LLMs are typically categorized based on their self-supervised pre-training objectives such as Masked Language Modeling (MLM) and Next Token Prediction (NTP), and are also classified into encoder-only, decoder-only, and encoder-decoder architectures [LYF⁺23].

A synonymous term for these models is foundation models, which refers to models trained on a broad range of data, often using large-scale self-supervision, and are adaptable to a wide range of tasks [BHA⁺21]. While pre-trained LLMs fall under this category, foundation models also include those trained on different modalities such as vision, audio, video or multimodal architectures that can process multiple modalities simultaneously.

2.8.1 Encoder-only Language Models

The objective of language modeling, which involves predicting the next token in a sequence, typically processes text in a single direction from left to right. While effective for many tasks, this approach can be limiting when future tokens provide important contextual information for interpreting the current tokens being processed. Bidirectional encoders, also known as encoder-only models, address this limitation by using encoder attention layers to analyze all words in a sentence including future tokens simultaneously.

They achieve this through the MLM pre-training objective, which differs from the traditional language modeling objective. Instead, MLM focus on predicting missing words within a sentence. For instance, rather than completing the sentence: "Peter loves ...", an MLM objective might require predicting the missing word in the sentence: "Peter mangoes", which ensures a deeper semantic understanding of the sentence. These models are particularly effective for tasks that require a comprehensive understanding of entire sequences like classification, extractive question answering, and named entity recognition.

A prominent example of encoder-only models is the Bidirectional Encoder Representations from Transformers (BERT) [DCLT19]. The pre-training process of BERT involves

masking 15% of the input tokens, where 80% of these are replaced with a *[MASK]* token, 10% are substituted with a random token, and the remaining 10% are unchanged. The primary objective is to accurately predict the original tokens at these masked positions using a bidirectional encoder. Additionally, BERT incorporates the Next Sentence Prediction (NSP) objective, which requires the model to determine whether two sentences appeared consecutively in the corpus. BERT was pre-trained on a large corpus, which includes the 800-million-word BooksCorpus [ZKZ⁺15] and the 2.5 billion-word English Wikipedia dataset.

The success of BERT inspired the development of several extensions and variants such as: Robustly Optimized BERT Pretraining Approach (RoBERTa) [LOG⁺19], Efficiently Learning an Encoder that Classifies Token Replacements Accurately (ELECTRA) [CLLM20] and Extra Long Network (XLNet) [YDY⁺19] among others. RoBERTa [LOG⁺19] introduces several improvements without changing BERT's underlying architecture. Key improvements include extended training duration with larger batch sizes and increased data. Unlike BERT, which emphasizes the NSP objective, RoBERTa's experiments showed that removing the NSP does not impact the model's performance. Additionally, RoBERTa adopts dynamic masking, generating a new masking pattern for each sequence during training, unlike BERT's static masking approach that reuses the same pattern across all epochs.

ELECTRA [CLLM20] proposes a pre-training method that uses replaced token detection, which differs from the MLM objective used in BERT. In this approach, the input text is corrupted with plausible alternatives generated by a small network, and a discriminator is trained to identify whether each token in the input text is original or replaced. This method addresses the mismatch of the *[MASK]* tokens between pre-training and fine-tuning observed with BERT.

XLNet [YDY⁺19] uses a permutation language modeling approach to combine the strengths of autoregressive and masked language models. XLNet is based on the idea that for a sequence of length n , there are $n!$ possible ways to order the autoregressive factorization orderings. By sharing model parameters across all these different orderings, XLNet gathers information from all positions in the sequence, allowing it to learn from the full context on both sides of each token.

2.8.2 Decoder-only Language Models

While encoder-only LLMs excel in tasks requiring a complete understanding of input sequences [DCLT19], they are less effective for generation tasks. In contrast, decoder-only models, also known as autoregressive models excel in generation tasks. They use attention layers at each stage to access only the words positioned before the current

word in the sequence using the NTP objective. Representative examples of decoder-only models include: the GPT series (GPT-1, GPT-2, GPT-3, GPT-4) [Ope23] and many SoTA LLMs like BigScience Large Open-science Open-access Multilingual Language Model (BLOOM) [SFA⁺22], LLAMA [TMS⁺23], and Open Pretrained Transformer (OPT) [ZRG⁺22].

GPT-1 [RNSS18] is the first version of the GPT models and laid a foundation for most modern LLMs. This model follows a two-stage training process: generative pre-training and fine-tuning on downstream tasks. In the pre-training stage, the model learns language patterns and contextual relationships by predicting the likelihood of the next token in a given sequence on a large corpus. The pre-training dataset includes the BooksCorpus and a 1-billion-word benchmark dataset. The model architecture consists of a 12-layer, decoder-only transformer with 768-dimensional hidden states and 12 attention heads.

GPT-2 [RWC⁺19] was proposed as an improvement upon GPT-1 by using the same transformer architecture but at a larger scale, incorporating more layers and attention heads. It increases the parameters from 117 million in GPT-1 to 1.5 billion. The vocabulary size is expanded to 50,257, and the context size is increased from 512 to 1,024 tokens. The pre-training dataset, WebText is much larger including 8 million documents and 40 GB of text from diverse internet sources, excluding Wikipedia. GPT-2 demonstrates early abilities to learn tasks without explicit supervision after pre-training.

GPT-3 [BMR⁺20] is an autoregressive language model with 175 billion parameters, ten times more than any previous LLM. It was trained on a vast and diverse dataset, including several curated high-quality datasets like an expanded WebText, Books1, Books2, and English Wikipedia. The final dataset mix included 41 shards of Common Crawl data from 2016 to 2019, totaling 45TB of compressed plaintext before filtering and 570GB after filtering. Unlike previous pre-trained language models that required extra fine-tuning, GPT-3 demonstrates emergent abilities of in-context learning, performing tasks with just human instructions or a few demonstration examples added to the context with no parameter updates. GPT-3 was reported to achieve SoTA performance across all benchmarks, surpassing any language model before it.

Unlike GPT-1 and GPT-2, GPT-3's pre-trained parameters were not released to the open-source community and could only be accessed via their API. However, several open-source models were published after GPT-3, achieving comparable performance. One notable example is the OPT [ZRG⁺22] released by Meta. OPT is open-source and includes a wide range of models with parameters ranging from 125 million to 175 billion, reported to achieve performance comparable to GPT-3 while requiring only one-seventh of its carbon footprint to develop [ZRG⁺22]. The pre-training corpus for

OPT comprised around 180 billion tokens, including datasets from those used to train RoBERTa [LOG⁺19], the Pile [GBB⁺20], and PushShift.io Reddit [BZK⁺20].

Another pre-trained language model is BLOOM [SFA⁺22], a decoder-only multilingual transformer with its largest version having 176 billion parameters. BLOOM was trained on the Responsible Open-science Open-access Text Sources (ROOTS) corpus, a dataset composed of hundreds of sources in 46 natural languages and 13 programming languages. There other hundreds of pre-trained models which can be accessed on the Huggingface platform.

2.8.3 Encoder-Decoder Language Models

These models combine the strengths of encoders and decoders using an encoder-decoder architecture. In this setup, the encoder's attention layers can access the entire input sequence, whereas the decoder's attention is limited to preceding tokens. The decoder generates output from left to right, conditioned on the encoder's output, using a fully-connected attention mask. Popular models in this category include: Text-to-Text Transfer Transformer (T5) [RSR⁺20] and Bidirectional and Auto-Regressive Transformer (BART) [LLG⁺20].

T5 [RSR⁺20] is a text-to-text transformer model that leverages transfer learning for NLP tasks through a unified framework, converting tasks like translation, classification, summarization, and question answering into a format where both the input and output are text strings. T5 uses an encoder to process the input text into hidden representations, which the decoder then uses to generate the output text. It was pre-trained on the Colossal Clean Crawled Corpus (C4) dataset, a large cleaned version of the Common Crawl corpus, using a denoising objective to learn to generate missing or corrupted parts of the input text. This unified framework and extensive pre-training was reported to enable T5 to achieve SoTA performance across various NLP benchmarks.

BART [LLG⁺20] is a denoising autoencoder designed for pre-training sequence-to-sequence models. It corrupts text using various noising techniques and then trains the model to reconstruct the original text. BART's architecture combines a bidirectional encoder, similar to BERT, which processes the entire input text to understand the context, and a left-to-right autoregressive decoder, like GPT, which generates or reconstructs text based on the encoder's understanding. During pre-training, noise is added to the input, such as shuffling sentences or replacing spans of text with a mask token, and the model is trained to restore the original sequence. BART was pre-trained on a large dataset, including BookCorpus, Wikipedia, CC-News, OpenWebText, and Stories, which was reported to be highly effective for both text generation and comprehension tasks.

2.9 Fine-tuning and Prompting

After pre-training, which enables LLMs to capture patterns and generalizations from large text corpora, they can be adapted for specific applications through fine-tuning on domain-specific datasets. Fine-tuning involves training the model on a labeled data relevant to the target task while retaining its general linguistic knowledge obtained during pre-training. The process starts by loading the pre-trained model weights and making necessary modifications, such as adding a classification layer for text classification. The model parameters are then optimized using loss functions to minimize the difference between predictions and actual labels. Unlike traditional ML or deep learning models that learn from scratch, fine-tuned LLMs leverage pre-existing knowledge, allowing them to learn new tasks efficiently with significantly less data compared to training a model from scratch [DCLT19].

As the scale of data and parameters increase during pre-training, as seen in models like GPT-3 with billions of parameters, LLMs develop an emergent ability known as in-context learning [BMR⁺20]. This ability allows LLMs to perform various tasks during inference without requiring explicit re-training. Instead of fine-tuning for each specific task, prompting enables zero-shot learning, where the model completes tasks based solely on natural language instructions. A prompt serves as a structured text input that guides the model to perform a specific task. For instance, sentiment analysis can be conducted by providing a straightforward prompt shown in Figure 2.4 using simple human written instructions.

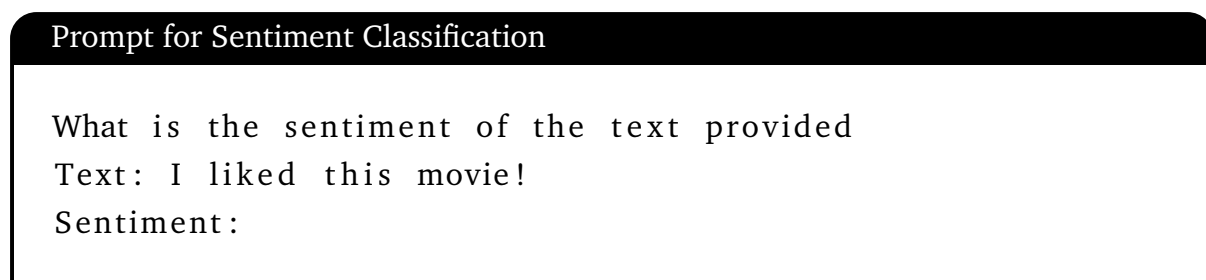


Figure 2.3: Zero-shot prompt template for sentiment classification

The process of optimizing prompts to enhance LLM outputs is known as prompt engineering, which typically involves iterative testing of different phrasings and structures to improve accuracy. Beyond basic instructions, few-shot prompting incorporates a small number of demonstration examples within the prompt, helping the model to better understand the task structure. Unlike traditional training methods, few-shot prompting does not modify the model's parameters but relies solely on the information provided in the prompt during inference. For instance, in sentiment analysis, few-shot

prompting may include labeled examples before the actual input query as in Figure 2.4.

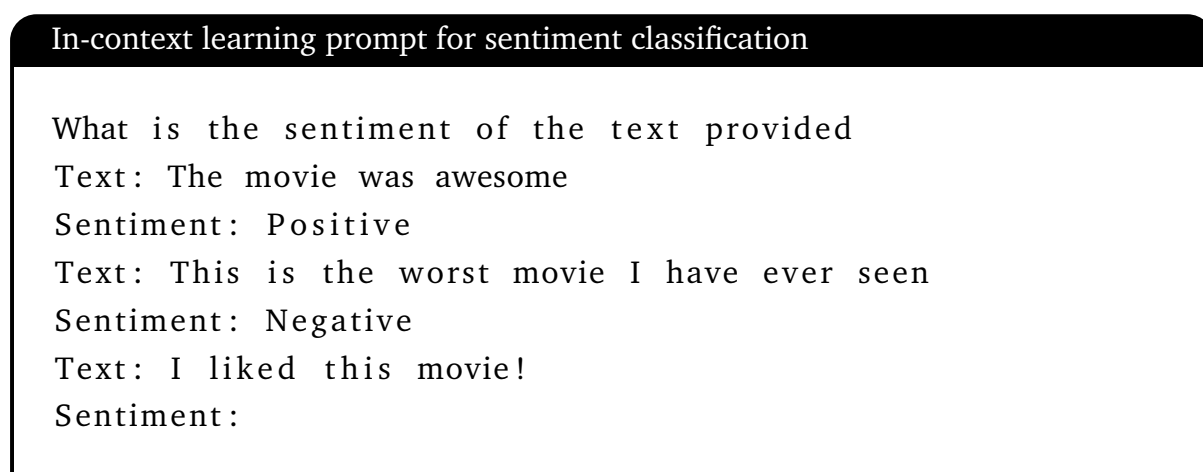


Figure 2.4: Few-shot prompt template for sentiment classification

Various improvements have been introduced to enhance prompting techniques, with Chain of Thought (CoT) and Tree of Thought (ToT) prompting being among the most common methods for improving the reasoning capabilities of LLMs. CoT prompting encourages step-by-step reasoning, either explicitly or through few-shot demonstrations that include intermediate explanations. This structured approach helps the model build logical reasoning chains before generating a final response, often allowing it to identify and correct errors [WWS⁺22]. ToT further enhances LLM problem solving by structuring reasoning in a tree-like format. Unlike CoT, which follows a linear sequence, ToT enables models to explore multiple reasoning paths, assess different solutions, and backtrack when necessary [YYZ⁺24b].

2.10 Language Model Alignment

Prompting makes LLMs highly versatile and adaptable, as they only require instructions, not new datasets, to perform tasks. However, LLMs are originally trained to predict the next token during pre-training, but now they are expected to follow user instructions and satisfy human preferences through these prompts. This creates a mismatch between the original training objective (NTP) and the new requirement (human instructions and preferences). This section describes techniques like instruction tuning and preference alignment, which aim to reduce this mismatch.

2.10.1 Instruction Tuning

Instruction tuning enhances an LLM's ability to follow instructions by applying supervised fine-tuning to a pre-trained model across multiple tasks. This process involves

training the model on a dataset of instruction-response pairs, where each instruction has a clearly defined correct response. While the model continues to predict the next token as in pre-training, instruction tuning introduces an additional constraint of ensuring the generated response aligns with the given instruction. This approach improves the model's accuracy and contextual understanding, enabling it to produce more relevant and precise responses across various tasks [SWR⁺21].

Instruction tuning has led to the development of instruction-tuned LLMs, which are significantly more effective at following user instructions compared to standard pre-trained models. Prominent models in this category include: Fine-tuned LAnguage Net-T5 (FLAN-T5) [CHL⁺24], Falcon [AAA⁺23], Mistral [JSM⁺23], and Dolly [CHM⁺23]. FLAN-T5 is based on the T5 architecture, and was fine-tuned using the FLAN dataset, which is composed of data points formatted into a template of an instruction, an input, and an output across multiple NLP tasks [LHV⁺23]. This fine-tuning was shown to significantly enhance its capabilities, enabling it to outperform the original T5 model and achieve performance comparable to larger models like Pathways Language Model (PaLM) [CND⁺23] in few-shot learning scenarios, and excelling across multiple evaluation benchmarks.

Falcon [AAA⁺23] is an LLM available in multiple parameter sizes, including 7B, 40B, and 180B. It is designed as a causal decoder-only model and trained on a diverse, high-quality dataset primarily sourced from web data. The largest variant, Falcon-180B, has been trained on over 3.5 trillion tokens. It was shown to outperform models such as PaLM while also surpassing the performance of models like LLAMA2. Furthermore, Falcon-180B was reported to achieve performance comparable to PaLM-2-Large while maintaining lower pre-training and inference costs.

Mistral [JSM⁺23] offers two categories of LLMs: open-weight models (Mistral-7B, Mixtral-8x7B, Mixtral-8x22B) and optimized commercial models. The open-weight models, available under the Apache2 license, can be fine-tuned for various downstream tasks. Mistral-7B was reported to surpass the 13 billion-parameter LLAMA2 across all benchmarks and also outperforming the 34 billion-parameter LLAMA1 in reasoning, language understanding and code generation. It incorporates grouped query attention for faster inference and sliding window attention to efficiently handle variable sequence lengths.

Dolly [CHM⁺23] is an instruction following LLM built on the Pythia-12B model [BSA⁺23]. It is fine-tuned on approximately 15,000 instruction-response pairs which covers various tasks such as: brainstorming, classification, question answering, text generation, information extraction, and summarization [CHM⁺23]. While not a SoTA model, Dolly-12B was reported to demonstrate strong instruction-following

capabilities. Smaller variants include: Dolly-7B and and Dolly-3B.

2.10.2 Preference Alignment

Although instruction tuning enhances LLMs' ability to respond to user queries, there remains a gap between the model's objective of generating responses and the goal of aligning with human preferences in those responses. This discrepancy raises the challenge of explicitly training models to better reflect user preferences. A common solution to this issue is to further refine LLMs using RLHF after instruction tuning [OWJ⁺22].

RLHF improves an LLM's alignment with human preferences through a three-step process. The first step is supervised instruction tuning, similar to the approach discussed in section 2.10.1. In this phase, labelers provide desired responses for various instructions, which are used to fine-tune the pre-trained LLM. This produces an instruction-tuned model capable of following user instructions, though not fully optimized for human preferences.

The second step involves training a reward model, which plays a crucial role in the final stage of refinement. To build the reward model, comparison data is collected by generating responses to sampled prompts using different models or configurations of the same model. Human labelers then rank these responses based on how well they align with the given prompt, selecting the preferred and less preferred responses. These rankings are then used to train a reward model that assigns a scalar reward to each response, guiding further optimization of the LLM.

The final step in the RLHF optimizes the instruction-tuned model using reinforcement learning. A prompt is sampled from the dataset, and the model being aligned (policy) generates a response, which is then evaluated by the reward model. The assigned reward is used to update the policy through reinforcement learning algorithms such as proximal policy optimization [SWD⁺17].

However, as training progresses, the model's outputs may diverge from the intended responses, sometimes generating incoherent text while still receiving high rewards from the reward model. To prevent excessive deviation, a penalty is applied to keep the updated model aligned with the original instruction-tuned version. A common method for this is the Kullback-Leibler divergence penalty, which regulates the difference between the newly trained model and the original instruction-tuned model, ensuring that responses remain consistent and meaningful.

InstructGPT [OWJ⁺22] models from OpenAI, trained using this approach, demonstrate significant improvements over GPT-3, with human evaluations showing that labelers consistently prefer InstructGPT outputs. Despite having 100 times fewer parameters,

the 1.3 billion-parameter InstructGPT model is favored over the 175 billion-parameter GPT-3. In addition to improved response quality, InstructGPT exhibits greater truthfulness and a reduced tendency to generate toxic content. This training method has influenced the development of modern LLMs such as ChatGPT [Ope22] and GPT-4 [Ope23]. While specific training details for these models remain undisclosed, their technical reports indicate that RLHF was incorporated into their training process. Other LLMs shaped by this approach include: LLAMA [TMS⁺23] and Claude3 [Ant24] among others

LLAMA [TMS⁺23] is a series of LLMs created by Meta with three versions so far: LLAMA1, LLAMA2, and LLAMA3 ranging from 7 billion to 70 billion parameters. LLAMA1 was the first iteration pre-trained with NTP having parameters between 7 billion to 65 billion. LLAMA2 improved on LLAMA1 enhancing chat capabilities and following instructions better with extra instruction tuning and RLHF. LLAMA3 is the latest version with 8 billion and 70 billion parameters. These models are much better than LLAMA2, and trained on a high quality dataset with 15 trillion tokens which is 7 times larger than the one used to train LLAMA2.

Claude3 [Ant24] is a family of LLMs trained by Anthropic: Claude3-Opus, Claude3-Sonnet, and Claude3-Haiku. Claude3-Opus is the most capable, combining skills and speed, while Claude3-Sonnet offers a balance of abilities, and Claude3-Haiku is the fastest and least expensive. These models are reported to have advanced language and vision capabilities, strong performance in reasoning, and coding benchmarks.

RLHF is a complex and sometimes unstable training process because it first requires fitting a reward model to represent human preferences, then using reinforcement learning to maximize this estimated reward without deviating too much from the original instruction-tuned model. Direct Preference Optimization (DPO) [RSM⁺23] simplifies this by optimizing a language model to follow human preferences without the need for explicit reward modeling or reinforcement learning. DPO combines reward modeling and preference alignment into a single step through a simple classification objective based on maximum likelihood, fitting an implicit reward model.

Collecting data for instruction tuning and preference alignment from humans is expensive and time consuming, so an alternative is to use other LLMs to generate this data. The Zephyr model from Huggingface [TBL⁺23] is an example of a model aligned purely with AI feedback. Instead of humans providing responses, an LLM simulates user-assistant interactions to create synthetic instruction and output datasets. Similarly, preference data is generated using multiple models, with GPT-4 ranking the responses instead of human evaluators. Zephyr's performance was reported to surpass all other 7-billion parameter models at the time and even outperformed the LLAMA2-70B model,

which was trained with human feedback.

A major limitation of LLMs, even after instruction tuning and preference alignment, is that their knowledge is confined to the data available during training. As a result, they require updates for new information. Given the continuous generation of new data, frequently updating large-scale models is impractical, especially for billion-parameter LLMs. To address this, non-parametric approaches such as RAG [LSC⁺22] and LLM agents [YZY⁺23] have gained popularity for integrating new knowledge during inference. These methods retrieve external information from databases and search engines, incorporating it into prompts to provide updated responses without the need for model re-training.

2.11 Retrieval Augmented Generation and Language Model Agents

LLMs often generate responses based on learned probabilities rather than factual data, leading to issues such as hallucinations, outdated knowledge, and unverifiable reasoning. RAG mitigates these challenges by incorporating a retrieval system that dynamically fetches relevant data, ensuring that generated responses are more accurate, reliable, and contextually relevant.

RAG consists of two main components: a retriever and a generator. The retriever identifies and retrieves external documents or data to enhance the model's responses using different retrieval methods. Traditional term-based techniques, such as keyword search and Best Matching 25 (BM25) [RZ⁺09], locate documents containing specific terms, while more advanced embedding-based retrieval methods convert both queries and documents into vector representations, allowing the system to retrieve semantically similar information. Some implementations combine term-based and embedding-based retrieval to balance efficiency and accuracy. Once relevant documents are retrieved, they are ranked based on relevance to the query and passed to the generator, often as context within a prompt.

The generator, usually an LLM, integrates the retrieved information with the user query to produce a final response. To enhance response quality, additional processing techniques are often applied. Query processing methods, such as query rewriting and expansion, help refine ambiguous or incomplete inputs, improving retrieval accuracy. However, the effectiveness of RAG-based models depends significantly on the quality of the documents stored in the vector database, which must be regularly updated with new information to maintain relevance and ensure accurate responses.

LLM agents extend capabilities of RAG by enabling models to access information beyond documents in the databases. The foundational approach to LLM agents was introduced through the Reasoning + Acting (ReACT) framework [YZY⁺23], which integrates reasoning techniques such as CoT and ToT with real-time interaction using external tools. These tools can include search engines, APIs, calculators, and user-defined utilities, in addition to the vector databases used in RAG. A key feature of LLM agents is their structured approach to task execution. When given a query, the agent first analyzes the problem, then formulates a step-by-step plan, and then determines the necessary tools for execution. Throughout this process, the agent continuously evaluates its progress, verifying each step before proceeding, which enhances accuracy.

Despite their advantages, deploying LLM agents presents challenges, particularly in reducing errors and inefficiencies. Since these agents often function autonomously, mistakes in planning, tool execution, or reasoning can accumulate, leading to incorrect outputs. Ensuring robustness requires careful system design, continuous monitoring, and mechanisms to dynamically correct errors.

2.12 Multimodality and General Purpose Language Models

Multimodal AI models have become a crucial advancement in AI, bridging different types of data and enabling machines to process and interpret multiple modalities simultaneously. Traditional ML models and even recent pre-trained models primarily operate within a single modality, such as language, vision, or audio, limiting their ability to generate complex, context-rich responses. However, human cognition is inherently multimodal, as we process language, visual information, and auditory cues simultaneously to understand our environment. To develop AI systems that function effectively in real-world applications, integrating multiple data types has become essential.

One of the earliest multimodal models is Contrastive Language Image Pretraining (CLIP). CLIP was trained on a large dataset of image-text pairs, enabling it to associate textual descriptions with corresponding images using contrastive learning [RKH⁺21]. Unlike earlier vision models that relied on labeled datasets with predefined categories, CLIP mapped both text and image embeddings into a shared embedding space, allowing it to generalize across a broad range of visual concepts. This capability enabled zero-shot classification, allowing the model to recognize objects and concepts it had never encountered during training.

Another prominent multimodal model is DALL-E, which was designed to generate images from textual descriptions. Unlike CLIP, which focused on understanding and clas-

sifying existing images, DALL-E was trained to create entirely new visual content based on linguistic input [RDN⁺22]. Building on these advancements, Flamingo [ADL⁺22] introduced an open ended text generation framework that combined both text and image processing. Instead of using separate pipelines for handling text and visual data, Flamingo integrated a vision encoder similar to CLIP for image processing, combined with CLIP for generating textual outputs, enhancing its ability to understand and respond to multimodal inputs.

Current SoTA LLMs, such as GPT-4, have introduced multimodal variants like GPT-4-Vision, an extension that enhances text generation with advanced multimodal reasoning capabilities [Ope23]. Similarly, the latest versions of other LLM families, including LLAMA, Qwen [YYZ⁺24a], and Gemini [TAB⁺23], support multimodal processing.

As multimodal AI continues to progress, researchers are exploring new techniques to incorporate additional data types, including video, audio, and three dimensional representations. The shift from single modality models to integrated multimodal systems, combined with prompt driven interactions in natural language, is gradually transforming these models from specialized tools into flexible, general purpose AI systems. These advancements enable AI to perform a wider range of tasks across multiple domains without the need for separate training for each application.

This chapter presented the essential concepts in ML and NLP necessary to understand the context of the work described in this Thesis. In chapter 3, a review of the literature associated with the contributions of the Thesis is presented.

Chapter 3

Related Work

This chapter provides a detailed review of existing research on the adaptation of ML and LLMs in government and public sector document processing. It is organized in three main sections. Section 3.1 reviews studies on classifying EPU in news articles, covering approaches from traditional keyword based methods to LLM techniques. It also highlights key limitations in existing research.

Section 3.2 reviews the use of LLMs and ML techniques for analyzing citizen feedback in various government applications and automating customer ticket processing. The methods discussed include: unsupervised ML, traditional ML, deep learning, and LLM based approaches. Additionally, it identifies gaps in existing research, highlighting the need for further study and serving as a motivation for the research conducted in this Thesis.

Section 3.3 reviews studies on applying LLMs and NLP to process public sector documents across various domains including political discourse, legislative proposals, policy briefs, and press releases. It discusses relevant works in areas such as fine-tuning LLMs for government applications, leveraging RAG and LLM agents with government documents, and tackling hallucinations in LLM generated responses. The section also identifies research gaps in using LLMs for government document processing, some of which serve as motivation for this Thesis.

3.1 Weakly Supervised Learning for Predicting Economic Policy Uncertainty from News Articles

With technological advancements, an increasing share of human interactions, communication, and cultural activities is being recorded as digital text. This includes social media posts, news articles, and transcripts of spoken exchanges in political, legal, and

3. RELATED WORK

economic contexts. Analyzing these digital records provides valuable insights into the relationship between language use, human behaviors, and social structures. However, the rapid growth of digital content has made manual analysis impractical, limiting scalability and efficiency while struggling to meet time constraints.

The rapidly progressing field of text-as-data utilizes computational methods to analyze unstructured digital text collections [GKT19]. This approach has found applications across various disciplines including sociology, economics, and political science [GS13]. In economics, for example, textual data has been applied on varied data sources such as newspaper articles [BBD16], central bank communications [R⁺19], financial earnings calls [KS19], and social media data [Ind20], providing valuable insights into social processes.

One prominent example of a text-as-data application is the EPU index proposed by Baker et al. (2016). This index was constructed by extracting news articles that contained keywords related to economy, policy, and uncertainty as a proportion of all news articles published within a certain time frame. The resulting index has proven to be useful for extracting economic signals from text data. However, the methodology of keyword searches used in the identification stage is less rigorous and prone to a high count of false positives [BBD16].

Automating the detection of EPU-related news articles is the most efficient way to minimize the count of false positives. Due to the significance of this application, several studies have applied unsupervised ML models including topic modeling, alongside other ML techniques to enhance detection accuracy. This section reviews relevant research that has contributed to improving EPU classification, categorized into unsupervised ML techniques, traditional ML approaches, deep learning, and LLM techniques, summarized in Table 3.1 and discussed next.

Table 3.1: A table summarizing relevant studies on automated methods for EPU classification

Category	Relevant Works
Unsupervised ML	[AGHOS20], [XRWW23], [AG17], [WYZG24], [MBMSC23], [KYZ21], [NO20], [AGDGP23], [JFS23], [BBD16]
Traditional ML	[LSL22], [TND ⁺ 18], [Sad22], [WLZF22]
Deep Learning & LLMs	[RTMS24], [CHY24], [QCDD21], [KTOM20], [CHLS24], [AGS24], [YHPC24], [ISO25]

Unsupervised ML Approaches: Most EPU detection methods have utilized unsupervised learning techniques to identify news articles related to EPU. The earliest and most widely used approach, introduced by Baker et al. (2016), relies on keyword matching

for classification. Building on this, alternative approaches have been proposed to enhance the detection process. For instance, Kaveh et al. (2021) constructed an EPU index by leveraging a word embedding representation space to refine keyword selection for EPU detection in news articles. Nyman and Ormerod (2020) further expanded the original uncertainty keyword list proposed by Baker et al. (2016) using nearest neighbor embeddings and examined the Granger causality between their expanded keyword set and the existing EPU index [NO20].

While keyword-based methods offer a simple and interpretable solution, they often suffer from high false positive rates and limited semantic understanding. To overcome these limitations, topic modeling techniques have been proposed as a more robust alternative for uncovering latent themes related to EPU. Relevant studies include: Azqueta et al. (2020), who applied topic models to construct EPU indices for European countries [AG17, AGHOS20, AGDGP23].

Wang et al. (2024) used Latent Dirichlet Allocation (LDA) to extract themes from newspapers articles and constructed the Chinese EPU index from them [WYZG24]. Miranda et al. (2023) implemented topic modeling with semantic clustering based on word embeddings to detect EPU in digital news [MBMSC23]. Jiang et al. (2023) applied BERTopic and regularized linear models to predict EPU using social media data from Reddit [JFS23]. Xu et al. (2023) utilized an unsupervised approach combining principal component analysis and random matrix theory to construct EPU indices [XRWW23]. Although topic modeling captures latent semantic structures, its unsupervised nature limits classification accuracy compared to supervised learning. To improve precision and reliability in EPU detection, traditional ML methods using labeled data have been developed.

Traditional ML Approaches: Relevant studies in this category include: Lolić et al. (2022) who applied ML techniques: linear regression, random forests, GBM, decision trees, extreme gradient boosting, and their ensembles for EPU classification [LSL22]. Tobback et al. (2018) used SVM to classify news articles related to EPU from Belgian news sources [TND⁺18]. Sadorsky (2022) employed various ML models such as random forests, extremely randomized trees, GBM, and Naïve Bayes for EPU classification [Sad22]. Wang et al. (2022) utilized neural networks, GBM, a dictionary-based approach, and ensemble classifiers, to classify EPU in Chinese news articles [WLZF22]. The reliance of traditional ML on manual feature engineering and limited capacity to capture complex language patterns has motivated a shift towards deep learning models and LLMs for more advanced text understanding.

Deep Learning & LLM Approaches: Representative studies include: Rao et al. (2024) who used LSTMs, GRUs, and multi-layer perceptrons to classify EPU in news articles

3. RELATED WORK

[RTMS24]. Chen et al. (2024) applied word embeddings based on Skip-Gram along with GRU, Convolution Neural Network (CNN), and capsule networks for EPU classification tasks [CHY24]. Qureshi et al. (2021) utilized BERT models, keyword extraction algorithms, and a combination of RoBERTa and sentence transformers to extract and classify EPU related keywords [QCDD21]. Keith et al. (2020) combined keyword matching, supervised ML techniques (logistic regression with BoW and BERT embeddings), and prevalence estimation techniques such as probabilistic classify and count, and implicit likelihood to classify news articles related to EPU [KTOM20].

Chang et al. (2024) leveraged BERT to classify trade policy uncertainty based on texts from management discussion sections of annual reports from Chinese public firms [CHLS24]. Audrino et al. (2024) adapted various LLMs, including GPT-4, LLAMA, and Zephyr-7B for EPU classification [AGS24]. Yeh et al. (2024) applied LLMs such as GPT-3.5-Turbo, GPT-4, and Claude3 to generate keywords for EPU classification [YHPC24]. Ito et al. (2024) fine-tuned LLMs to detect EPU, contributing to the development of Japan's monetary policy index for the period between 2015 and 2016 [ISO25]. Although these methods also used LLMs like the approaches proposed in this Thesis, their reliance on costly human-curated labels limits their applicability.

Research Gap: Numerous previous studies discussed focused on unsupervised ML approaches, primarily using topic models and keyword matching. While these methods are simple and do not require training, they are less effective than supervised ML techniques. In contrast, supervised ML methods, including traditional ML and deep learning, rely on human-curated datasets, making them less scalable. Constructing a new index for a different year, country, or region requires training new models and generating new labeled datasets, which is resource-intensive. Furthermore, prior research has been limited to binary classification, identifying only whether an article contains EPU or not, with no existing studies attempting to classify specific types of EPU within an article.

In this Thesis, chapter 4 presents our proposal to use LLMs for classifying news articles containing EPU. Instead of relying on human-curated datasets, which are costly to obtain, it proposes a weakly supervised learning approach. LLMs are leveraged to generate training labels or directly classify news articles through prompting. The labels obtained via prompting can either serve as the final classification labels or be refined using ensemble aggregation techniques with multiple LLMs to reduce label noise. Furthermore, this Thesis proposes extending beyond binary classification by adapting LLMs for multi-label and hierarchical classification of EPU-related news articles, an aspect not previously explored in existing research.

3.2 Citizen Feedback Analysis with Large Language Models

Across various government sectors such as defense, transportation, finance, and health care, a shared challenge arises: extracting actionable insights from the extensive datasets they gather to inform decisions and improve service delivery. Central to this data is citizen feedback, whether sourced from suggestion boxes, online surveys, or social media. This type of feedback offers a valuable perspective to enable government agencies to understand citizens' concerns and viewpoints [MR18]. By understanding these insights, agencies can better tailor their products and services to align with public needs and expectations [Kow20, SM22].

Traditional methods for collecting feedback such as surveys, town hall meetings, and suggestion boxes are resource intensive, time consuming, and costly. Large scale surveys require substantial investment making them infrequent for many government agencies. Furthermore, even when these efforts are undertaken, the focus is mainly on structured data, often neglecting the valuable insights that could be extracted from unstructured textual feedback [KEJM20].

The vast resource provided by the digital revolution offer public agencies with a valuable opportunity to make informed decisions without relying solely on traditional surveys. For instance, government agencies providing online services can integrate feedback forms into their digital interactions or websites. Additionally, they can extract valuable insights from public commentary on platforms like X and Facebook [MH12, Har12, Fel14, SC15, KS16].

These resources hold the potential to improve and optimize services. However, much of this data is unstructured, primarily in text form, making manual analysis impractical, while conventional analytical methods often struggle with its complexity and high dimensionality [BAD⁺12]. In such cases, automated approaches, particularly from NLP, are being utilized to analyze and derive insights from these large datasets.

There is relatively limited research on citizen feedback, particularly using real-world data, as such data is often not publicly available or may not be collected at all. Most existing studies rely on data from public platforms. Although this Thesis focuses on citizen feedback, our review also includes research on a related area: analyzing customer complaints in the private sector, which is a more developed area. To structure the discussion, the Thesis categorizes related work into unsupervised ML, traditional ML, deep learning, and LLM approaches, summarized in Table 3.2 and described next.

Unsupervised ML approaches: Unsupervised ML techniques have been widely ap-

Table 3.2: Citizen feedback and customer support relevant works

Category	Relevant Works
Unsupervised ML	[FB16], [Kow20], [KZYY18], [KYP ⁺ 21], [FSLRJ24]
Traditional ML	[TMT ⁺ 25], [AT14], [HHI21], [VT24], [SAP ⁺ 24], [AZFSK22], [GVPP22], [WMS22], [TB24], [SM22], [FK14], [FBR ⁺ 19], [YAKMS24]
Deep Learning	[MLL ⁺ 21], [MZW18], [ASN ⁺ 23], [DMG23]
LLMs	[ZMS ⁺ 23], [LBJ23], [Eng24], [PŞAS24], [BPC ⁺ 24], [BKYS21], [WM24], [APG ⁺ 23], [FSL22], [DSB21], [FSLRJ24], [SVCR24], [KDJ ⁺ 24], [PYC ⁺ 24], [XCG ⁺ 24]

plied to analyze and extract patterns from large-scale citizen feedback and complaint datasets. Among these techniques, topic modeling has been the most commonly used approach. Relevant studies in this category include: Forte and Brazdil (2016) who analyzed sentiment polarity in customer service feedback from a major Portuguese telecommunications company using a lexicon-based approach enhanced with domain-specific terms and rules [FB16]. Kowalski (2020) explored the use of topic modeling to extract insights from citizen comments on public services, demonstrating how such methods enable real-time performance measurement of public sentiment regarding general practice services in England [Kow20].

Kalyoncu et al. (2018) used LDA to analyze customer complaints in Turkey’s mobile network sector, developing an interactive visualization tool to help operators understand subscribers’ concerns [KZYY18]. Kim et al. (2021) proposed a dynamic topic modeling approach to forecast public demand trends based on 160,000 civic queries submitted through a smart city participation platform in Seoul [KYP⁺21]. Fu et al. (2024) utilized LLMs and topic models, specifically ChatGPT, to analyze public feedback topics collected from online submissions in New Zealand [FSLRJ24]. In addition to uncovering patterns and themes within feedback using unsupervised ML techniques, classification of this text data is essential for efficient re-routing to the appropriate personnel or departments. We next discuss relevant works employing traditional ML approaches for this task.

Traditional ML Approaches: Studies in this category include: Taha et al. (2025) who developed a web-based smart ticketing system integrating Random Forests and GBM to categorize and prioritize customer inquiries, significantly reducing response times [TMT⁺25]. Altintas and Tantug (2014) proposed an automated ticket classification framework using SVM, Naïve Bayes, and K-Nearest Neighbors (KNN), achieving improvements in ticket assignment efficiency [AT14]. Similarly, Harun et al. (2021) designed a supervised learning model to classify helpdesk support queries using SVM and Naïve Bayes, effectively reducing misclassification rates in ticket routing [HHI21].

Several other studies have focused on refining classification methodologies. Varghese and Tian (2024) proposed a knowledge-graph embedding ranking system to enhance support ticket resolution [VT24]. Sunil et al. (2024) applied TF-IDF vectorization and Word2Vec embeddings with a logistic regression model for customer complaint classification [SAP⁺24]. Ali et al. (2022) conducted a comparative analysis of five ML classifiers, including SVM, Naïve Bayes, Decision Trees, Random Forests, and KNN, identifying SVM as the most effective approach for multi-class text classification in customer service systems [AZFSK22].

Gandla et al. (2022) applied logistic regression, Decision Trees, and SVM to build an ML-based ticket classification system, improving ticket tagging and incident management [GVPP22]. Wahba et al. (2022) proposed a hybrid SVM model that integrated handcrafted rules to enhance classification accuracy, particularly in datasets with overlapping ticket categories [WMS22]. Truss and Boehm (2024) examined the use of Automated Machine Learning (AutoML) for ticket classification, making AI-driven solutions more accessible to non-experts [TB24]. Sangil and Maceda (2022) utilized NLP and association analysis to assess citizen feedback on local government initiatives [SM22].

Fauzan and Khodra (2014) employed learning-to-rank models for classifying government complaint texts [FK14]. Filgueiras et al. (2019) applied multiple ML classifiers to analyze citizen complaints related to economic and food safety regulations [FBR⁺19]. Yiugit et al. (2024) conducted sentiment classification on municipal service reviews using an SVM to assess corporate reputation [YAKMS24]. As previously noted, traditional ML models, although interpretable, typically rely on manual feature engineering and often underperform compared to deep learning methods, which automatically extract features and utilize deep architectures to learn complex patterns. This motivates the use of deep learning approaches for this task, with relevant studies discussed next.

Deep Learning Approaches: Relevant studies in this category include: Meng et al. (2021) who proposed a hierarchical deep learning model that integrates word- and character-level representations for classifying customer service tickets in the power sector, significantly outperforming traditional ML models [MLL⁺21]. Molino et al. (2018) proposed a ticket classification system utilizing CNN and RNN architectures to enhance the accuracy and efficiency of automated customer support responses [MZW18]. Air-langga et al. (2023) applied LSTM and GRU models to analyze citizen feedback on Jakarta's government services, demonstrating deep learning's superiority over traditional methods [ASN⁺23].

Dashdorj et al. (2023) developed a Mongolian-language feedback classification system using LLM and CNN models, showcasing the potential of deep learning for low-resource language processing [DMG23]. The deep learning methods discussed in this section

train models from scratch, but the introduction of transfer learning and pre-training has motivated the use of pre-trained LLMs, which start from pre-trained weights and require significantly less data through fine-tuning or prompting. Techniques in this category adopted for citizen feedback are discussed next.

LLM approaches: Relevant studies in this category include: Zangari et al. (2023) who developed a BERT-based ticket classification system, demonstrating that transformer models significantly improve classification accuracy [ZMS⁺23]. Liu et al. (2023) proposed "Ticket-BERT", a fine-tuned version of BERT specifically designed for incident ticket categorization [LBJ23]. Engelbrechtsmuller (2024) compared fine-tuned GPT-2 and BERT models for ticket classification in an IT helpdesk environment, finding that GPT-2 outperformed BERT in four out of five classification tasks [Eng24].

Paket et al. (2024) investigated LLM-based ticket classification using fine-tuned BERT models, demonstrating their effectiveness in multilingual IT support systems [PŞAS24]. Bonechi et al. (2024) compared BERT-based models with traditional ML techniques including SVM classifiers, concluding that BERT consistently achieved higher classification accuracy in banking customer service applications [BPC⁺24].

Bannihatti et al. (2021) proposed SupportNet, a BERT-based model designed for summarizing customer support tickets and extracting key information, leading to improved response efficiency [BKYS21]. Wulf and Meierhofer (2024) investigated the use of LLMs in automating technical customer service, integrating RAG to enhance text retrieval and classification [WM24]. Arici et al. (2023) utilized GPT-4 to automate ticket assignment in an Italian IT service application, optimizing the allocation process [APG⁺23].

Feng et al. (2022) proposed TaDaa, a deep-learning-based system fine-tuned with BERT, Distilled BERT (DistilBERT), and RoBERTa to enhance ticket classification accuracy and response times [FSL22]. Dong et al. (2021) implemented a semi-supervised learning approach by combining BERT and A Lite BERT (ALBERT) to classify customer contact intents with minimal labeled data [DSB21]. Fu et al. (2024) further validated the effectiveness of LLMs by comparing ChatGPT's classification performance against traditional NLP techniques in analyzing citizen feedback [FSLRJ24].

In another application, Subbarao et al. (2024) trained deep neural networks on a dialogue-based dataset for ticket resolution, demonstrating the effectiveness of LLMs in conversational AI applications [SVCR24]. Koukal et al. (2024) explored ticket category balancing strategies for fine-tuned BERT models [KDJ⁺24]. Xu et al. (2024) integrated knowledge graphs with RAG to enhance customer service question answering, reducing ticket resolution times [XCG⁺24]. The research conducted in this Thesis mainly utilized LLMs for analysis of citizen feedback, and the research gaps in the reviewed

LLM approaches are discussed next.

Research Gap: Most of the reviewed research on unsupervised learning for analyzing customer and citizen complaints primarily relies on topic modeling techniques such as LDA and BERTopic. However, these methods do not account for the relationship between metadata and the discovered topics. This Thesis addresses this gap by conducting an extensive statistical and unstructured analysis to uncover hidden patterns in citizen feedback using real complaint data from a government organization. Additionally, it incorporates Structural Topic Model (STM) to analyze textual feedback alongside metadata, providing deeper insights into citizen inquiries.

Several reviewed studies have also utilized traditional ML approaches for classifying and analyzing customer complaints and citizen feedback. Among those that incorporated LLMs, most have focused on fine-tuning earlier pre-trained models such as BERT, RoBERTa, and GPT-2. This Thesis builds on this related work by adapting the latest LLMs, including LLAMA, and comparing their performance with traditional ML methods and earlier LLMs used in prior studies for predicting categories and subcategories of citizen feedback.

Existing research on prompt-based approaches has primarily relied on zero-shot prompting, often using closed-source LLMs such as GPT-4. However, no prior work has explored the use of open-source LLMs for prompting, despite potential concerns about data privacy, security, and dependence on external APIs in government applications. This Thesis proposes the adaptation and use of open-source LLMs that can be hosted locally, ensuring greater control over data.

Additionally, unlike previous studies that focus solely on zero-shot prompting, this Thesis incorporates few-shot prompting to assess its impact on classification performance. While most studies emphasize classification and citizen feedback analysis, this Thesis extends the scope by proposing and evaluating LLMs for summarization and question answering in citizen feedback processing.

3.3 Large Language Models for Analysis of Public Sector Document Repositories

Beyond citizen feedback, complaints, and general communication, text is widely used across the public sector. Politicians and government entities communicate their goals and perspectives through written text on digital platforms including campaigns, debates, and policy announcements. Additionally, government agencies oversee vast collections of documents such as legislative proposals, policy briefs, administrative

records, and historical archives, which are primarily written in text format or a combination of text and images [JM23].

To enhance transparency and improve accessibility, many government agencies have transitioned their services online, and also have their policies, regulations, and procedural guidelines available on their official websites [AMGCM20]. Citizens often rely on these digital platforms to access information on various government services and announcements. However, the sheer volume of online documents makes manual searching and navigation inefficient and time-consuming. A more effective solution involves leveraging automated techniques to extract key insights from this vast amount of information.

LLMs have recently shown great potential in analyzing extensive document collections and supporting a wide range of document processing tasks such as QA, summarization, and classification among others [BMR⁺20, DCLT19]. This potential has prompted significant investment from the private sector, resulting in their incorporation into various tools and products, enhancing customer engagement and boosting efficiency [TMS⁺23, Ope23]. In the public sector, however, adoption of LLMs has been more gradual despite of their potential.

The deployment of LLMs in the public sector can significantly change the way citizens engage with the government and the way the government attends to citizen demands. These capabilities make LLMs highly valuable for the public sector. Table 3.3 provides an overview of related studies on the use of LLMs and NLP for document processing in government and the public sector. Related work is categorized into fine-tuning, RAG, LLM agents, and prompting, along with related studies on hallucination and reliability.

Table 3.3: Related work on applying LLMs in public sector and government applications

Category	Relevant Work
Fine-tuning	[dPPdGC ⁺ 24], [GTDS ⁺ 24], [PMF ⁺ 23], [CGTB24], [HkLX ⁺ 24], [KB24]
RAG, Agents, and Prompts	[MKK ⁺ 23], [PAT ⁺ 24], [PSMT24], [FBH23], [KFFB24], [AADH ⁺ 22], [Lin23], [YTK24]
Hallucinations & Reliability	[Dai24], [JPWK23], [SHB ⁺ 24], [AM19], [CGH24]

Fine-tuning Approaches: Several studies have used fine-tuning approaches for LLMs to enhance their applicability in government and administrative tasks. Relevant studies include: Han et al. (2024) who proposed a domain-specific LLM trained on government service counseling data collected from Beijing’s official websites, significantly improving the accuracy of automated citizen support [HkLX⁺24]. Paiva et al. (2024) fine-tuned LLAMA2 and Mistral models through continued pre-training on Portuguese government texts, demonstrating improved product identification in textual purchase descriptions

[dPPdGC⁺24]. Gesnouin et al. (2024) introduced LLaMandement, a fine-tuned model developed by the French government to summarize legislative proposals, improving efficiency in processing parliamentary session documents [GTDS⁺24].

Peña et al. (2023) applied LLMs, including RoBERTa and GPT-2, to classify Spanish parliamentary documents into 30 different topics, showing their effectiveness in processing domain-specific legislative data [PMF⁺23]. Caudron et al. (2024) investigated domain adaptation of BERT and RoBERTa for clustering pledges on the transition pathway for tourism in the European public sector, revealing enhancements in interpretability and usability [CGTB24].

Kleiman and Barbosa (2024) explored the use of a chatbot, ChatPGD trained on curated government documents to assist with the management and performance program in Brazil, demonstrating improved response times and effectiveness [KB24]. Fine-tuning approaches discussed in this section often require continuous model training as new data becomes available, which demands significant hardware resources. Next, we discuss non-parametric methods such as RAG and LLM agents that update LLM knowledge at inference without requiring retraining.

RAG, Agents, and Prompting: Relevant studies in this category include: Mamalis et al. (2023) who investigated augmenting GPT-3.5-Turbo with the Scottish Open Statistics Portal to enable real-time statistical queries, assessing its effectiveness in open government data retrieval [MKK⁺23]. Pereira et al. (2024) introduced INACIA, a system integrating LLMs into the Brazilian federal court of accounts for automated case analysis, information extraction, and legal plausibility assessment [PAT⁺24]. Papageorgiou et al. (2024) proposed a modular RAG-based architecture for deploying LLM-powered virtual assistants in e-government, ensuring scalability and ethical compliance while utilizing Organisation for Economic Co-operation and Development (OECD) documents for experimentation [PSMT24].

Følstad and Bjerkreim (2023) analyzed 2,663 chatbot dialogues from Norwegian municipalities and identified patterns of user interaction that contributed to utility-oriented responses and improved chatbot efficiency [FBH23]. Kurian et al. (2024) developed GovInfoHub, an RNN-based chatbot trained on government schemes, insurance policies, and identity card details to provide structured, real-time assistance for citizens [KFFB24]. Alhalabi et al. (2022) introduced an AI-driven M-Government mobile application in the United Arab Emirates, integrating Telegram support to improve accessibility and streamline various government services [AADH⁺22].

Lin (2023) conducted a comparative study on Guangdong Province's government chatbots and ChatGPT concluding that chatbots remain in the exploratory phase, with opportunities for optimization [Lin23]. Yun et al. (2024) surveyed 128 government

officials in Korea regarding AI adoption in e-government, revealing AI's potential to improve efficiency and responsiveness in administrative tasks [YTK24]. The studies discussed so far on RAG and LLM agents focus on reducing hallucination by augmenting LLMs with updated information. Despite this, LLMs can still produce hallucinations which are outputs with errors or incorrect information. We next review approaches that have identified or attempted to mitigate these challenges for government applications.

Hallucinations and Reliability: Relevant studies include: Dai (2024) who conducted a survey on LLM applications in smart government, emphasizing the necessity of regulated AI applications and robust data protection policies to mitigate misinformation risks [Dai24]. Jiang et al. (2023) performed a systematic review of NLP adoption in governments, highlighting the limited empirical studies on AI reliability [JPWK23].

Straub et al. (2024) examined AI-driven automation in the United Kingdom government, estimating that 84% of bureaucratic transactions could be automated, but warning of potential hallucinations in AI-driven decision-making [SHB⁺24]. Al-Mushayt (2019) proposed deep learning models based on multi-layer perceptrons for e-government automation, identifying challenges in AI reliability and accuracy [AM19]. Chen and Gasco-Hernandez (2024) conducted an empirical study on AI chatbot deployments in the United States of America state government, finding that the lack of empirical chatbot evaluations has hindered understanding of chatbot effectiveness and the impact of hallucinations in real-world applications [CGH24]. None of the reviewed studies implement hallucination detection for public sector chatbots, unlike the work presented in this Thesis.

Research Gap: While previous studies on fine-tuning LLMs have used public sector documents, none have specifically focused on government services, procedural details, and press releases, which are the key areas covered in this Thesis. Additionally, there is a lack of open-source, curated datasets available for training and evaluating LLMs in this domain. Some related works have explored the use of agents and RAG, but their applications were limited to specific documents, and no detailed evaluation of RAG components was conducted.

Moreover, past research has primarily focused on text-only LLMs, even though government documents often contain both text and images. Another major challenge in using LLMs in the public sector is the issue of hallucinations, where models generate incorrect or misleading information. While previous studies acknowledge this problem, none have empirically measured hallucination rates in this domain or systematically evaluated methods for detecting them.

To address these gaps, chapter 6 of this Thesis uses both unimodal (text-only) and multimodal (vision-language) LLMs to process public sector documents. The datasets

were collected from publicly available Irish government sources, including the official government publications portal, health information services, citizens information, and the central statistics office. This Thesis contributes by creating large-scale datasets for training and evaluating LLMs in the government sector, including instruction-answer examples, a multimodal dataset with vision-language samples, and datasets with different LLM responses to human-curated questions, along with a human evaluation of these responses.

In addition to creating datasets, this Thesis develops and evaluates RAG systems by testing various components and hyper-parameters, including chunk size, embedding models, generation models, and chunk overlap. Unlike previous studies that focused only on text-based models, this research evaluates and adapts multimodal LLMs for government documents that include both text and images. Additionally, while earlier research has implemented LLM agents, this Thesis conducts a detailed evaluation of their effectiveness in using different tools, with a human-curated dataset created to assess their performance in government applications. Finally, in chapter 7, this Thesis goes beyond simply recognizing hallucinations as a challenge. Instead, it conducts an empirical evaluation of different hallucination detection methods and develops datasets specifically designed to measure hallucination rates in LLM responses.

The next chapters, starting with chapter 4, discuss the methodology and results of the proposed approaches aimed at addressing the identified gaps in the literature and answering the research questions raised in this Thesis.

Chapter 4

Weakly Supervised Learning for Predicting Economic Policy Uncertainty from News Articles

This chapter describes the adaptation of LLMs for economic monitoring in the public sector to automatically determine whether an article discusses EPU and to identify its specific type. Previous studies either rely on keywords, which often result in a high count of false positives, or use ML approaches that require a large number of quality human labeled data that is costly and time consuming to acquire. This chapter proposes approaches based on weak supervision techniques, using generative LLMs to create synthetic labels through prompting, making the approach both cost-effective and scalable. Additionally, it discusses the proposed methods and results for multi-label and hierarchical classification of articles related to EPU. Some of the experiments and methodologies discussed in this chapter were published in the Springer Nature Journal Neural Computing and Applications, in an article titled: *Understanding the Influence of News on Society Decision Making: Application to Economic Policy Uncertainty*.

4.1 Methodology

This section describes the methodology used for analyzing and classifying news articles related to EPU. The process begins with EPU classification, where weakly supervised labels generated by LLMs are compared with human annotated labels. Next, multi-label text classification is applied to identify different types of EPU present in the articles, with labels assigned only to articles classified as EPU-related. Since this classification follows a hierarchical structure, additional experiments are conducted to assess whether incorporating this hierarchy enhances classification performance.

4.1.1 Weakly supervised EPU classification

This chapter starts by discussing the weakly supervised approach used for EPU classification. We consider a dataset D^U of n unlabeled data points: $D^U = \{x_i\}_{i=1}^n$. The goal of text classification is to assign each data point x_i to a label y_i where each label y_i belongs to one of m possible classes: $C = \{c_1, \dots, c_m\}$, resulting in a labeled dataset: $D = \{(x_i, y_i)\}_{i=1}^n$. Unlike traditional supervised learning, which relies on large volumes of human labeled data, weak supervision relies on synthetic labels generated by LLMs to train the classifiers. However, a human labeled dataset is used during evaluation to accurately assess the model's performance.

We formulate the EPU classification as a binary classification task with two target classes: $C = \{"No EPU", "EPU"\}$. The proposed approach comprises three main stages. In the initial stage, LLMs are used to generate labels via prompting, guided by instructions embedded in the prompt illustrated in Figure 4.1. This prompt design is inspired by the annotation guidelines developed by Baker et al. (2016) for human annotators [BBD16].

While the work by Paul et al. (2023) [TzM23], which this chapter builds on employed multiple labeling functions such as keyword lookups and semantic similarity, the results presented here are based solely on LLM-based prompting. A single prompt (Figure 4.1) is used to encode the labeling logic, serving as a soft rule-based alternative that simplifies the process and leverages recent advances in LLM capabilities.

As a result, a label matrix is automatically constructed, where each news article receives multiple weak labels from k LLMs represented as: $\lambda = \{\lambda_1, \dots, \lambda_k\}$. For a given LLM λ_j , the output is $\lambda_j(x_i) \in \{0, 1\}$, where 1 corresponds to an "EPU" label and 0 corresponds to a "No EPU" label. Since different LLMs may assign different labels to the same input x_i , the second stage focuses on aggregating the k generated synthetic labels to produce a less noisy single label. To accomplish this, various ensemble methods are employed which are detailed in Section 4.1.1.1.

Label Generation Prompt

Given a news article or passage, determine whether it discusses Economic Policy Uncertainty (EPU).

If it does, classify it as EPU=1; otherwise, classify it as EPU=0.

Classification Criteria:

1. General Economic Uncertainty

- The article must mention uncertainty about economic conditions (examples include; job market fluctuations, inflation concerns, recession fears).
- If the article only discusses general economic uncertainty but does not link it to government policy, classify as EPU=0.

2. Policy-Related Economic Uncertainty

- The article must explicitly connect economic uncertainty to government policy actions, decisions, or inactions.
- Mentions of economic policy uncertainty in the past, present, or future all qualify for EPU=1.

Further Indicators for EPU=1 Classification:

The article should discuss uncertainty related to policy decisions in any of the following areas:

- Monetary policy (examples include; central bank actions, interest rates, money supply)
- Fiscal policy (examples include; government spending, tax reforms)
- Regulations (examples include; labor laws, environmental policies, financial regulations)
- Trade policy (examples include; tariffs, trade agreements, export/import restrictions)
- Government leadership uncertainty (examples include; elections, political conflicts)
- Sovereign debt and currency stability (examples include; exchange rate policies, foreign reserves)
- National security and terrorism (if linked to economic uncertainty)
- Social safety nets and entitlement programs (examples include; welfare, healthcare policies)

Figure 4.1: EPU label generation prompt

4.1.1.1 Weak Label Aggregation

In this stage, various aggregation methods including: Data Programming (DP), Majority Vote, Worker Agreement with Aggregate (WAWA), and the Dawid-Skene model, are used to consolidate multiple noisy labels generated by different LLMs into a single label.

- **Majority Vote:** Majority Vote [Pen46] is a simple yet powerful model often used in ensemble learning techniques. Consider a setting with k classifiers in our case the output of k LLMs on the same input text, each providing a vote for one of the possible classes $c_i \in C$ in a classification task. The Majority Vote is determined by aggregating these votes and selecting the class with the highest number of votes.

If we let v_{jc_i} to be a binary indicator where $v_{jc_i} = 1$ if classifier λ_j votes for class c_i , and $v_{jc_i} = 0$ otherwise. The total number of votes for class c_i is then given by: $V_{c_i} = \sum_{j=1}^k v_{jc_i}$. The predicted class $\hat{y}$ is the one with the maximum V_{c_i} shown in Equation 4.1.

$$\hat{y} = \arg \max_{c_i} V_{c_i} \quad (4.1)$$

This method is based on the idea that combining outputs from multiple classifiers can produce more accurate results, especially when each classifier brings different and complementary strengths to the task.

- **DP:** DP [RDSW⁺16] is an aggregation method that assigns a *noisy* label to each news article in an unsupervised learning manner. The outputs generated by LLMs through prompting can be viewed as multiple annotators labeling the same article, similar to a crowd-sourcing setup [MKBH21]. These labels may contain conflicts and correlations, even when assigned by domain experts.

Instead of a simple Majority Vote, DP employs a probabilistic framework to capture the structure and relationships within the label matrix. The probabilistic model $P_\theta(\Lambda, Y)$ is defined as the joint probability of the LLMs' outputs (label matrix) Λ and the latent (unobserved) true class labels Y . The label matrix is encoded using a factor model with three factor types representing conflicts, correlations, and propensity (instances where LLMs generate valid labels).

The generative model is expressed in Equation 4.2.

$$P_\theta(\Lambda, Y) = Z_\theta^{-1} \exp\left(\sum_{i=1}^n \theta^T \phi_i(\Lambda_i, y_i)\right) \quad (4.2)$$

where Z_θ is the normalizing constant, $\phi_i(\Lambda_i, y_i)$ represents the concatenated feature vector derived from all LLM outputs for a given sample news article x_i , and θ denotes the parameter vector. The estimate of the true parameter denoted as $\hat{\theta}$, is obtained by minimizing the negative marginal likelihood using the observed label matrix Λ , relying solely on the patterns of agreement and disagreement within Λ since the true labels are unavailable.

The estimated parameters are then used to produce probabilistic noisy labels:

$\hat{Y} = P_{\hat{\theta}}(Y \mid \Lambda)$, which can optionally serve as training labels for downstream models.

- **WAWA:** The WAWA algorithm [UPT24] aggregates noisy labels through a three-step process. It begins by determining the Majority Vote label for each task. Then, it assesses each worker’s skill by measuring the proportion of their responses that align with the Majority Vote. Finally, it computes a weighted Majority Vote, where the weights are derived from these skill estimates. The algorithm is based on the premise that responses agreeing with the Majority Vote are more likely to be accurate, and that a worker’s reliability can be inferred from the consistency of their answers with the majority.
- **Dawid-Skene:** The Dawid-Skene aggregation model [DS79] is a probabilistic method used to estimate the expertise level of workers (in this case LLMs) in a crowd-sourcing environment. This model evaluates the reliability of each worker by constructing a confusion matrix that captures the probability of correctly or incorrectly labeling. The confusion matrix provides insight into how consistently a worker assigns labels in a classification problem.

In addition to individual worker reliability, Dawid-Skene incorporates a prior class probability vector, which represents the likelihood of each class occurring before considering the responses provided by workers. Since the true label of a task is unknown, it is treated as a latent variable that the model seeks to estimate. The probability of a task belonging to a specific class is influenced by the prior probabilities and the observed responses from workers. The optimization of these parameters, including the prior class probabilities, worker confusion matrices, and latent variables representing true labels, is achieved using the expectation maximization algorithm.

4.1.1.2 Discriminative Model

The aggregated labels were used to fine-tune pre-trained LLMs: BERT and RoBERTa, and to train an LSTM model. For pre-trained LLMs, this was achieved by adding a feed-forward neural network and a classification head to the last layers while keeping the other pre-trained layers frozen to facilitate adaptation to our downstream EPU classification task.

For an LSTM, the news article text was first tokenized and converted into sequences of integers. These sequences were then mapped into dense vector representations using an embedding layer, which captures semantic relationships between words. Initially, these embeddings are randomly initialized with a uniform distribution and are progressively updated during training. The transformed embeddings are then fed into the

LSTM layers, which learns to identify sequential dependencies and contextual relationships within the text.

For both LLMs and an LSTM, the final layer consists of a dense network with a softmax function, which assigns the text to the appropriate EPU category. The training process optimizes the model by minimizing the expected loss over noisy labels formulated in Equation 4.3 [RDSW⁺16].

$$\hat{\theta} = \arg \min_{\theta} \sum_{i=1}^n \mathbb{E}_{\hat{y}_i \sim \hat{Y}} [\mathcal{L}(h_{\theta}(x_i), \hat{y}_i)] \quad (4.3)$$

where $\hat{\theta}$ represents the estimated model parameters, n is the number of training examples, $\mathbb{E}$ denotes expectation, $\mathcal{L}$ is the expected loss with respect to $\hat{Y}$, $h_{\theta}(x_i)$ is the output predicted by the discriminative model, x_i is a training instance, and $\hat{y}_i$ is the corresponding noisy probabilistic label from $\hat{Y}$.

4.1.1.3 Multi-label Classification for EPU Articles

For articles positive for EPU, this section discusses the methodology used to identify specific subcategories, with their class distribution detailed in Table 4.1. In contrast to the binary EPU classification task, where each article receives a single label, the subcategories here may simultaneously belong to multiple classes selected from a predefined set: $\{a, b, \dots, p\}$. Each article is a multi-hot vector s_i , indicating the presence or absence of each corresponding subcategory.

In this section, experiments relied solely on human labeled data rather than synthetic labels. This decision stemmed from the absence of detailed guidelines in the labeling guide for assigning multiple labels to news articles, unlike the case with EPU classification. Consequently, it was challenging to develop clear and effective instructions for a reliable prompt-based method comparable to that used in EPU classification.

We used the same models as in EPU classification: BERT, RoBERTa, and LSTM. The text processing approach remained similar to that used for EPU classification, but the subcategories were converted into multi-hot vectors, denoted as s_i . Both models were trained using binary CE loss across all subcategories defined in Equation 4.4.

$$\mathcal{L}_{\text{multi}}(s_i, \hat{s}_i) = -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m \left[s_{ij} \log(\hat{s}_{ij}) + (1 - s_{ij}) \log(1 - \hat{s}_{ij}) \right] \quad (4.4)$$

where s_{ij} represents the true label for subcategory j of instance i , where 1 indicates that the article belongs to subcategory j and 0 indicates it does not. The term $\hat{s}_{ij}$ is the predicted probability that the article belongs to subcategory j , outputted by the model.

4.1.1.4 Hierarchical Classification for EPU Articles

Beyond multi-label classification, we performed hierarchical classification, where the classification task follows a structured dependency between an article’s general classification as EPU-related and its assignment to multiple subcategories. Unlike standard multi-label classification, where labels are treated as independent, hierarchical classification ensures that subcategories are only assigned when an article is identified as EPU-related.

The dataset is now structured as: $\{(x_i, y_i, \mathbf{s}_i)\}_{i=1}^n$, where x_i represents a news article, $y_i \in \{0, 1\}$ is a binary label indicating whether the article discusses EPU, and $\mathbf{s}_i$ is a multi-hot vector indicating the presence or absence of multiple subcategories. We experimented with various hierarchical classification approaches: basic summation, masking, weighted, and a two-stage, discussed next.

- **Basic Summation:** A single model is trained to predict both y_i and $\mathbf{s}_i$ simultaneously. It consists of a shared encoder followed by two classification heads: one for binary EPU classification and another for multi-label subcategory classification. For a given input text x_i , the model processes it through the shared encoder and applies separate linear classifiers for each prediction. The loss function used for training is defined in Equation 4.5.

$$\mathcal{L}_{\text{basic}}(y_i, \hat{y}_i, \mathbf{s}_i, \hat{\mathbf{s}}_i) = \mathcal{L}_{\text{binary}}(y_i, \hat{y}_i) + \mathcal{L}_{\text{multi}}(\mathbf{s}_i, \hat{\mathbf{s}}_i), \quad (4.5)$$

where $\mathcal{L}_{\text{multi}}$ is a multi-label loss, and $\mathcal{L}_{\text{binary}}$ is a binary CE loss for predicting y_i defined in Equation 4.6.

$$\mathcal{L}_{\text{binary}}(y_i, \hat{y}_i) = -\frac{1}{n} \sum_{i=1}^n [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)] \quad (4.6)$$

- **Masking:** To enforce dependency constraints between binary and multi-label classification, this approach incorporates a masking mechanism. The loss function for subcategory prediction is applied only when $y_i = 1$, that is when the article is classified as EPU-related. The modified loss function is defined in Equation 4.7.

$$\mathcal{L}_{\text{masking}}(y_i, \hat{y}_i, \mathbf{s}_i, \hat{\mathbf{s}}_i) = \mathcal{L}_{\text{binary}}(y_i, \hat{y}_i) + \mathbb{1}[y_i = 1] \cdot \mathcal{L}_{\text{multi}}(\mathbf{s}_i, \hat{\mathbf{s}}_i), \quad (4.7)$$

where $\mathbb{1}[y_i = 1]$ is an indicator function that applies the multi-label loss only to articles classified as EPU.

- **Weighted:** Building upon the masking-based approach, this method further mod-

ifies the multi-label loss by introducing a weighting scheme to handle cases where articles are mistakenly classified as non-EPU but contain some subcategory information. Instead of completely ignoring subcategory loss for $y_i = 0$, different penalty weights are assigned for false positives and false negatives. The weighted loss function $\mathcal{L}_W$ is defined in Equation 4.8.

$$\mathcal{L}_W(y_i, \hat{y}_i, \mathbf{s}_i, \hat{\mathbf{s}}_i) = \mathcal{L}_{\text{binary}}(y_i, \hat{y}_i) + \lambda_{\text{pos}} \mathbb{1}[y_i = 1] \mathcal{L}_{\text{multi}}(\mathbf{s}_i, \hat{\mathbf{s}}_i) + \lambda_{\text{neg}} \mathbb{1}[y_i = 0] \mathcal{L}_{\text{multi}}(\mathbf{s}_i, \hat{\mathbf{s}}_i) \quad (4.8)$$

where λ_{pos} and λ_{neg} are hyperparameters controlling the relative penalty for subcategory errors when y_i is 1 versus 0.

- **Two-stage:** Instead of jointly predicting y_i and $\mathbf{s}_i$, this approach separates the task into two stages: binary classification and multi-stage classification, trained separately unlike in the masking approach.

Binary Classification Stage: The first stage is a binary classifier trained solely on predicting whether an article discusses EPU. The model is optimized using the CE loss $\mathcal{L}_{\text{binary}}(y_i, \hat{y}_i)$ defined in Equation 4.6.

Multi-Label Classification Stage: After training the binary classifier, its predicted labels $\hat{y}_i$ are used to select articles for the second-stage multi-label classifier, which is applied only to samples where $\hat{y}_i = 1$. The loss function $\mathcal{L}_{\text{multi}}(\mathbf{s}_i, \hat{\mathbf{s}}_i)$ is the same as in standard multi-label classification but limited to this filtered subset.

4.1.2 Data

We used a dataset of 12,000 news articles provided by Baker et al. (2016), selected from USA news articles that included the terms “economy” and “uncertainty”. Domain experts assigned binary labels to these articles based on whether they discussed EPU. Among these, 4939 were identified as related to EPU. These articles were then classified into one or more multi-label subcategories, which explains why the total count in the class distribution shown in Table 4.1 is 6037 instead of 4939, as some articles fall into multiple subcategories.

The subcategories include: government spending (entitlement programs, social safety nets, welfare programs), trade policy (tariffs, quotas, trade rules), financial regulation (banking and equity markets), energy and environmental regulation (natural resources, commodities), sovereign debt (exchange rate policy, foreign reserves), political conflict and leadership changes (elections, governance crises), and legal policy (judicial decisions, regulatory changes). Further details about the dataset can be obtained from the

authors' website¹ [BBD16].

Table 4.1: Distribution of subcategories for news articles describing EPU

Subcategory	Count
Other policy matters that do not fit into pre-defined categories	1176
Political conflict and leadership changes	827
National security and terrorism	820
Trade policy (tariffs, quota, trade rules)	712
Taxes	659
Monetary policy	372
Government spending	367
Energy & environmental regulation, natural resources and commodities	187
Competition policy	182
Labor regulations	166
Legal policy	164
Sovereign Debt, exchange rate policy, foreign reserves	127
Fiscal policy	125
Financial regulation (including banking and equity markets)	100
Entitlement programs, social safety net, welfare programs	46
Health care programs and regulations	7
Grand Total	6037

4.1.3 Experimental Setup and Evaluation

All experiments were conducted in Python with BERT and RoBERTa being implemented using the Huggingface Transformer library². LSTM [HS97] was implemented using the Keras library³ [Ket17].

For the hyperparameters, the maximum text length was set to 512 tokens, the Adam optimizer was used, with a learning rate of 1×10^{-3} , a batch size of 50, and training was performed over 10 epochs. All model experiments were conducted on a single A100 40GB GPU machine.

For synthetic label generation, we primarily utilized generative LLMs, specifically open-source models from the LLAMA, Mistral, and Qwen families, using the prompt shown in Figure 4.1. The selected models represented the SoTA at the time of the experiments, that we could perform inference on with the hardware that was available to us. The

¹<https://www.policyuncertainty.com>

²<https://huggingface.co/>

³<https://keras.io/>

specific versions used in the experiments include: Mistral-24B, Qwen2-7B, LLAMA3-8B, Mistral3-7B, and Qwen2-14B.

Binary classification performance was evaluated using accuracy, F1 , Geometric Mean (GM), Area Under the Curve (AUC), and Area Under the Precision Recall Curve (AUPRC). Accuracy represents the overall proportion of correctly classified instances. F1 is the harmonic mean of precision and recall. GM accounts for both sensitivity and specificity, suitable under class imbalance setting. AUC measures the model’s ability to distinguish between classes across different threshold settings, while AUPRC assesses precision-recall trade-offs, making it particularly relevant for imbalanced data.

Multi-label classification performance was further assessed using Exact Match Ratio (EMR) and Label Ranking Precision (LRP) [ZZ13]. EMR calculates the percentage of instances where the predicted set of labels exactly matches the true set. LRP evaluates how well the model ranks correct labels higher than incorrect ones, focusing on the order of label relevance rather than requiring an exact match.

In addition to classification metrics, we used Cohen’s Kappa (CK) and Matthews Correlation Coefficient (MCC) to measure the correlation and agreement of different label generators with humans. CK [HF03] evaluates the level of agreement between the predicted and actual labels while accounting for the possibility of random agreement. MCC [CJ23], on the other hand, provides a balanced measure of correlation between predictions and ground truth, even when class distributions are uneven by considering all elements of the confusion matrix.

The code, datasets, and hyperparameters referenced in this chapter are available in the GitHub repository: <https://github.com/TrustPaul/Weak-supervision-in-Economic-Policy-Uncertainty>.

4.2 Results and Discussion

This section presents the results of the various experiments for binary EPU classification, as well as multi-label and hierarchical classification.

Table 4.2: Evaluation metrics (CK and MCC) for different label generators for the binary EPU classification task

Label Generator	CK	MCC
Majority Vote	-0.0121	-0.0139
WAWA	-0.0011	-0.0013
Mistral3-7B	0.0460	0.0553
LLAMA3-8B	0.2970	0.3058
Qwen2-14B	0.2849	0.3486
Qwen2-7B	0.2849	0.3486
Mistral-24B	0.3537	0.3766
DP	0.4156	0.4156
Dawid-Skene	0.4813	0.4908

Analysis of Different Label Generators: Table 4.2 presents evaluation metrics (MCC and CK) comparing the performance of models of different label generators. The evaluation includes individual LLMs such as Mistral-24B, Qwen2-14B, Qwen2-7B, LLAMA3-8B, and Mistral3-7B, as well as aggregation methods like Dawid-Skene, WAWA, and Majority Vote. Dawid-Skene achieved the highest correlation with human labels, with an MCC of 0.4908 and a CK of 0.4813, followed by the DP method, which obtained an MCC and CK of 0.4156.

Among individual LLMs, Mistral-24B showed the strongest correlation with human labels (MCC = 0.3766, CK = 0.3537), while Majority Vote had the lowest correlation. These results indicate that proper aggregation or the use of more effective models leads to higher agreement with human labels, and the high values of MCC and CK observed especially by top methods like Dawid-Skene, DP and Mistral-24B suggest that these labels can be reliably used for training other LLMs for this task.

Table 4.3: Performance evaluation metrics (F1, accuracy and GM) for different label generators for the binary EPU classification task

Label Generator	Accuracy	F1	GM
Mistral3-7B	42.06	27.36	41.29
Majority Vote	55.25	46.65	49.44
WAWA	55.41	47.69	49.95
LLAMA3-8B	64.15	42.85	56.09
Qwen2-14B	68.89	61.46	62.85
Qwen2-7B	68.89	61.46	62.85
Dawid-Skene	75.01	49.30	63.12
Mistral-24B	70.72	66.80	66.61
DP	71.83	70.78	70.77

Beyond correlation and agreement metrics, classification performance is of greater interest for this task. Table 4.3 therefore presents the performance of various data generators without additional training. For LLMs, this corresponds to zero-shot prompting performance. DP achieved the highest performance results, with a GM of 70.77% and an F1 score of 70.78%, followed by Mistral-24B, which obtained a GM of 66.61% and an F1 score of 66.80%. The lowest performance was observed with Mistral3-7B and the Majority Vote method. Based on these results, labels generated by DP were selected as synthetic labels for training other models.

The findings indicate that ensembling improves performance, but the choice of aggregation method significantly impacts the outcome. The zero-shot performance of LLMs like Mistral-24B suggests that sufficiently large and high-performing models can achieve reasonable classification results through prompting alone. However, even in cases where zero-shot prompting performs well, achieving such results requires models with at least 7 billion parameters, which come with significant inference costs. Therefore, it remains beneficial to use labels generated by these LLMs or aggregation methods in a distillation process to train smaller models, such as BERT, which offer lower training and inference costs.

Fine-tuning results for binary classification of EPU: Table 4.4 presents the performance of different models along with their training data sources, which include human-labeled data and weak supervision (WS) where labels are generated through aggregation. The models compared are LSTM, BERT, and RoBERTa.

In general, models perform better when trained on human labeled data. This is particularly evident with BERT, which achieves an F1 score of 74.66% and a GM of 73.77% when trained on human labeled data, compared to an F1 score of 71.92% and a GM

of 71.66% when trained on synthetic labels. A similar trend is observed with RoBERTa, which performs almost as well as BERT but achieves slightly higher F1 and GM scores when trained on synthetic data. LSTM has the lowest performance on both datasets, showing only minor differences in F1 and GM scores between human labeled and synthetic data.

Table 4.4: Performance comparison of different models and datasets on EPU binary classification. Dataset WS in the case means that labels generated by DP were used to train the respective model

Dataset	Model	Accuracy	F1	GM	AUC	AUPRC
Human	BERT	75.99	74.66	73.77	0.8423	0.7750
Human	RoBERTa	74.97	73.54	72.59	0.8440	0.7833
Human	LSTM	63.41	60.79	58.96	0.6452	0.5352
WS	DP	71.83	70.78	70.55	0.7338	0.6654
WS	LSTM	63.46	62.10	61.66	0.6482	0.5359
WS	BERT	71.93	71.33	71.83	0.7859	0.6877
WS	RoBERTa	73.03	71.92	71.66	0.7936	0.7114

The results in Table 4.4 further reveal that models trained on synthetically generated data such as RoBERTa, outperform the zero-shot prompting performance of the LLM used to generate the synthetic data. RoBERTa achieves a higher GM and F1 score than DP and also exceeds the best zero-shot prompting LLM: Mistral-24B, in terms of GM (71.66% versus 66.61%).

Although the performance of zero-shot prompting and aggregation methods is close to that of fine-tuned models, the competitive results are mainly achieved using extremely large LLMs, as previously noted. Despite this, using LLMs to generate synthetic labels for training smaller models remains a promising approach, as human annotation is costly and time-consuming, making synthetic data generation a more efficient and scalable alternative.

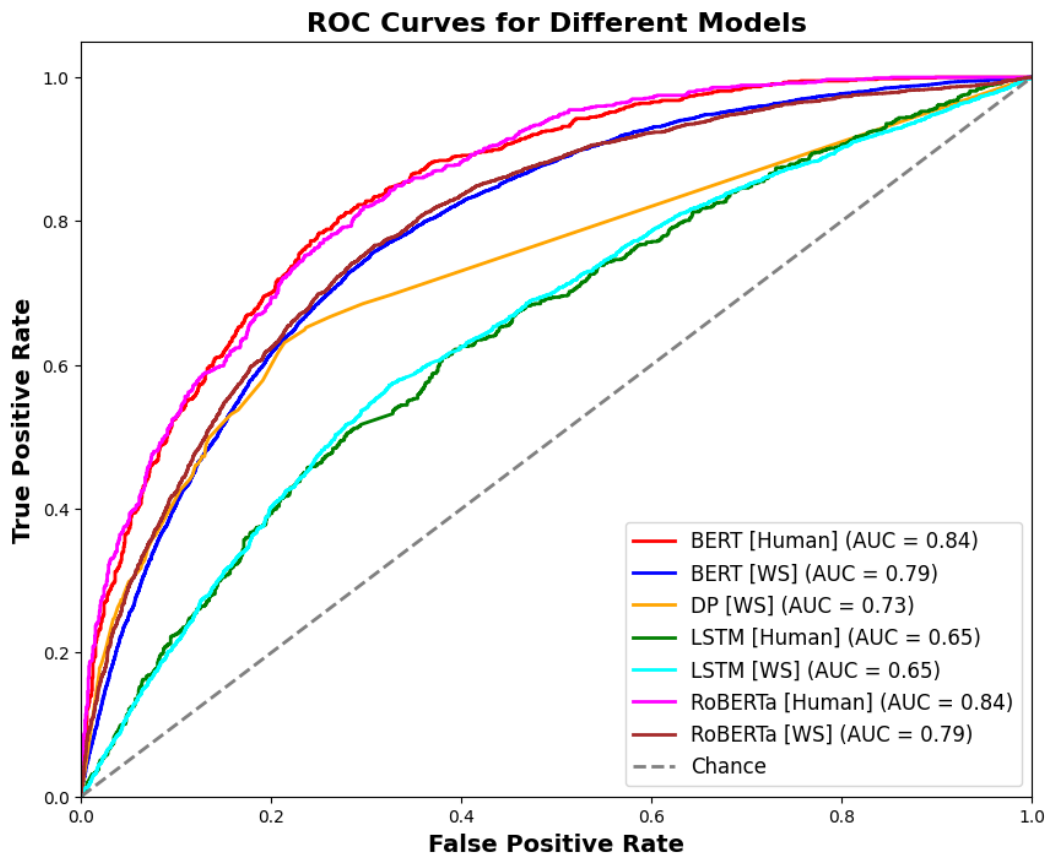


Figure 4.2: Receiver Operating Curves (ROC) curves for different models for binary EPU classification task. WS in the case means that labels generated by DP were used to train the respective model.

In addition to the classification metrics in Table 4.4, Figure 4.2 presents the ROC curves for different models. The trends in the ROC align with those observed in the F1 and GM scores. Models trained on human labeled data generally achieve higher AUC values, with both RoBERTa and BERT reaching approximately 0.84. In contrast, LSTM shows consistently lower AUC values, with little difference between models trained on weak supervision and those trained on human labeled data.

Table 4.5: Performance metrics for different models on the multi-label text classification task

Model	EMR	LRP	AUPRC	AUC
LSTM	58.15	75.40	0.0466	0.5931
BERT	61.70	83.70	0.1634	0.7727
RoBERTa	60.43	83.78	0.1676	0.7930

Multi-Label Text Classification: For EPU binary classification, articles identified as positive for EPU were further assigned to 16 multi-label categories, with the class distribution detailed in Table 4.1. For multi-label classification, only human labeled data was

used for fine-tuning. The models evaluated include: RoBERTa, BERT, and LSTM, assessed using multi-label evaluation metrics such as EMR, LRP, AUC, and AUPRC shown in Table 4.5.

In terms of EMR, all three models exhibited comparable performance, with BERT achieving the highest score. LSTM recorded an EMR of 58.15%, while BERT and RoBERTa attained 61.70% and 60.43%, respectively. For the LRP metric, BERT and RoBERTa performed similarly, both reaching approximately 83.70%, whereas LSTM had a lower LRP score of 75.40%.

The AUPRC values were generally low across all models, with RoBERTa obtaining the highest AUPRC and LSTM the lowest. In terms of AUC, RoBERTa achieved the highest value of 0.7930, while LSTM had the lowest at 0.5931. These results indicate that pre-trained LLMs such as BERT and RoBERTa perform better in multi-label classification compared to LSTM. However, their performance is significantly lower than in binary classification, which is expected since multi-label classification is a more complex task.

Table 4.6: Performance evaluation of various models and hierarchical strategies on the binary EPU classification task

Model	Hierarchical Strategy	Accuracy	F1	AUC	AUPRC	GM
BERT	basic	74.81	74.33	83.86	0.7727	74.54
	masking	74.72	74.30	84.16	0.7724	74.58
	two-stage	74.64	74.33	83.80	0.7641	74.80
	weighted	75.62	74.88	83.93	0.7741	74.90
RoBERTa	basic	76.03	75.48	84.68	0.7895	75.60
	masking	76.19	75.59	84.81	0.7905	75.66
	two-stage	76.15	75.40	84.81	0.7891	75.41
	weighted	75.91	75.22	84.15	0.7801	75.25
LSTM	basic	61.62	58.66	61.93	0.5069	59.08
	masking	62.31	57.93	61.03	0.5023	59.21
	two-stage	61.86	57.95	59.18	0.4644	58.89
	weighted	62.15	56.15	60.56	0.4967	58.88

Hierarchical EPU Classification: Since the task was hierarchical, where articles classified as EPU could be further assigned to 16 multi-label categories, we investigated multiple strategies for hierarchical text classification. Details on these methods are provided in Section 4.1.1.4 of the methodology.

The strategies include a basic approach, where the model predicts both binary and

multi-label classifications simultaneously, a masking approach, where multi-label predictions are made only for articles classified as EPU, a weighted approach, which applies an additional loss penalty for misclassifying articles as not containing EPU, and a two-stage approach, where two models are trained separately, one for binary classification and another for multi-label classification.

Table 4.6 presents the binary EPU classification results for different models across these approaches. Overall, performance differences among the four methods are minimal across all models and evaluation metrics. Among the models, RoBERTa achieved the highest GM of 75.66% using the masking approach. BERT performed best with the two-stage approach, reaching a GM of 74.80%. For LSTM, the highest GM was obtained using the masking approach.

Table 4.7: Performance evaluation of various models and hierarchical strategies on multi-label classification task

Model	Hierarchical Strategy	EMR	AUPRC	AUC	LRP
BERT	basic	59.45	0.0702	0.7080	74.41
	masking	59.45	0.0811	0.7189	74.10
	two-stage	61.94	0.1974	0.7526	81.21
	weighted	59.45	0.0678	0.7046	73.53
RoBERTa	basic	59.45	0.0765	0.7070	74.39
	masking	59.45	0.0963	0.7219	75.22
	two-stage	61.66	0.2231	0.7745	81.02
	weighted	59.45	0.0679	0.6999	73.48
LSTM	basic	59.45	0.0395	0.5879	68.48
	masking	59.45	0.0399	0.5807	67.55
	two-stage	59.25	0.0599	0.5919	67.77
	weighted	59.45	0.0377	0.5620	66.90

Hierarchical Multi-label Text Classification: Table 4.7 presents the EMR, LRP, AUPRC, and AUC metrics for the three models across the four hierarchical approaches. The EMR values remain nearly identical across all approaches. For LRP, the best performance is achieved by BERT using the two-stage approach, followed by RoBERTa again under the same approach with an LRP of 81.02%. LSTM has consistently lower LRP values overall, but its best result is obtained with the basic approach at 68.48%. RoBERTa achieves the highest AUC value of 0.7745 using the two-stage approach.

4.3 Conclusion

In this chapter, we assessed the application of LLMs in the public sector, focusing on economic monitoring by using news articles as indicators to measure EPU levels. This approach was inspired by the research of Baker et al. (2016), which highlighted its potential for developing indices to quantify EPU across different regions and countries. However, their method relied on keyword matching, leading to a high rate of false positives. Despite this limitation, their study provided valuable datasets for both EPU classification and multi-label categorization of news articles identified as discussing EPU, which were essential for evaluating ML models on this task.

Prior research primarily focused on binary EPU classification, where the goal was simply to determine whether an article contained EPU-related content or not. Through our experiments, we extended this by evaluating and developing classifiers for multi-label classification, testing various approaches, and finding that fine-tuned pre-trained LLMs outperformed other methods. Additionally, while earlier studies improved on keyword-based methods by employing either unsupervised ML techniques or supervised ML with human labeled data, this chapter assessed the potential of weakly supervised learning.

Instead of relying on human annotators, we leveraged other LLMs to generate labels. The results showed that extremely large LLMs, such as Mistral-24B, could produce high quality labels in a zero-shot setting using prompting. Furthermore, we observed that aggregation techniques enhance model performance and reduce noise in generated labels, with DP and Dawid-Skene emerging as the most effective methods.

Although classifiers trained on human-labeled data achieved the highest performance, the difference in accuracy between these models and those trained on LLM generated labels was relatively small. However, since human annotation is a time consuming and resource intensive process, using LLM generated data provides a cost-effective alternative for developing EPU classifiers.

Given the hierarchical nature of the classification task, we also tested methods that take advantage of this structure. The two-stage approach, where models are trained separately for binary and multi-label classification, showed the best performance. However, the differences in performance among all the methods studied were minimal.

In the next chapter (chapter 5), the application of LLMs to citizen-generated data, such as feedback and complaints, is presented for tasks including classification, summarization, and question answering.

Chapter 5

Large Language Models for Citizen Feedback Data

With increased digital adoption by citizens and government agencies, it has become more convenient for citizens to share their concerns and feedback, and for government institutions to collect this information efficiently. However, much of the feedback is unstructured, making manual analysis time consuming and ineffective. This chapter evaluates the potential of automated NLP methods, including traditional ML techniques, topic modeling, and LLMs for categorizing, summarizing, and answering questions related to unstructured citizen feedback.

The chapter uses real-world citizen feedback data from URA. URA is a government institution established in 1991, responsible for tax collection and compliance in Uganda [ATJA23]. This chapter is based on the manuscript: *Leveraging Topic Modeling and Large Language Models to Enhance Analysis of Government Citizen Feedback*.

5.1 Methodology

This section describes the proposed methods for analyzing citizen feedback with an overview shown in Figure 5.1.

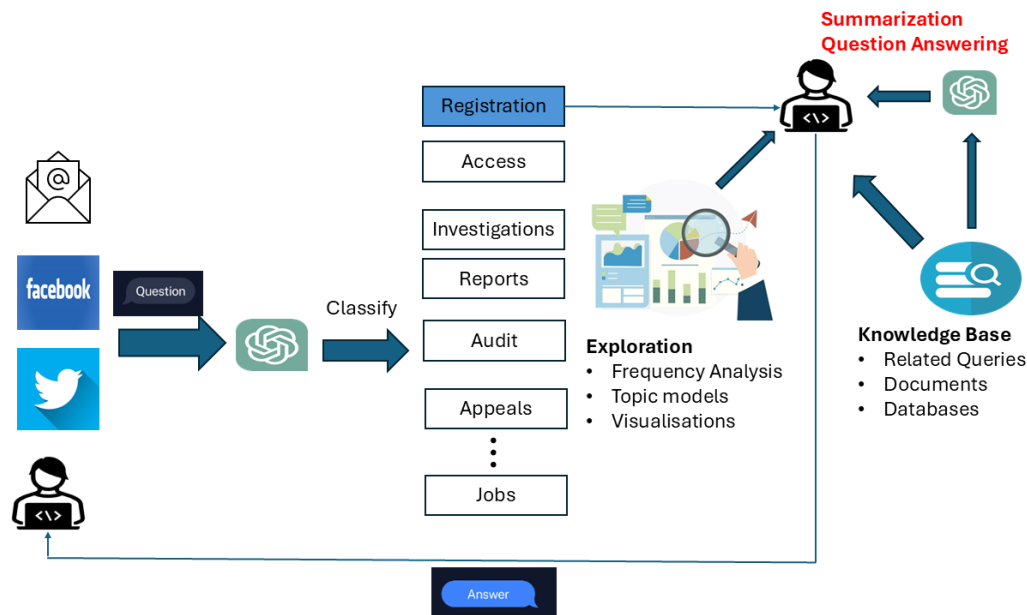


Figure 5.1: Overview of the citizen feedback analysis process, including classification, statistical and descriptive analysis, summarization, and question answering.

5.1.1 Data

The collected dataset includes citizen feedback, inquiries, and complaints gathered from URA through various channels such as calls, emails, intranet, and reports. Feedback received in non-digital formats was processed and stored as text. To ensure privacy, all personally identifiable information or any details that could link individuals to specific data were anonymized or removed by URA.

The dataset comprises 12,558 documents, each annotated by URA staff with relevant categories, summaries, and responses to the submitted inquiries. The distribution of categories is shown in Table 5.1 respectively.

The distribution of categories shows that most frequent inquiries involve registration, access control, and motor vehicle registration.

Table 5.1: Class distribution of categories. Categories with less 40 observations were combined into “Rare categories”

Categories	Number of tickets
Registration	3480
Access control	2749
Unclassified	1500
Motor vehicle registration	1051
Assessments	977
Returns	918
Other portal functionalities	367
Payments	355
Taxpayer accounting	341
Data change	224
Objection and appeals	208
Rare categories	122
NTR	73
DMS	52
Debt collection	51
Stamp duty	48
Audit	42
Grand total	12558

5.1.2 Topic Modeling for User Feedback

To understand statistical patterns and visual mappings in the collected textual feedback, count-based methods: n-grams, word distributions, frequency plots, and word clouds were first used. These methods provided valuable insights but were limited in capturing deeper relationships within the text. Therefore, we extended our analysis by using topic modeling approaches, specifically LDA [BNJ03], BERTopic [Gro22], and STM [RST19].

LDA is a probabilistic generative model that identifies hidden topics within a collection of documents. It assumes that each document is a mixture of multiple topics, and each topic is characterized by a probability distribution over words. In the LDA framework, each topic u is associated with a distribution over the vocabulary, denoted as ϕ_u , which is drawn from a Dirichlet distribution with parameters β as shown in Equation 5.1.

$$\phi_u \sim \text{Dirichlet}(\beta) \quad (5.1)$$

Similarly, each document x_i is assumed to follow a topic distribution, denoted as θ_{x_i} , which is drawn from a Dirichlet distribution with parameters α , as shown in Equation 5.2.

$$\theta_{x_i} \sim \text{Dirichlet}(\alpha) \quad (5.2)$$

For each word in a document, LDA follows a two-step process. First, a topic assignment $z_{x_i,j}$ is sampled from a multinomial distribution parameterized by θ_{x_i} as shown in Equation 5.3.

$$z_{x_i,j} \sim \text{Multinomial}(\theta_{x_i}) \quad (5.3)$$

Once the topic is chosen, the actual word $w_{x_i,j}$ is sampled from the multinomial distribution associated with the selected topic $\phi_{z_{x_i,j}}$ shown in Equation 5.4.

$$w_{x_i,j} \sim \text{Multinomial}(\phi_{z_{x_i,j}}) \quad (5.4)$$

The parameters α and β are hyperparameters that control the distribution of topics in documents and words in topics respectively. To estimate the latent variables and infer the topic structure from the data, techniques such as variational inference and Gibbs sampling are commonly employed [BNJ03].

BERTopic enhances traditional topic models by utilizing contextualized embeddings that capture more meaningful semantic relationships between words. It begins by converting each document into a dense vector representation using models like sentence-BERT [RG19]. These vectors are then reduced in dimensionality through methods like Uniform Manifold Approximation and Projection, and clustered using Hierarchical Density-Based Spatial Clustering of Applications with Noise. Each cluster corresponds to a unique topic. To identify the key terms for each topic, BERTopic applies a modified version of TF-IDF called c-TF-IDF.

In BERTopic, the TF-IDF formula is modified to work with clusters by combining all documents within a cluster into one large document. This approach allows the TF-IDF calculation to reflect the importance of terms at the cluster level rather than individual documents. The transformation from documents to clusters is illustrated in Equation 5.5.

$$W_{t,b} = TF_{t,b} \cdot \log \left(1 + \frac{A}{TF_t} \right) \quad (5.5)$$

Here, $W_{t,b}$ is the weight of term t in cluster b , and $TF_{t,b}$ is the term frequency of term

t in that cluster. The parameter A represents the average number of words per cluster, while TF_t is the total frequency of term t across all clusters. This method highlights terms that are more frequent within a specific cluster and less common across others, resulting in clearer and more interpretable topic representations.

STM allows the inclusion of metadata into the prior distributions for document-topic and topic-word proportions. For our case, metadata includes the category of feedback, the date it was received, whether it was resolved upon initiation, and the receiving channel, among other factors. STM starts by assigning each document a distribution of topics, where this distribution is influenced by the document's metadata. This relationship is modeled using a logistic-normal generalized linear model based on a vector of document covariates X_{x_i} shown in Equation 5.6.

$$\theta_{x_i} | X_{x_i} \gamma, \Sigma \sim \text{LogisticNormal}(\mu = X_{x_i} \gamma, \Sigma) \quad (5.6)$$

where γ is a matrix of coefficients and Σ is the covariance matrix, and μ is the mean of the multivariate normal distribution.

After determining the topical prevalence, STM then assigns words to topics based on these document-level topic distributions (Equations 5.7 and 5.8).

$$z_{x_i,q} | \theta_{x_i} \sim \text{Multinomial}(\theta_{x_i}) \quad (5.7)$$

In this context, $z_{x_i,q}$ represents the topic assigned to word q in document x_i , which is drawn from the topic distribution θ_{x_i} . The next step involves assigning words to topics given the topic assignment, using a word distribution specific to each topic.

$$w_{x_i,q} | z_{x_i,q}, \beta_{x_i,u=z_{x_i,q}} \sim \text{Multinomial}(\beta_{x_i,u=z_{x_i,q}}) \quad (5.8)$$

In Equation 5.8, $w_{x_i,q}$ represents the observed word, while $\beta_{x_i,u}$ refers to the word distribution for topic u . These steps illustrate how STM integrates document metadata into the topic modeling process, accounting for the effects of covariates on both the topic distributions and the specific words within each topic.

5.1.3 Machine Learning for User Generated Feedback

For ML experiments, we consider a training dataset consisting of n labeled examples, represented as: $D = \{(x_i, y_i)\}_{i=1}^n$, where each instance comprises an input-output pair. In the context of this chapter, x_i denotes the collected textual feedback, while y_i repre-

sents the corresponding target output which could be: a category, summary, or response to x_i .

The objective is to develop a model capable of accurately predicting the target output y_{test} for a new test input x_{test} coming from the same distribution as the training dataset. This section discusses the methodologies used to train models for classification, summarization, and question answering using the collected feedback data.

5.1.3.1 Classification

For the classification experiments, the main objective was to assign each piece of textual feedback to the most probable category. The distribution of categories was presented in Table 5.1. During classification, categories with fewer than 40 examples were merged into “Rare categories” category to ensure sufficient representation across training and testing splits.

Traditional ML techniques were trained from scratch, with their methodology described in section 2.3, while LLMs were fine-tuned starting from pre-trained weights. The collected dataset had an uneven distribution of categories, necessitating strategies to mitigate class imbalance during training and fine-tuning.

To address class imbalance in this chapter, we used undersampling, oversampling, and cost-sensitive learning. Undersampling reduces the number of samples in the majority class to match the minority class, while oversampling increases the number of samples in underrepresented classes to create a more balanced dataset [JK19]. For cost-sensitive learning, we used weighted CE loss [ZWM19], which assigns different weights to each class based on its frequency.

Fine-tuning for category classification: Fine-tuning was conducted on pre-trained LLMs using full 16-bit precision. The models were initialized with pre-trained weights θ_0 and further trained on the collected dataset using supervised learning. The objective of fine-tuning was to optimize the model by minimizing the weighted CE loss, updating the pre-trained parameters θ_0 to task-specific parameters θ . The corresponding loss function is provided in Equation 5.9.

$$\mathcal{L}_{\text{weighted CE}}(y, \hat{y}) = -\frac{1}{n} \sum_{i=1}^n \sum_{c=1}^{n_c} w_c \cdot y_{i,c} \cdot \log(\hat{y}_{i,c}) \quad (5.9)$$

Here, n is the total number of training examples, n_c is the number of possible categories, $y_{i,c}$ is a binary indicator that takes the value 1 if class c is the correct category for input x_i and 0 otherwise, and $\hat{y}_{i,c}$ is the predicted probability for class c , obtained via softmax over model logits. The class weight w_c is determined based on the frequency f_c of class

c in the dataset defined in Equation 5.10.

$$w_c = \frac{n}{n_c \cdot f_c} \quad (5.10)$$

Fine-tuning iteratively updates θ_0 through backpropagation and gradient-based optimization techniques, allowing the model to retain existing knowledge while adapting to the category classification task.

Prompting for category classification: For category classification, a natural language prompt was created to define the classification task and outline the relevant categories for a given text. In the zero-shot prompting approach, no demonstration examples were provided, as shown in the template in Figure 5.2.

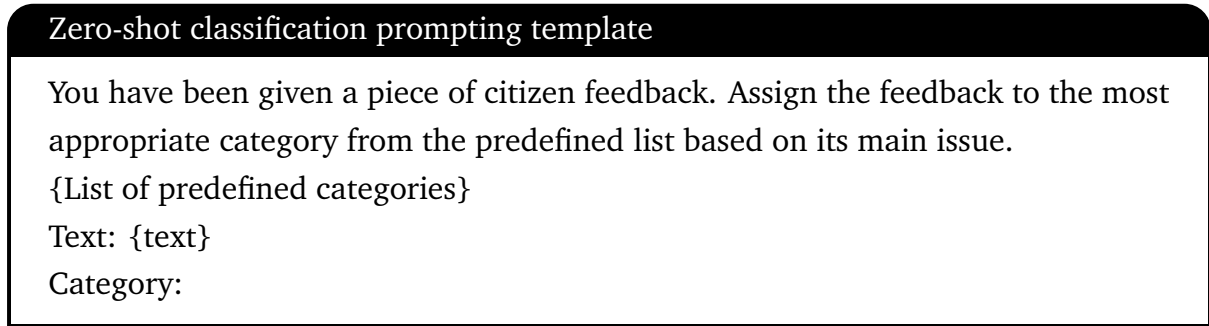


Figure 5.2: Template for classification with zero-shot prompting. The variables in brackets are replaced with actual values at inference

In the few-shot prompting approach, demonstration examples were incorporated into the prompt, as illustrated in Figure 5.3. These examples were selected based on their semantic similarity to the input text. Experiments were conducted with different numbers of demonstration examples, ranging from 1 to 50. Despite the inclusion of these examples, there were no updates to the model parameters.

Few-shot classification prompting template

You have been given a piece of citizen feedback. Your task is to categorize the feedback into one of the predefined categories given below.

{List of predefined categories}

To help you understand the classification criteria, you will first see a few demonstration examples illustrating how the feedback should be categorized.

Below are the demonstration examples that serve as a reference for assigning categories:

{Demonstration examples with text and expected category}

Now, using these examples as guidance, classify the following feedback into the most appropriate single category.

Text: {text}

Category:

Figure 5.3: Template for classification with few-shot prompting. The variables in brackets are replaced with actual values at inference

The few-shot prompt template including a set of demonstration examples is represented as: $I_c = \{I_P, s(x_1, y_1), \dots, s(x_r, y_r)\}$, where $s(x_r, y_r)$ denotes the r -th demonstration example pair and I_P represents a natural language description of the task. The objective was to guide the LLM in selecting the most probable category by determining the option with the highest probability score based on the provided instructions I_c defined in Equation 5.11.

$$\hat{y}_i = \arg \max_{y_i} P_\theta(y_i \mid I_c, x_i) \quad (5.11)$$

5.1.3.2 Summarizing and Question Answering

This section describes the methodology used to evaluate the effectiveness of LLMs in automating the summarization and question answering of citizen feedback. LLMs were applied to both tasks using prompting and fine-tuning techniques. The instruction templates for these tasks are shown in Figures 5.4 and 5.5. In these templates, the expected summary or response was not included during prompting. The inference process follows a similar approach to the one used in the classification task described in section 5.1.3.1, with the key distinction being that instead of assigning categories, the model generates summaries or relevant answers to the input text.

For fine-tuning, supervised instruction tuning was used to adapt generative LLMs for summarization and question answering tasks. Instruction tuning is a form of fine-tuning that involves continued pre-training on a dataset formatted as instruction-response

pairs while maintaining the same training objective used during pre-training, specifically NTP. Although this objective remains self-supervised, it differs from the original NTP because the LLM is additionally constrained to generate correct response to the given instruction as it generates the next token [ZDL⁺23].

The initial step in instruction tuning involved structuring the dataset into instruction-response pairs using predefined templates. In this chapter, similar templates to those used in prompting were applied, with distinction that the expected output is included in the prompt for this case. As illustrated in Figure 5.4 for the summarization task, both the input text and its corresponding summary were provided to the LLM.

Summarization Template
You are provided with text containing textual citizen feedback Summarize the text capturing the main points Text: {text} Summary: {summary}

Figure 5.4: A prompt template for instruction tuning language models in summarization tasks, where text in {brackets} is replaced with actual feedback, and summaries are substituted with expert-curated summaries

The same approach used for the summarization task was applied to the question-answering dataset. The dataset was structured into question-answer pairs following the template shown in Figure 5.5.

Question Answering Template
Answer the question based on the provided information. Question: {question} Answer: {answer}

Figure 5.5: A prompt template for instruction tuning language models in question-answering tasks, where {question} is replaced with user queries and {answer} with model-generated or expert-provided responses.

The final processed dataset comprises instruction-response pairs, where each pair I_i is a single concatenated sequence that includes the instruction, input text, and the corresponding expected response. The supervised instruction tuning loss function $\mathcal{L}_{SIT}$ is defined in Equation 5.12 [ZDL⁺23].

$$\mathcal{L}_{SIT}(I_i, R_i) = -\frac{1}{n} \sum_{i=1}^n \sum_{t=1}^{T_i} \log P_{\theta}(R_{i,t} \mid I_i, R_{i,<t}) \quad (5.12)$$

where θ denotes the parameters of the language model, T_i is the length of the response sequence, $R_{i,t}$ is the token at position t in the response, and $R_{i,<t}$ represents the sequence of tokens preceding position t .

The models selected for this task were within the 7 billion parameter range; however, the available GPU hardware was not sufficient to support fine tuning LLMs at the standard 16 bit precision. To overcome this constraint, we employed a parameter efficient fine-tuning technique known as QLoRA [DPHZ23], which combines low rank adaptation, approximating weight matrices using lower rank representations to reduce the number of trainable parameters, with 4 bit precision quantization, enabling compression of model weight updates instead of using the typical 16 bit format. This method strikes a balance between computational efficiency and model performance, reducing GPU memory requirements by over four times while incurring only minimal accuracy loss.

5.1.3.3 Training and Implementation Details

For experiments, we implemented various LLMs, selecting a mix of earlier pre-trained models, generative and discriminative LLMs, as well as modern versions of top performing models such as LLAMA and Qwen families. The decision to use smaller versions of LLAMA and Qwen was driven by the need to fine-tune all models in 16-bit precision for a fair comparison while being constrained by hardware limitations that prevented fine-tuning models larger than 3 billion parameters in 16-bit precision.

The models used for classification include: BERT, RoBERTa, DistilBERT, DistilRoBERTa, OPT-125M, OPT-350M, Qwen2-3B, Qwen2-0.5B, LLAMA3-1B, and LLAMA3-3B. Additionally, traditional ML models, including Decision Trees, Random Forests, GBM, and SVM, were implemented using the Scikit-Learn library [PVG⁺11].

For zero-shot and few-shot prompting category classification, the Qwen2-7B model was chosen for its strong performance among the 7 billion parameter models available at the time of this research. Its large context window of 1 million tokens made it particularly suitable for testing with a larger number of few-shot examples, which would not have been feasible with models that had smaller context limits.

For summarization and question answering we used 3 models: Zephyr-7B, LLAMA2-7B, Mistral2-7B. These models were selected because they were SoTA among the 7 billion parameter models at the time of experiments. These models were fine-tuned

with QLoRA in 4-bit precision [DPHZ23].

For implementation, the fine-tuning process was primarily conducted using the Hugging Face Transformers library, while prompting and few-shot prompting was carried out using a combination of Langchain and Transformers. Experiments were conducted on a single A100 40GB GPU machine hardware. The code, datasets, and hyperparameters referenced in this chapter are available in the GitHub repository: https://github.com/TrustPaul/citizen_feedback_analysis.

5.1.4 Evaluation Setup and Metrics

The performance of various models was evaluated using F1, accuracy, AUC, AUPRC and GM detailed in section 4.1.3. To evaluate summarization and question answering tasks, we utilized Recall Oriented Understudy for Gisting Evaluation (ROUGE) which is specifically designed to evaluate automatic summarization and machine translation. The ROUGE metric compares an automatically generated summary or translation with one or more human-produced reference summaries or answers. It includes variants such as ROUGE-1, ROUGE-2, and ROUGE-L, which measure unigram, bigram, and longest common subsequence matches respectively with the reference text [Lin04].

Due to the abstractive nature of our summarization and question answering tasks, the generated summary or answer may not always contain the same n-grams as the reference summary, even if it conveys a similar meaning. BERTScore was used which leverages pre-trained contextual embeddings from BERT and calculates the cosine similarity between words in the candidate and reference summaries or answers [ZKW⁺20].

5.2 Results and Discussion

This section presents the results and discussion of the proposed approaches applied to citizen feedback data from URA.

5.2.1 Statistical and Descriptive Analysis

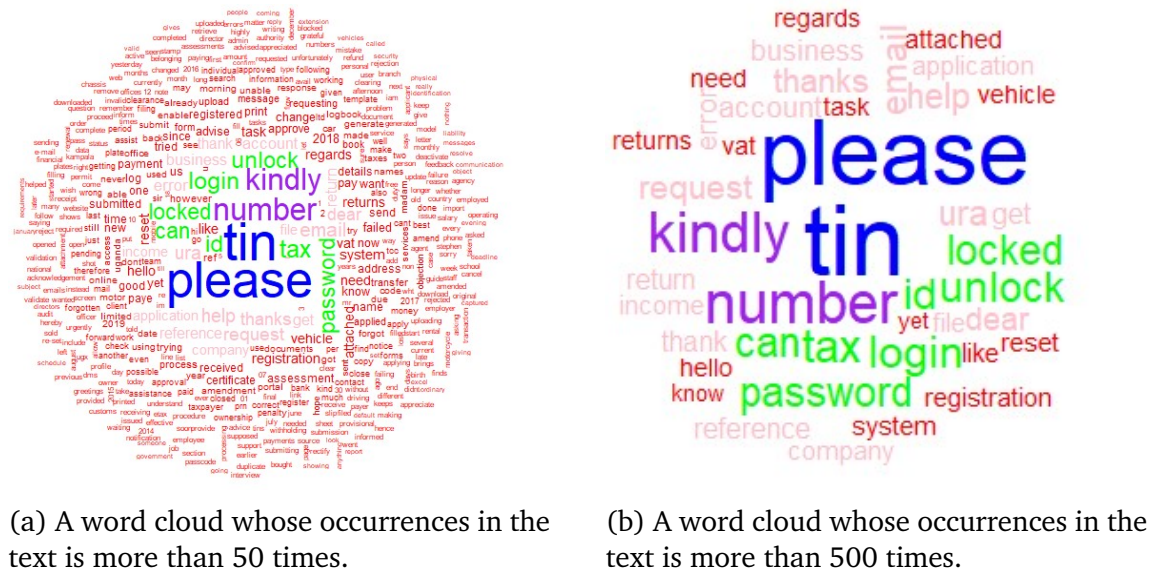


Figure 5.6: Word plots that summarize key words in the textual feedback

We utilized frequency based methods for the descriptive and statistical analysis of textual feedback to uncover common patterns and topics. Our first discussion involves word clouds that highlights the most prevalent terms in the dataset. Figure 5.6b is a word cloud for terms that appear more than 500 times, while Figure 5.6a presents a word cloud for terms occurring over 50 times. The word size in each cloud is proportional to their frequency in the dataset.

The frequent appearance of terms such as “locked”, “password”, and “unlock” reflects common client concerns like system access issues. Similarly, terms like “tax”, “number”, and “tin” likely pertain to inquiries about processing and acquiring Tax Identification Number (TIN).

5.2.2 Topic modeling results

To analyze word distributions beyond basic word frequencies, topic models were applied for a detailed statistical analysis, details about the methodology are provided in section 5.1.2.

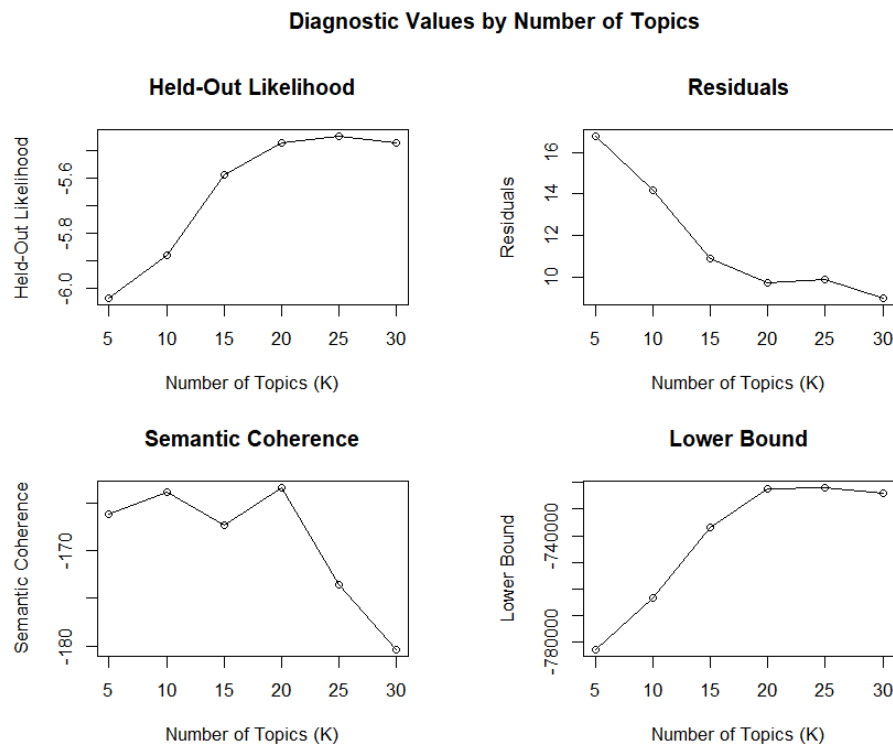


Figure 5.7: Diagnostic values to determine the optimal number of topics

Determining the optimal number of topics: Before applying topic models, a key hyperparameter that must be determined is the number of topics. Figure 5.7 presents several diagnostic metrics such as: held-out likelihood, residuals, semantic coherence, and lower bounds to estimate the optimal number of topics for our data. The held-out likelihood increases with the number of topics and levels off around 25, while residuals decrease as the number of topics grows from 5, stabilizing around 15, suggesting diminishing improvements in model accuracy beyond this point.

Semantic coherence peaks at 10 topics and declines afterward, indicating that topics become less coherent and meaningful beyond this point. Although the lower bound continues to rise, relying on it solely may lead to less interpretable topics. After reviewing all the metrics, the optimal number of topics is more likely to be between 10 and 15, balancing semantic coherence and model fit. We chose to fit a 13-topic model.

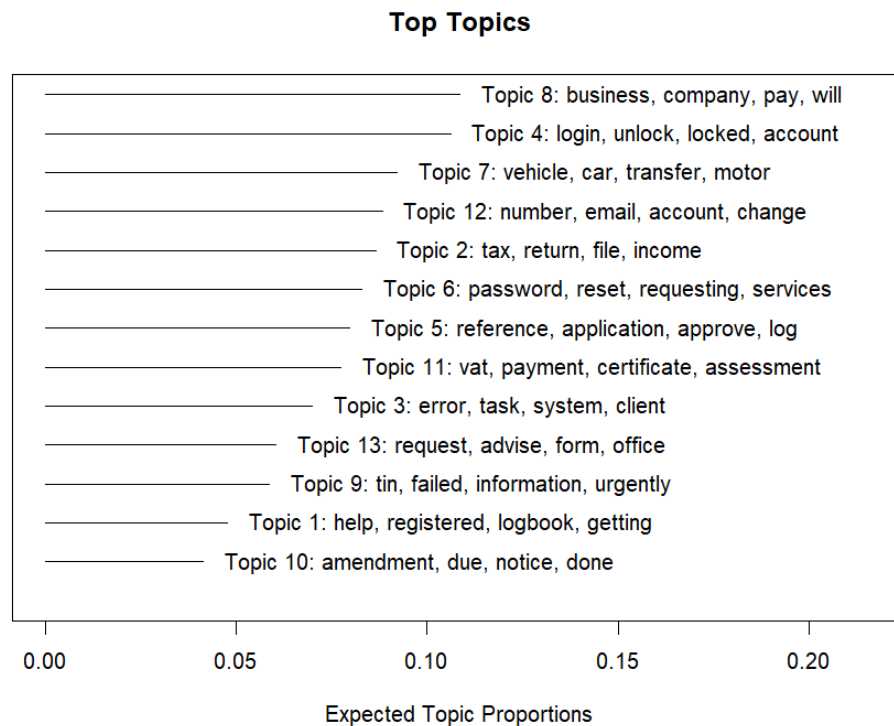
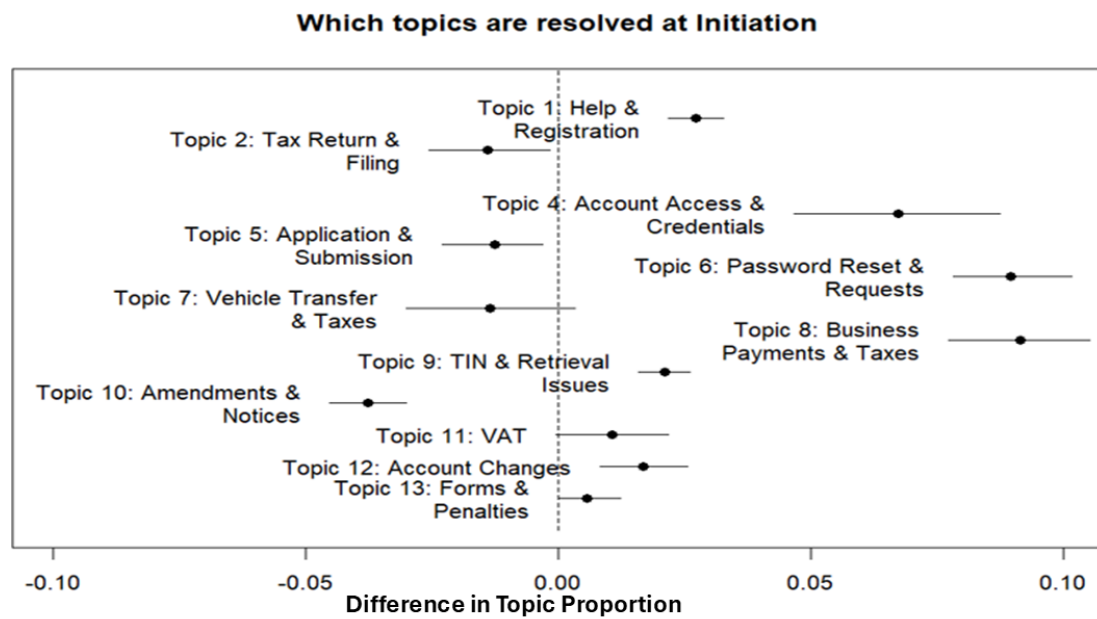


Figure 5.8: Distribution of words within topics

Topic distributions: Figure 5.8 presents the top 13 topics identified using STM with metadata included. In the discussion, we use one or two words to label topics for easy followup during the discussion. The most prevalent topic is Topic 8: Business Payments & Taxes which relates to business payments and tax issues, with key terms such as “business”, “company”, and “pay”. The second most prevalent is Topic 4: Account Access & Credentials which focuses on account access and credential issues, with top words like “login”, “unlock”, and “account”. Topic 7: Vehicle Transfers & Taxes pertains to vehicle transfers and associated taxes, while Topic 12: Account Changes revolves around account changes highlighted by terms such as “email”, “account”, and “change”.

Topic 2: Tax Return & Filing is primarily concerned with tax returns and income filings, and Topic 6: Password Reset & Requests covers password resets and service requests. The remaining topics cover various issues, including references and approvals (Topic 5: Application & Submission), Value Added Tax (VAT), payment certificates (Topic 11: VAT), system errors and client tasks (Topic 3: System Errors & Tasks). The least discussed topics are Topic 1, which addresses registration, help and logbook issues, and Topic 10: Amendments & Notices, which deals with amendments and due notices.



More Not Resolved More Resolved

Figure 5.9: Topics associated with inquiries that are resolved at initiation versus those that are not.

Topics that are more likely to be resolved at initiation: In the next analysis, we studied how metadata affects topic distributions and examine the differences across treatment groups within the metadata. Figure 5.9 shows the differences between topics more likely to be resolved at the initiation stage and those requiring further action. The horizontal error bars in the graph represent difference in topic proportions for each effect estimate, with the dashed vertical line at zero as a reference point for no effect. Topics such as “Topic 1: Help & Registration”, “Topic 6: Password Reset & Requests”, “Topic 8: Business Payments & Taxes”, and “Topic 12: Account Access & Credentials” show positive effects, indicating a higher likelihood of resolution at the initiation stage.

These topics are related to account access, password resets, and help requests, which can often be resolved easily. On the other hand, topics such as “Topic 10: Amendments & Notices”, “Topic 9: TIN & Retrieval Issues”, and “Topic 13: Forms & Penalties” have estimates near zero or negative. This suggests that these inquiries are more likely to require additional authorization, further information from the individual, or may experience delays before being resolved.

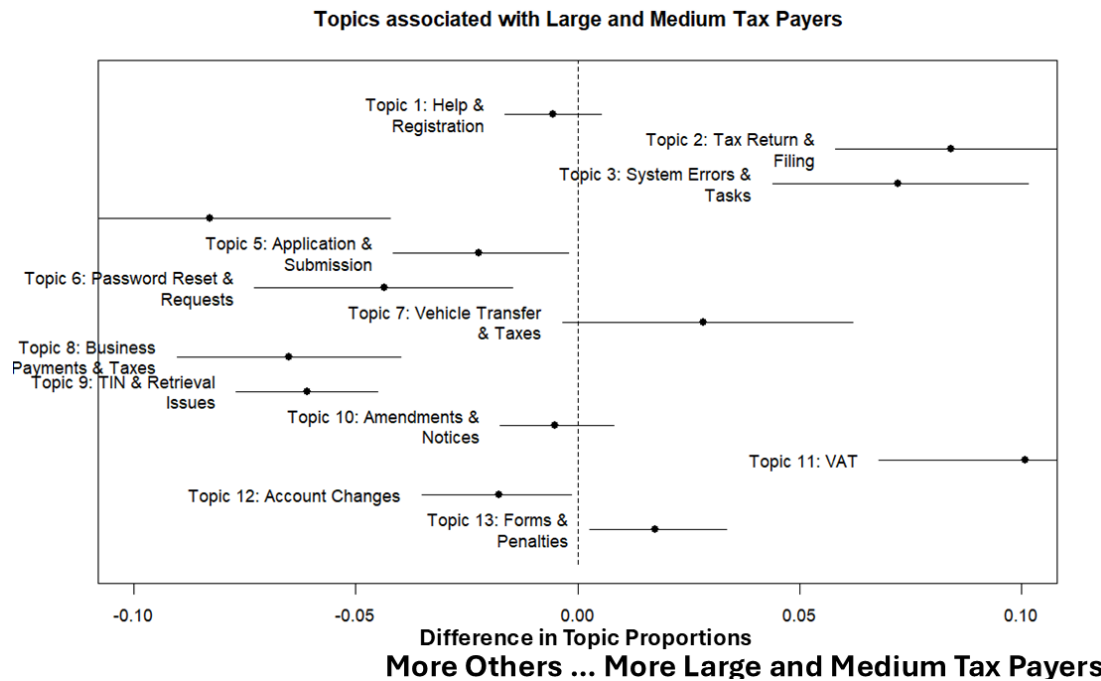


Figure 5.10: Topics received from Large and Medium tax payers versus the rest of the tax payers

Topics that are more associated with large and medium tax payers: Figure 5.10 highlights the differences in topics associated with large and medium taxpayers compared to other taxpayers. Topics to the right of the zero line appear more frequently in inquiries from large and medium taxpayers, while those on the left are more commonly linked to other taxpayers. Topics such as “Topic 11: VAT” and “Topic 7: Vehicle Transfer & Taxes” are more often associated with large and medium taxpayers, who typically raise tax-related concerns rather than issues like password resets.

In contrast, topics like “Topic 8: Business Payments & Taxes”, “Topic 9: TIN & Retrieval Issues”, and “Topic 12: Account Changes” tend to be more associated with other taxpayers. Some topics such as “Topic 1: Help & Registration” and “Topic 2: Tax Return & Filing”, are equally likely to come from either large and medium taxpayers or other taxpayers.

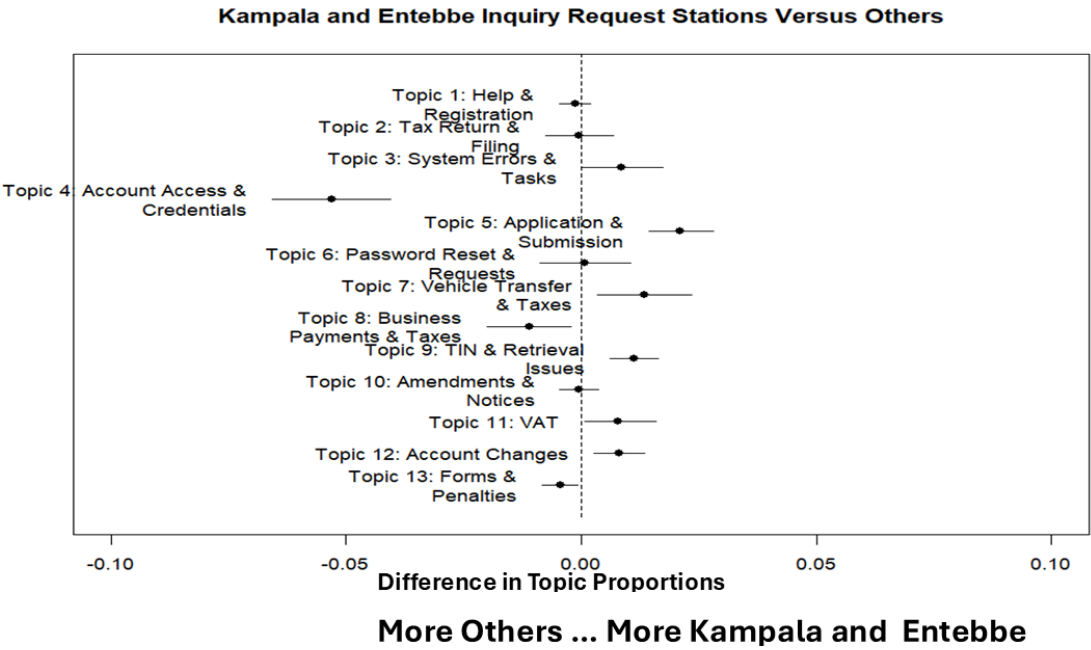


Figure 5.11: Topics handled by Kampala and Entebbe Stations versus the rest of the stations

Topics that are more likely to come from capital city locations (Kampala and Entebbe): Figure 5.11 illustrates the differences in citizen feedback topics received from the Kampala and Entebbe stations (capital city stations) compared to other locations. Topics positioned to the right of the zero line are more likely to have originated from Kampala and Entebbe, while those on the left are more associated with other stations. Most topics cluster near the center, indicating that the issues received at Kampala and Entebbe stations are generally similar to those handled at other stations.

While close to the center, topics like “Topic 1: Help & Registration” and “Topic 5: Application & Submission” are slightly more likely to be received from Kampala and Entebbe. In contrast, “Topic 4: Account Access & Credentials” and “Topic 8: Business Payments & Taxes” appear more frequently at other stations. Topics such as “Topic 3: System Errors & Tasks” and “Topic 13: Forms & Penalties” show minimal differences, suggesting these issues are equally likely to come from any station.

5.2.3 Classification of citizen feedback

This section discusses the results of the proposed ML techniques for automatically classifying feedback into predefined categories, with their class distributions presented in Table 5.1.

5.2.3.1 Citizen Feedback Classification

In Table 5.2, we present the average performance of different models across various class imbalance handling strategies for category prediction. This task involves a larger number of classes, with 17 categories, as detailed in Table 5.1. The categories also have an imbalanced distribution, we implemented multiple strategies to address this issue, including undersampling, oversampling, and applying weighted CE loss functions during training and fine-tuning.

Table 5.2: Evaluation metrics: F1, GM, AUC, AUPRC of different Models for category prediction. FS in this table means Few-shot

Model	Accuracy	F1	GM	AUC	AUPRC
Decision Trees	65.61 $\pm$ 7.69	53.52 $\pm$ 5.09	73.36 $\pm$ 0.69	0.5049 $\pm$ 0.0013	0.1199 $\pm$ 0.0062
Random Forests	78.04 $\pm$ 4.12	66.21 $\pm$ 3.62	79.95 $\pm$ 4.57	0.5854 $\pm$ 0.0117	0.1890 $\pm$ 0.0089
GBM	76.83 $\pm$ 5.14	67.67 $\pm$ 3.58	82.03 $\pm$ 2.50	0.5427 $\pm$ 0.0187	0.1686 $\pm$ 0.0106
SVM	79.26 $\pm$ 5.79	70.37 $\pm$ 4.70	82.56 $\pm$ 2.21	0.5721 $\pm$ 0.0156	0.1928 $\pm$ 0.0113
OPT-125M	82.86 $\pm$ 2.71	73.79 $\pm$ 2.47	84.73 $\pm$ 0.98	0.9620 $\pm$ 0.0067	0.7879 $\pm$ 0.0161
DistilBERT	83.35 $\pm$ 2.01	74.08 $\pm$ 1.51	84.87 $\pm$ 0.77	0.9634 $\pm$ 0.0057	0.7885 $\pm$ 0.0112
DistilRoBERTa	83.04 $\pm$ 2.31	74.23 $\pm$ 2.34	85.13 $\pm$ 0.65	0.9593 $\pm$ 0.0071	0.7795 $\pm$ 0.0082
RoBERTa	83.20 $\pm$ 2.17	74.23 $\pm$ 1.45	85.44 $\pm$ 1.19	0.9593 $\pm$ 0.0096	0.7924 $\pm$ 0.0088
OPT-350M	83.42 $\pm$ 2.54	74.46 $\pm$ 1.32	85.24 $\pm$ 1.16	0.9577 $\pm$ 0.0077	0.7892 $\pm$ 0.0193
BERT	83.34 $\pm$ 2.48	74.71 $\pm$ 2.02	85.84 $\pm$ 1.19	0.9606 $\pm$ 0.0067	0.7978 $\pm$ 0.0114
Qwen2-7B-Prompt	30.50 $\pm$ 0.38	33.77 $\pm$ 0.72	33.95 $\pm$ 1.02	—	—
Qwen2-7B-FS	44.62 $\pm$ 0.38	61.64 $\pm$ 0.72	57.53 $\pm$ 1.02	—	—
LLAMA3-1B	85.85 $\pm$ 0.38	76.10 $\pm$ 0.72	84.85 $\pm$ 1.02	0.9746 $\pm$ 0.0005	0.8280 $\pm$ 0.0067
Qwen2-0.5B	84.72 $\pm$ 0.37	75.68 $\pm$ 0.98	85.56 $\pm$ 0.62	0.9709 $\pm$ 0.0018	0.8240 $\pm$ 0.0065
Qwen2-3B	87.30 $\pm$ 0.27	78.92 $\pm$ 0.67	88.13 $\pm$ 0.63	0.9564 $\pm$ 0.0067	0.8057 $\pm$ 0.0148
LLAMA3-3B	87.68 $\pm$ 0.49	79.81 $\pm$ 1.58	88.51 $\pm$ 0.62	0.9541 $\pm$ 0.0082	0.8084 $\pm$ 0.0284

In terms of F1 score, the fine-tuned LLAMA3-3B achieved the highest score at 79.81%, followed by LLAMA3-1B with 76.10%. Among earlier pre-trained LLMs, OPT-125M performed best with an F1 score of 73.79%, followed by OPT-350M (74.46%) and RoBERTa (74.23%). Among traditional ML models, SVM achieved the highest F1 score at 70.37%, followed by GBM (67.67%), while Decision Trees performed the worst with 53.52%. The lowest overall performance was observed in the zero-shot version of Qwen2-7B, which achieved an F1 score of 33.77%, followed by its few-shot version with 61.64%.

For GM, LLAMA3-3B achieved the highest value at 85.51%, closely followed by LLAMA3-1B with 85.40%. Among earlier pre-trained LLMs, RoBERTa performed well with a GM

of 84.15%, followed by BERT (84.11%) and DistilRoBERTa (84.87%). Among traditional ML models, SVM had the highest GM at 82.56%, followed by GBM (82.03%), while Decision Trees had the lowest GM within this category at 73.36%. The zero-shot version of Qwen2-7B again showed the weakest performance, with a GM of just 33.95%.

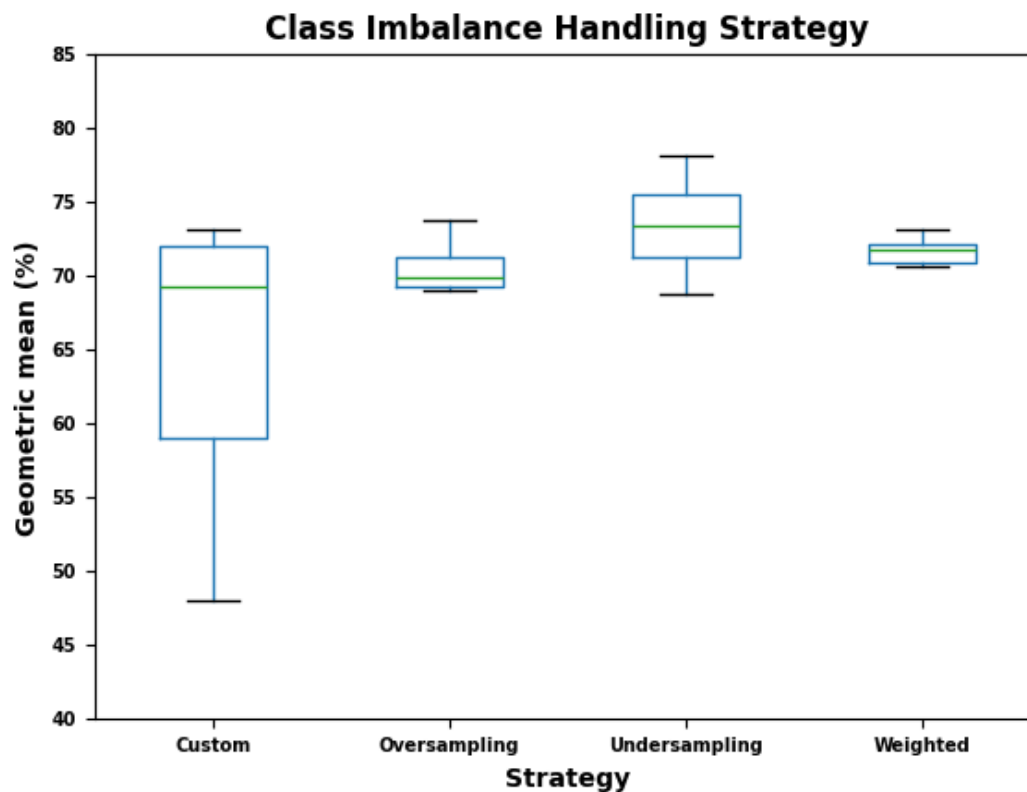


Figure 5.12: Box plots showing different strategies to handle class imbalance for category classification

Figure 5.12 is a box plot showing the results of various class imbalance handling strategies, measured using GM, during model training or fine-tuning. In this graph, "Custom" represents models trained with standard CE loss function without specific class imbalance adjustments, while the "Weighted" approach refers to the use of a weighted CE loss.

Undersampling achieved both the highest maximum and median GM, while the weighted and oversampling approaches had almost the same median values. The custom training method showed the lowest median. Overall, applying strategies to address class imbalance led to improved performance on average.

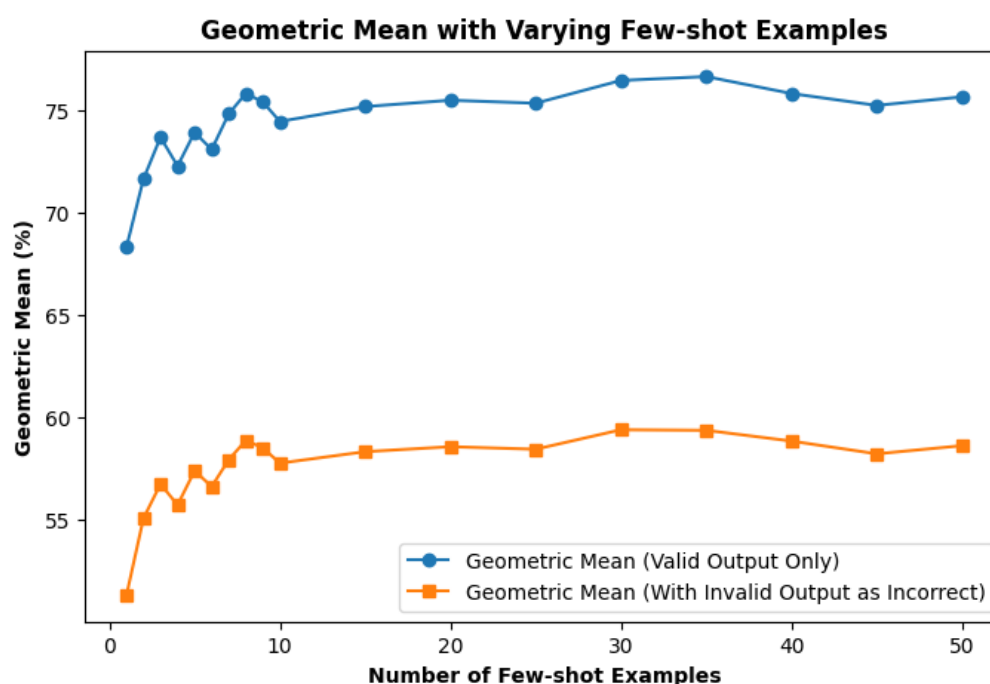


Figure 5.13: Line graphs showing how number of demonstration examples varies with GM

For the few-shot category classification task, we used the Qwen2-7B model, one of the top-performing models in the 7-billion parameter family, with few-shot examples ranging from 1 to 50. Figure 5.13 shows the GM under two conditions: one considering only valid outputs and another treating invalid outputs as incorrect. It is essential to distinguish between incorrect and invalid outputs; incorrect outputs are valid responses that are factually wrong, while invalid outputs do not follow the expected format or structure.

The first observation is that these graphs differ significantly, indicating that the model produces a large number of invalid outputs. This is expected, as category classification involves a much higher number of classes compared to category classification, making it more challenging for the model to generate valid responses.

In this task, increasing the number of demonstration examples leads to a noticeable improvement in performance, particularly between 1 and 10 examples, where some fluctuations are observed but the overall trend remains positive. However, after reaching 20 examples, the performance levels off in both graphs, indicating diminishing returns. This pattern suggests that while few-shot learning provides substantial gains initially, adding more examples beyond a certain point does not yield further improvement.

5.2.3.2 Citizen Feedback Summarization

In addition to classifying the collected feedback, summarizing it is a crucial task. Feedback can often be lengthy and unclear, making it challenging for the recipient to process efficiently. Providing a precise summary helps streamline the response process, allowing quicker and more effective communication.

For this task, we utilized the LLMs: Zephyr-7B, Mistral2-7B, and LLAMA2-7B. Since these are generative models, the focus was on abstractive summarization, where the summary is generated in a way that captures the essence of the content without directly extracting specific sections from the original text.

Table 5.3: ROUGE and BERTScore metrics for prompting and fine-tuning approaches across models for summarization.

Adaptation	Model	ROUGE Scores				BERTScore
		ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-Lsum	
Prompting	Zephyr-7B	0.1195	0.0355	0.1115	0.1117	0.7087
Prompting	LLAMA2-7B	0.1433	0.0501	0.1327	0.1334	0.7369
Prompting	Mistral2-7B	0.1705	0.0617	0.1576	0.1577	0.7482
Fine-tuning	Zephyr-7B	0.1866	0.0703	0.1777	0.1780	0.7270
Fine-tuning	LLAMA2-7B	0.1515	0.0562	0.1408	0.1415	0.7398
Fine-tuning	Mistral2-7B	0.3055	0.1248	0.2856	0.2856	0.7771

Table 5.3 presents the summarization performance of the evaluated models using ROUGE and BERTScore metrics. The fine-tuned models consistently outperformed their prompt-based counterparts. On average, fine-tuning achieved a ROUGE-1 score of 0.2145 compared to 0.1444 from prompting, and a ROUGE-L score of 0.2014 versus 0.1340 from prompting.

The best performing model was the fine-tuned Mistral2-7B, achieving a ROUGE-1 score of 0.3055, ROUGE-L of 0.2856, and a BERTScore 0.7771. For BERTScore, this was followed by LLAMA2-7B 0.7398 and Zephyr-7B 0.7270. In terms of ROUGE-Lsum scores, Zephyr-7B ranked second with 0.1780, followed by LLAMA2-7B with 0.1415.

Among the prompt-based models, Mistral2-7B performed the best, obtaining a ROUGE-1 score of 0.1705, ROUGE-L of 0.1576, and a BERTScore of 0.7482. In the BERTScore evaluation, LLAMA2-7B was the second best performing model with 0.7369, followed by Zephyr-7B with 0.7087. These results indicate that fine-tuning significantly enhances summarization performance compared to using prompting alone.

Table 5.4 presents the results for the question answering task. Among the fine-tuned models, Mistral2-7B demonstrated the highest overall performance, achieving a ROUGE-1 score of 0.2142, ROUGE-2 of 0.0608, ROUGE-L of 0.1549, and ROUGE-Lsum

Table 5.4: ROUGE and BERTScore metrics for prompting and fine-tuning approaches across models for Question Answering

Adaptation	Model	ROUGE Scores				BERTScore
		ROUGE -1	ROUGE -2	ROUGE -L	ROUGE-Lsum	
Prompting	Zephyr-7B	0.1904	0.0376	0.1318	0.1482	0.7413
Prompting	LLAMA2-7B	0.1944	0.0339	0.1326	0.1571	0.7473
Prompting	Mistral2-7B	0.1935	0.0365	0.1351	0.1539	0.7487
Fine-tuning	Zephyr-7B	0.2187	0.0697	0.1657	0.1841	0.7468
Fine-tuning	LLAMA2-7B	0.1951	0.0324	0.1316	0.1567	0.7463
Fine-tuning	Mistral2-7B	0.2142	0.0608	0.1549	0.1753	0.7556

of 0.1753. It also recorded the highest BERTScore of 0.7556. Zephyr-7B closely followed, attaining the highest ROUGE-1 score of 0.2187, along with ROUGE-2 of 0.0697, ROUGE-L of 0.1657, ROUGE-Lsum of 0.1841, and a BERTScore of 0.7468. Although fine-tuned LLAMA2-7B had slightly lower ROUGE scores, it remained competitive with a BERTScore of 0.7463.

Among the prompt-based models, Zephyr-7B achieved the best results with a ROUGE-1 score of 0.1904, ROUGE-2 of 0.0376, ROUGE-L of 0.1318, ROUGE-Lsum of 0.1482, and a BERTScore of 0.7413. LLAMA2-7B followed closely, recording a BERTScore of 0.7473, while Mistral2-7B achieved 0.7487. These findings reinforce that fine-tuning improves question answering performance, following a similar pattern observed in summarization tasks.

5.3 Conclusion

In this chapter, the focus was on analyzing citizen generated data, in contrast to chapter 4, which examined news articles. We assessed the potential of LLMs in classification, summarization, and question answering for citizen feedback. The key takeaway from the classification experiments is that pre-trained LLMs such as BERT and RoBERTa still outperform traditional ML techniques like SVM, GBM, Decision Trees, and Random Forests. The latest generative LLMs from the Qwen and LLAMA families achieved the best results.

Unlike in chapter 4, where LLMs demonstrated reasonable zero-shot performance on news data, their performance on citizen feedback was significantly lower. This discrepancy may be attributed to the private nature of the citizen data, which is unlikely to have been included in pre-training, unlike news articles. The inclusion of a few demonstration examples in the prompts significantly improved performance, with gains plateauing as more examples were added. The high accuracy achieved through fine-

tuning indicates that LLMs can be effectively deployed for classifying citizen feedback. For question answering and summarization, fine-tuning remained the more effective approach, though the performance gap between prompting and fine-tuning was smaller than in classification. In the next chapter 6, the focus will be on non-parametric methods to enhance LLM knowledge, specifically RAG and LLM agents, as alternatives to fine-tuning. These approaches improve the accuracy of LLM responses by integrating relevant context from documents or external tools within prompts.

Chapter 6

Large Language Models for Enhanced Interaction with Government Data Repositories

In the modern digital landscape, government agencies handle vast amounts of digital documents from citizens while also publishing information about their services and policies online to improve transparency and public engagement. However, the sheer volume of content can be overwhelming for both citizens searching for information and government employees processing these documents. This chapter evaluates the potential of unimodal and multimodal LLMs, RAG, and LLM agents in automating the navigation and management of this extensive information. Additionally, it creates unimodal and multimodal datasets for training and evaluating LLMs in this domain, along with datasets specifically designed to assess the performance of RAG systems and LLM agents. This chapter is based on the manuscript: *Augmenting Large Language Models for Enhanced Interaction with Government Data Repositories*.

6.1 Methods

In this chapter, the objective is to leverage LLMs to generate a response y for a given user query x . Language models represent probability distributions over probable sequences of tokens. For generative LLMs, this probability is typically modeled through the NTP objective, where each generated token is conditioned on the preceding tokens and the user query shown in Equation 6.1.

$$p(y|x) = \prod_{i=1}^n p_{\theta}(y_i|y_1, \dots, y_{i-1}, x) \quad (6.1)$$

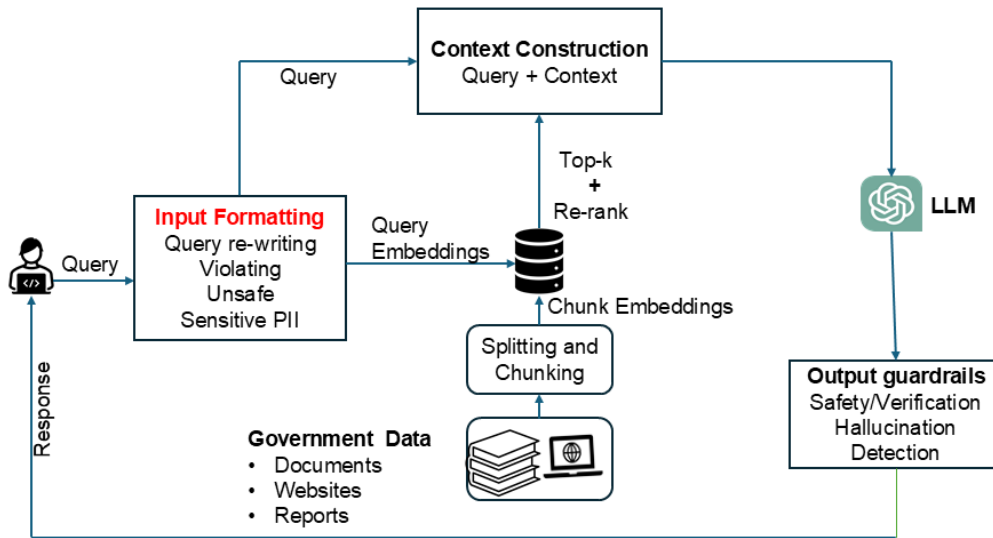


Figure 6.1: An overview pipeline that combines RAG and LLM agents

During prediction, the LLM utilizes the parametric knowledge stored in its parameters θ , which is acquired during pre-training and can be further refined through post-training methods such as instruction tuning and preference alignment [TMS⁺23].

When factual questions fall outside the knowledge encoded in θ , the model often generates incorrect responses known as hallucinations, regardless of its parametric size. The conventional approach to updating θ requires re-training the model with new data, a process that is computationally expensive [DCLT19]. This chapter instead adapts RAG and LLM agents, which are non-parametric approaches that update the knowledge encoded in θ at inference without requiring re-training.

These methods were applied to tasks: question answering, summarization, and the development of intelligent chatbots that can provide information and answer queries related to documents from three Irish government websites: the official Irish government portal (IRISH), the Citizens Information website, and the HSE information site. Additionally, the chatbots can process and generate responses based on documents uploaded by users.

The process begins with extracting documents from the three websites and dividing them into smaller, manageable chunks. Different chunk sizes, ranging from 100 to 2000 tokens, were tested with overlaps of 0, 100, and 200 tokens. The assumption is that introducing overlaps helps maintain continuity, particularly when a document is split at a point where the next section contains critical information. The same chunking method is also applied to user-uploaded documents to ensure consistency in processing.

The chunked documents: $\{d_1, \dots, d_m\}$ are converted into embedding representations,

denoted as $E(d_i)$ for each document. Multiple embedding models are evaluated balancing SoTA performance and computational efficiency, and they include: albert-small-v2, msmacro-distilbert-dot-v5, all-mpnet-base-v2, all-distilroberta-v1, and bge-base-en-v1.5.

These embeddings, along with their corresponding text and any extra information, are stored in a vector database to enable efficient retrieval of relevant information for answering queries. This chapter utilizes Chroma and Facebook AI Similarity Search (FAISS) as vector databases, chosen for their speed and open-source availability. Although the main objective is to retrieve the original text rather than the embeddings, pre-computing and storing embeddings in advance significantly reduces computational overhead during inference compared to generating them dynamically.

For a user query x , the retriever seeks to find and return the most relevant documents from the vector database. The process begins by pre-processing the query. This involves refining unclear or misspelled queries and breaking very long queries into smaller segments to improve both the relevance and quality of the retrieved results [MGH⁺23]. After pre-processing, the query is converted into its embedding representation $E(x)$, which is produced using the same embedding models that were applied to the documents in the vector database. This ensures consistency between the query and document embeddings.

This chapter primarily used similarity search retrieval, which selects documents most similar to the query in the embedding space, but also tested alternative retrieval methods including Max Marginal Relevance Retriever [XXL⁺15] and SVM Retriever [CV95]. The Max Marginal Relevance Retriever selects examples by balancing similarity to the query while ensuring diversity. It identifies embeddings with the highest cosine similarity to the query and iteratively adds them while penalizing those too similar to previously selected examples. The SVM Retriever applies an SVM to rank and retrieve the top k document chunks based on their relevance to the query.

For similarity search retrieval, the similarity ($S(d_i, x)$) between document d_i and query x is measured using the cosine similarity metric, denoted as $\cos$ in Equation 6.2. The top k documents with the highest cosine similarity to the user query are chosen.

$$S(d_i, x) = \cos(E(d_i), E(x)) \quad (6.2)$$

In addition to retrieving the top k documents during the retrieval process, this chapter optionally applies re-ranking to further refine the selection and remove redundant documents. Re-ranking is aimed to improve the relevance of the retrieved document chunks by reassessing their importance. For example, if 4 document chunks are re-

quired, the system initially retrieves the top 20 and then re-ranks them to identify the most relevant 4, ensuring that the most crucial information is prioritized. Re-ranking is performed only on the top k retrieved documents rather than the entire dataset because it is computationally more intensive than the initial retrieval stage. Limiting re-ranking to only top k documents balances efficiency while improving the accuracy of the final selection.

Contextualized Late Interaction over BERT (ColBERT) [SKSF⁺22] is used as a re-ranker. ColBERT is a late-interaction dual encoder that enhances re-ranking by computing fine-grained interactions between query tokens and document tokens. Unlike traditional models that merge token vectors into a single representation, ColBERT retains per-token representations, allowing for more precise similarity calculations and improving the relevance of retrieved documents.

The re-ranking process in ColBERT involves encoding both the query x and each document d_i , where the tokenized inputs are fed to the ColBERT encoder to obtain token-level embeddings. The query embeddings are denoted as: $E(x) = \{E(\mathbf{x}_1), E(\mathbf{x}_2), \dots, E(\mathbf{x}_{N_x})\}$ and the document embeddings as: $E(d_i) = \{E(\mathbf{d}_{i1}), E(\mathbf{d}_{i2}), \dots, E(\mathbf{d}_{iN_d})\}$, where N_x and N_d represent the number of tokens in the query and document, respectively. For each token in the query, the model computes the Maximum Cosine Similarity (MaxSim) with any token in the document using the Equation 6.3.

$$\text{MaxSim}(E(\mathbf{x}_j), E(d_i)) = \max_g \cos(E(\mathbf{x}_j), E(\mathbf{d}_{ig})) \quad (6.3)$$

where $\cos(E(\mathbf{x}_j), E(\mathbf{d}_{ig}))$ represents the cosine similarity between the j -th query token embedding and the g -th document token embedding. The final relevance score $S(x, d_i)$ between the query and the document is obtained by summing the maximum similarities across all query tokens shown in Equation 6.4.

$$S(x, d_i) = \sum_{j=1}^{N_x} \text{MaxSim}(E(\mathbf{x}_j), E(d_i)) \quad (6.4)$$

By using this token-level late interaction mechanism, ColBERT improves document ranking by capturing detailed semantic relationships between query and document tokens, resulting in more precise retrieval and ranking of relevant documents.

RAG utilizes the top documents with the highest similarity scores or those refined by the re-ranker as context, incorporating them into the prompt provided to the LLM during predictions. Consequently, the prediction for the user query x is influenced by the retrieved context denoted as R_k . Thus, the updated probability decomposition for the

LLM, conditioned on the retrieved k documents, is given in Equation 6.5.

$$p(y|x, R_k) = \prod_{i=1}^n p_{\theta}(y_i|y_1, \dots, y_{i-1}, x, R_k) \quad (6.5)$$

For implementation, we utilized the prompt in Figure 6.2 within our RAG framework. In this prompt, "context" represents the top k documents obtained from the retrieval process, while "question" corresponds to the query of interest

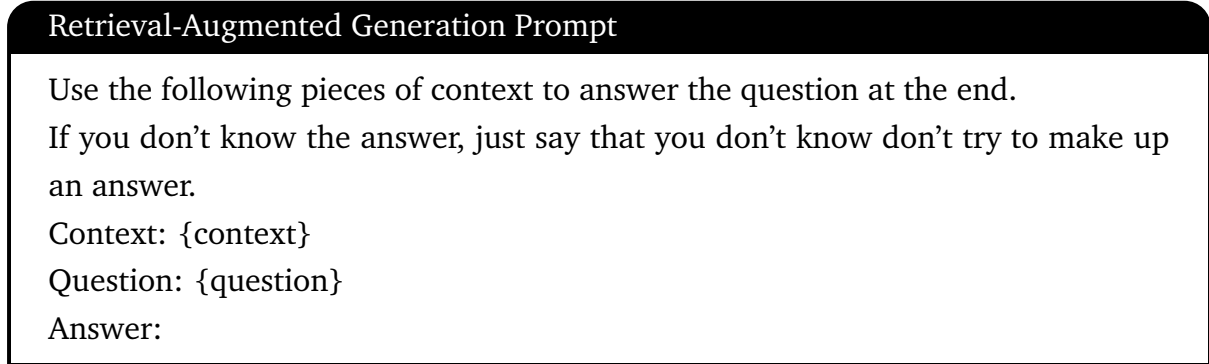


Figure 6.2: RAG Prompt template where the values in brackets (context and question) are replaced with actual values at inference.

6.1.1 Tool Augmented Generation

Since the RAG system requires downloading and indexing documents before LLM generation, this approach may not be sustainable in the long term. To overcome this limitation, this chapter further employs tool augmented LLMs by utilizing LLM agents that can access real-time information through search engines, following the framework proposed by Yao et al. (2023) [YZY⁺23].

When a user submits a query, the LLM first evaluates whether it needs to use external tools to generate a response or if it can rely solely on its parametric knowledge. To facilitate structured reasoning while interacting with these tools, the prompt shown in Figure 6.3 was utilized. The LLM agent had access to two tools: a search engine tool powered by the DuckDuckGo search API, selected for its free accessibility, and a vector database tool for retrieving relevant information from the vector database. The prompt was adapted from a conversational react description agent used in Langchain, guiding the LLM in systematically deciding whether to use a tool, either the vector database or the search API, to acquire necessary information.

Additionally, the prompt instructs the model to organize its output into distinct stages, including thought, action, observation, and final answer, ensuring a structured and logical reasoning process. This iterative approach continues until the agent generates

a well informed and relevant response to the user's query. By following this structured reasoning framework, the LLM agent can dynamically assess when and how to use external tools, improving response accuracy and relevance while maintaining a clear decision making process.

Tool Augmented Generation Prompt

You are InformationGPT, a language model that answers questions by consulting your own knowledge, Government documents about policies and services, and, if necessary, performing online searches. Your decision process for each question should be:

1. Check your own knowledge first.
 - If you are confident you can answer without any external information, answer immediately without using any tools.
2. If you are not confident or need more details, consult the following tools:
 - `vector_database_search`: Searches for information about government services, policies, and updates in a specialized vector database.
 - `DuckDuckGo Search`: Searches information on the internet outside the vector database.
3. Use the tools in this order:
 1. `vector_database_search`
 - If the database provides sufficient information to answer, respond using that information.
 2. `DuckDuckGo Search`
 - Only use this if no relevant information was found in the `vector_database_search`.
 - If `DuckDuckGo Search` provides an answer, provide it.
4. If neither the vector database nor online search yields a clear answer, say: I do not know.

Important:

- You can use more than one tool if you need to confirm the information from multiple sources.

To use a tool, format your response as follows:

Thought: Do I need to use a tool? Yes
Action: [Search or Database]
Action Input: [input]
Observation: [result]

When responding to the Human, or if no tool is needed, use this format:

Thought: Do I need to use a tool? No
AI: [your response here]
Begin!

Previous conversation history:
{chat_history}
New input: {input}
{agent_scratchpad}

Figure 6.3: Tool Augmented Generation Prompt for the LLM agent

6.1.2 Data

This section describes the datasets, their sources, and the generation process used throughout this chapter of the Thesis. While most experiments are based on text only documents, some also include documents containing both text and images to evaluate the capabilities of multimodal LLMs. Portions of these datasets were divided into document chunks, with some serving as external knowledge stored in a vector database for the RAG application described in section 6.1. Other chunks were used to create datasets derived from both synthetic generation and human evaluations, supporting the training and assessment of LLMs in this domain.

Text Data: Documents were collected from public IRISH government websites to create training and evaluation datasets. Sources include: the official government publications portal¹, the HSE², and the Citizens Information website³. These documents cover a broad spectrum of topics, including government policies, procedures for obtaining official documents, and press releases. The first application of these documents was to segment them into manageable chunks and store them in a vector database, supplementing the context provided to LLMs in a RAG system.

The second application of these datasets involved creating instruction tuning datasets comprising potential questions and answers that citizens might ask based on the provided information. To generate synthetic datasets, different LLMs were used. One set of models generated questions, while another set generated answers to those questions. This approach ensured diversity in both question formulation and response generation, simulating a realistic range of citizen inquiries.

To minimize hallucinations, the dataset generation process required LLMs to derive both questions and answers from specific document chunks rather than relying on their parametric knowledge. Extracted text from government websites was provided as context in the prompt, guiding the LLMs to generate factual responses based strictly on the given content. The prompt used for this task is detailed in Figure 6.4. This process resulted in a dataset containing approximately 800,000 question-answer pairs, along with the corresponding source text. The generated responses were assessed based on groundedness, relevance, and standalone quality with their description shown in Table 6.1.

A RAG evaluation dataset was created by developing a set of human curated questions. These questions were designed so that some could be answered using documents stored in the vector database. Different LLMs were used to generate responses by retrieving

¹<https://www.gov.ie/>

²<https://www.hse.ie/>

³<https://www.citizensinformation.ie>

relevant context from the vector database and incorporating it into the prompt. The generated responses were then evaluated by a human to determine whether they adequately addressed the question and if they were free from hallucinations.

A total of 260 curated questions were developed across IRISH government publications and HSE documents. Each question was evaluated using three different models, resulting in a total of 780 test examples. These datasets provide a test set for assessing the performance of RAG applications, which currently lack widely available evaluation sets.

Data Generation Prompt

Your task is to generate a factual question and provide an answer using the given context.

The question should be answerable with a specific, concise piece of factual information from the context.

Make sure the question is framed similarly to what users might ask in a search engine.

Avoid phrases like “according to the passage” or “context” in your question.

Format your response as follows:

Output:

Fact-based question: (your fact-based question)

Answer: (your answer to the fact-based question)

Here is the context.

Context: {context}

Output:

Figure 6.4: Data Generation Prompt template, the context in brackets is replaced by actual document chunks at inference.

Statistical Graphs: Some government websites contain highly multimodal information, such as government reports that integrate extensive visual content alongside text. To evaluate LLMs for processing such documents, we used reports from the CSO, which feature a combination of textual data and numerous visual elements. The CSO is an independent government agency responsible for collecting and disseminating statistical data related to the Irish economy, society, and various activities. From the CSO website, we collected charts along with their associated textual captions. These captions do not summarize or interpret the charts; rather, they offer descriptive explanations of the contents presented in the reports.

To create a visual instruction tuning dataset, we used these textual captions alongside

text-based LLMs. For each caption, an LLM was prompted to generate relevant questions and answers based on the provided text, following the same methodology used for text-based datasets, as shown in Figure 6.4. The key distinction in this process is that each generated question-answer pair is explicitly linked to its corresponding chart or report. For simplicity, we focused on single-page reports or charts. This resulted in a dataset comprising 460 examples, including questions, answers, textual descriptions, and associated visuals. This dataset provides a resource for training and evaluating multimodal LLMs for government applications.

6.1.3 Evaluation of the Language Model Generation

In this chapter, generative LLMs are employed to generate responses that are typically open-ended or abstractive in nature. A major challenge is ensuring the accuracy of these outputs by filtering out wrong and misleading outputs. This is particularly difficult as many applications lack definitive ground truth answers for direct comparison. Even when reference answers are available, LLMs generate free-form text, where different variations may convey the same meaning. Ideally, human reviewers would verify the correctness and relevance of the generated responses. However, manual evaluation is often impractical, especially in production environments where queries can originate from multiple users simultaneously.

Similarity metrics such as: ROUGE [Lin04] and BLEU [PRWZ02] (see section 5.1.4 for details), alongside LLM judges [ZCS⁺23] were used to evaluate different LLM outputs. For LLM judge evaluation, models are prompted to determine whether the generated response aligns with the given question, providing a binary “yes” or “no” judgment. In some instances, LLMs assign a score between 1 and 5, based on how effectively the response addresses the query, following the evaluation framework outlined in Figure 6.5.

Answer Evaluation Prompt

Your role is to assess answers submitted by different language models and assign them scores based on predefined criteria.

Included in this task are a directive (potentially containing an input), an answer for assessment, an exemplary answer rated at a perfect score of 5, and detailed scoring guidelines.

1. Offer specific critiques based on the provided scoring guidelines, avoiding broad judgments.
2. Present a score, ranging from 1 to 5, strictly adhering to the provided scoring guidelines.
3. Format your feedback as a JSON object with the following structure: 'score': a number between 1 and 5.
4. Exclude any introductory or concluding remarks and focus solely on delivering the score.

The directive for evaluation:

instruction

Answer to assess:

{response}

Exemplary answer:

{reference_answer}

Evaluation Guidelines:

[Assess whether the answer aligns with the exemplary answer in terms of correctness, precision, and factual content.]

Score 1: The answer is entirely wrong, imprecise, and/or factually incorrect.

Score 2: The answer is predominantly wrong, imprecise, and/or factually incorrect.

Score 3: The answer shows partial correctness, precision, and factual content.

Score 4: The answer largely demonstrates correctness, precision, and factual content.

Score 5: The answer perfectly aligns with correctness, precision, and factual content.

score:

Figure 6.5: Answer Evaluation Prompt, the values in brackets, response is replaced with the actual answer generated by the model and reference_answer is replaced with the ground truth for that question

To evaluate the usefulness of questions generated by LLMs during synthetic data generation, we focused on three key metrics: relevance, groundedness, and standalone.

In the context of this chapter of the Thesis, groundedness assesses whether the question can be answered from the given context. Relevance evaluates if the generated questions or answers are relevant to users. Standalone determines if the question is understandable without any context for someone with domain knowledge [ZCS⁺23]. These metrics are assessed through the prompts detailed in Table 6.1.

Table 6.1: Overview of the prompts used to evaluate the groundedness, relevance and standalone of the generated questions.

Evaluation Criteria	Prompt Description
Groundedness	You will be given a context and a question. Provide a “score” (1-5) on how well the question can be answered with the context. 1 = not answerable, 5 = clearly answerable. Your answer should be in json format: “score”: (1-5). No explanations needed. Question: {question} Context: {context} Answer:
Relevance	You will be given a question. Provide a “score” (1-5) on how useful the question is for understanding IRISH government services. 1 = not useful, 5 = extremely useful. Answer in json: “score”: (1-5). No explanations needed. Question: {question} Answer:
Standalone	You will be given a question. Provide a “score” (1-5) on how context-independent the question is. 1 = needs context, 5 = makes sense alone. Answer in json: “score”: (1-5). Question: {question} Answer:

LLM agent and Tool Evaluation: To assess how LLMs interact with external tools, we curated a tool evaluation dataset comprising of manually crafted questions related to government services and documents, with a particular focus on current issues. Unlike

the dataset used for RAG evaluation, this dataset includes a diverse set of question types such as informational queries, numerical calculations, logical reasoning tasks, and multi-step problems requiring operations like addition, data aggregation, or retrieving details from multiple paragraphs within a single document or across different sections of various documents.

Of the total questions, 63 could be answered using the documents stored in the vector database, while another 63 required the search tool, since the relevant information was not present in the vector database but could be obtained from the internet.

These questions were presented to the LLM with structured instructions, as illustrated in Figure 6.3. The LLM was expected to follow a systematic approach: first assessing whether the question could be answered using its internal knowledge, then retrieving information from the vector database if necessary, and finally utilizing the search tool if no relevant data was found. This process allowed us to capture the reasoning trajectory of the LLM, including the sequence of actions taken, tools accessed, and their respective outputs.

Tool usage was evaluated by a human reviewer who analyzed the steps taken by the LLM agent to determine whether the appropriate tool was selected for each query. This assessment was independent of whether the tool ultimately led to a correct answer; the focus was solely on verifying if the LLM invoked the correct tool based on the query requirements.

6.1.4 Implementation and Language Models Used

This chapter utilized both unimodal and multimodal LLMs, selected based on their performance, model size, and the availability of detailed technical reports or research papers describing their training process. Most of the open-source models used belong to the LLAMA and Mistral families, as other open-source models available at the time were primarily improvements or fine-tuned versions of these. For closed-source models, OpenAI's models and Gemini were primarily used due to their SoTA performance and extensive adoption within the research community.

The specific versions of the models used in this research can be accessed via Huggingface for open-source models and through their respective websites for closed-source models. These include: LLAMA2-7B, LLAMA2-13B, LLAMA3-8B, LLAMA3-70B, Zephyr-7B, Mistral2-7B, Mistral3-7B, Mistral2-8X7B, LLAVA-7B, LLAVA-13B, Gemini, GPT-40-mini, GPT-4, and GPT-3.5-Turbo.

The experiments and systems described in this chapter were implemented using Huggingface Transformers, Langchain, and Streamlit. The LLMs referenced include mainly

open-source models from the Huggingface platform, along with some closed-source models from OpenAI used in tool-augmented language models. The source code and data are available at the github page: https://github.com/TrustPaul/inter_convqa.

6.2 Results and Discussion

This section presents a discussion of results for various models including unimodal and multimodal LLMs for processing government document publications. It discusses the effectiveness of different LLM data generators used to create datasets from these documents. Additionally, it analyzes the performance of RAG systems and LLM agents, focusing on how different hyper-parameter configurations influence the accuracy of LLM generated responses in this domain.

Table 6.2: Accuracy scores for different LLM data generators across relevance, groundedness, and standalone criteria.

Dataset	LLM Data Generator	Evaluation Criteria		
		Groundedness	Relevance	Standalone
CSO	Mistral2-7B	2.93	20.77	20.77
	LLAMA3-8B	75.62	9.03	8.58
	Mistral3-7B	4.51	45.37	46.28
HSE	Mistral2-7B	96.88	51.92	51.92
	LLAMA3-8B	95.19	48.08	50.98
	Mistral3-7B	83.65	65.35	65.12
IRISH	Mistral2-7B	82.00	51.33	52.67
	LLAMA3-8B	76.67	52.35	50.00
	Mistral3-7B	80.67	59.33	62.00

Accuracy of LLM Generators: Table 6.2 presents the accuracy scores for different LLM data generators: Mistral2-7B, LLAMA3-8B, and Mistral3-7B across the CSO, HSE, and IRISH government publications datasets. The generated questions and answers were evaluated by LLM judges according to three criteria: relevance, groundedness, and standalone quality, with detailed instructions provided in Table 6.1. Judges assigned scores on a scale from 1 to 5 for each criterion, and prediction accuracy was defined as the proportion of responses that received scores of 4 or 5 out of the total number of responses.

On the CSO dataset, among the LLM data generators, Mistral3-7B achieved the highest accuracy in relevance and standalone criteria, while LLAMA3-8B had the best score in groundedness with an accuracy of 75.62%. On the HSE dataset, Mistral2-7B scored highest in both groundedness and relevance, while Mistral3-7B led in the standalone

category. For the IRISH dataset, Mistral2-7B had the highest accuracy in groundedness and standalone, while LLAMA3-8B performed best in relevance.

Across all datasets, the accuracy scores were lower on the CSO dataset compared to the others. This could be due to the questions being generated from image captions or descriptions rather than standard text, which may lead to questions that do not reflect typical information-seeking queries. As a result, the LLM judges may have assigned lower scores. Among the evaluated metrics, the LLM data generators scored higher on average in the groundedness criterion. Performance was more balanced across models in the relevance and standalone criteria. Overall, there was no single model that consistently outperformed others across all metrics and datasets.

Table 6.3: Accuracy scores from human evaluations, LLM judge assessments, and similarity metrics: ROUGE and BLEU for different RAG systems on the IRISH and HSE datasets.

Dataset	Model	Evaluators					
		Human	LLAMA3-8B	Mistral2-8X7B	Mistral3-7B	ROUGE	BLEU
IRISH	LLAMA2-7B	79.58	50.32	94.57	98.08	0.5962	0.4564
	LLAMA2-13B	90.21	67.74	97.60	95.20	0.7903	0.6615
	Mistral2-7B	76.39	52.72	55.91	97.60	0.6111	0.4857
HSE	LLAMA2-7B	81.96	91.73	81.20	94.74	0.5511	0.3989
	LLAMA2-13B	96.12	96.99	93.23	97.74	0.7816	0.6156
	Mistral2-7B	88.72	96.99	93.98	95.49	0.6936	0.5361

Accuracy of different RAG Systems: For RAG evaluation, a combination of automated and human evaluations were used to assess the performance of different RAG generation models: LLAMA2-7B, LLAMA2-13B, and Mistral2-7B. The automated evaluation involved using LLM judges alongside similarity metrics: BLEU and ROUGE to compare RAG generated responses with reference texts. For LLM judge-based assessments, accuracy was determined following the same scoring method used for assessing LLM generators but instead a different prompt shown in Figure 6.5 was used.

On the IRISH dataset, as shown in Table 6.3, LLAMA2-13B demonstrated the best overall performance across most evaluators. It achieved the highest human evaluation accuracy of 90.21% and received top scores from two LLM judges: 67.74% from LLAMA3-8B and 97.60% from Mistral2-8X7B. The only exception was Mistral3-7B, which rated LLAMA2-7B slightly higher. LLAMA2-13B also achieved the highest similarity scores, with a ROUGE score of 0.7903 and a BLEU score of 0.6615, indicating strong alignment between its generated outputs and the reference answers.

Among the models with 7 billion parameters, LLAMA2-7B performed better than Mistral2-7B in human evaluation, achieving an accuracy of 79.58% compared to

76.39%, a difference of 3.19%. However, in similarity-based evaluations, Mistral2-7B scored slightly higher, with a ROUGE score of 0.611 versus 0.5962 and a BLEU score of 0.4857 compared to 0.4564. In the LLM judge evaluation, LLAMA3-8B rated Mistral2-7B higher at 52.72% compared to 50.32% for LLAMA2-7B, whereas Mistral2-8X7B showed a preference for LLAMA2-7B, assigning it a score of 94.57% versus 55.91%. In contrast, Mistral3-7B gave both models nearly identical scores, with ratings close to 98%.

A similar pattern was observed when the models were evaluated on the HSE dataset, as shown in Table 6.3. LLAMA2-13B demonstrated the highest factual correctness, achieving a human evaluation accuracy of 96.12%. Among the 7 billion parameter models, Mistral2-7B outperformed LLAMA2-7B for human evaluation with an accuracy of 88.72% compared to 81.96%. In terms of ROUGE and BLEU scores, LLAMA2-13B again led with a ROUGE score of 0.7816 and a BLEU score of 0.6156, followed by Mistral2-7B (ROUGE 0.6936, BLEU 0.5361), with LLAMA2-7B scoring the lowest.

For LLM judge evaluations, LLAMA2-13B consistently achieved the highest accuracy for most of the evaluators. Among the 7 billion parameter models, Mistral2-7B received the highest ratings from most LLM judges. LLAMA3-8B assigned Mistral2-7B an accuracy of 88.92% versus 81.96% for LLAMA2-7B, while Mistral2-8X7B rated Mistral2-7B at 93.98% compared to 81.20% for LLAMA2-7B.

Overall, the models achieved higher accuracy when processing content from the HSE website compared to the IRISH publications site. This difference in performance may be related to the type of content on each platform. The IRISH website mainly features public statements and updates that often include numerical figures, which can vary over time and makes it harder for models to produce accurate responses. In contrast, information from the HSE site is more factual and stable, even though it may be updated occasionally. Because this type of health information tends to remain consistent across time and place, it is likely that some of it is already captured in the models' parametric knowledge, which helps improve their accuracy on this type of content.

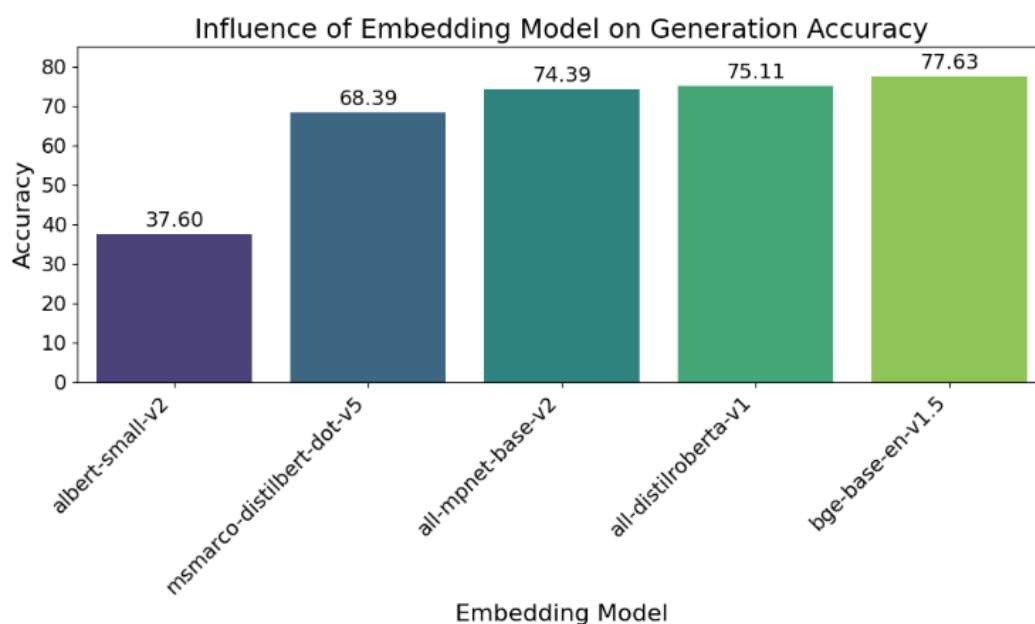


Figure 6.6: A bar graph showing the influence of an embedding model on RAG accuracy

Influence of the choice of an embeddings model: A wide range of embedding models are introduced regularly, making it essential to assess their influence on the performance of downstream RAG systems. To investigate this, we conducted experiments with different embedding models to determine their effect on LLM generation accuracy. The findings illustrated in Figure 6.6, show that the choice of embedding model may have a significant influence on performance.

We evaluated five embedding models: albert-small-v2, msmacro-distilbert-dot-v5, all-mpnet-base-v2, all-distilroberta-v1, and bge-base-en-v1.5. The results showed clear performance differences between the first three models and the last two. The bge-base-en-v1.5 model achieved the highest average accuracy of 77.63%, nearly 40% higher than the lowest performing model, albert-small-v2, which had an accuracy of 37.60%. The second best performing model, all-distilroberta-v1, attained an accuracy of 75.11%, followed by all-mpnet-base-v2 at 74.39%. These findings highlight the importance of selecting the most suitable embedding model when designing RAG systems, as it directly affects the accuracy of generated responses.

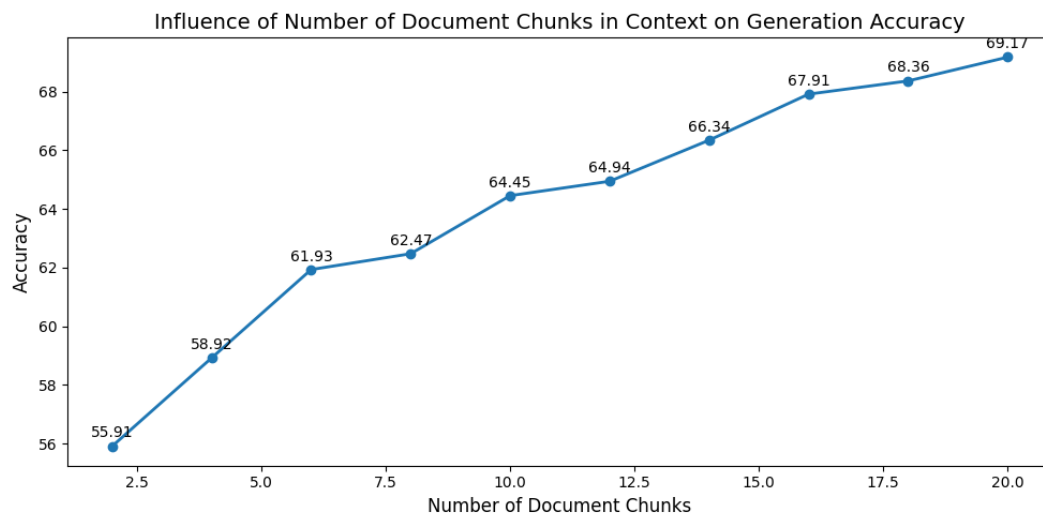


Figure 6.7: A line graph showing the influence of the number documents in a context on generation accuracy

Influence of the Number of document Chunks: One of the most critical hyperparameters in a RAG system is the number of document chunks, as these chunks inject non-parametric knowledge into the LLM's context. The number of chunks included in the context is an essential consideration, as too few chunks may limit the available context, while too many may exceed the model's processing capabilities. Determining the optimal number of document chunks is crucial for maximizing accuracy.

Figure 6.7 shows the relationship between generation accuracy and the number of document chunks in the context. Our experiments tested chunk numbers ranging from 2 to 20, as beyond 20 exceeded the models' maximum input length. The results show a positive correlation between the number of document chunks and generation accuracy up to a certain chunk number tested (20 chunks). The lowest accuracy (55.91%) was observed with only 2 chunks in the context, while accuracy increased consistently as more chunks were included, reaching its highest value with 20 chunks.

Nearly every 2-point increase in the number of chunks resulted in improved accuracy. Further experiments with LLMs that support larger context windows would be valuable to examine whether this trend continues, stabilizes, or declines at a certain point.

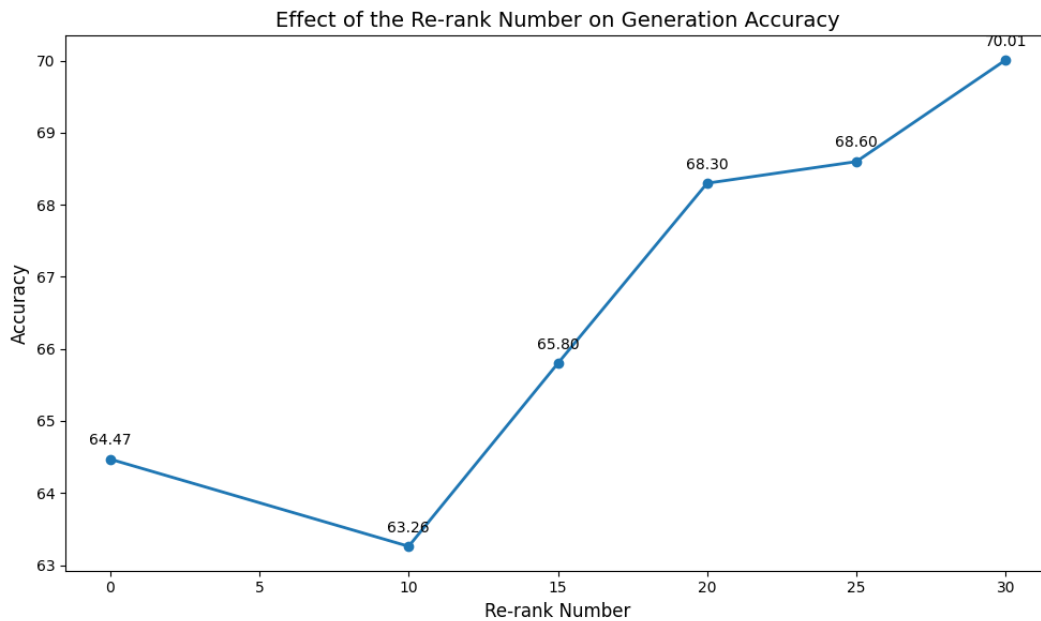


Figure 6.8: A line graph showing the influence of number of re-rank documents on generation accuracy

Influence of re-rank Number on Generation Accuracy: Instead of simply selecting the top k documents for inclusion in the context, an alternative approach is to initially retrieve more than k documents and then perform a re-ranking step to extract a more refined subset of the top k . In Figure 6.8, we present the results of experiments designed to evaluate whether this re-ranking strategy provides any improvements in RAG generation accuracy.

Figure 6.8 is a line graph that shows how different re-rank numbers with a constant $k = 4$ in the context affect generation accuracy. The accuracy value of 64.47% at 0 indicates that no re-ranking was applied, serving as our baseline observation. The best accuracy of 70.01% which was obtained with re-ranking is approximately 6 points higher than without re-ranking (64.47%).

The generation accuracy of the RAG system improves with higher re-rank numbers. However, using a re-rank number of 10 to select 4 document chunks results in slightly lower accuracy compared to not applying re-ranking (63.26% versus 64.47%). Overall, increasing the re-rank number up to 30 leads to better generation accuracy, showing that using a higher re-rank number can improve RAG system performance.

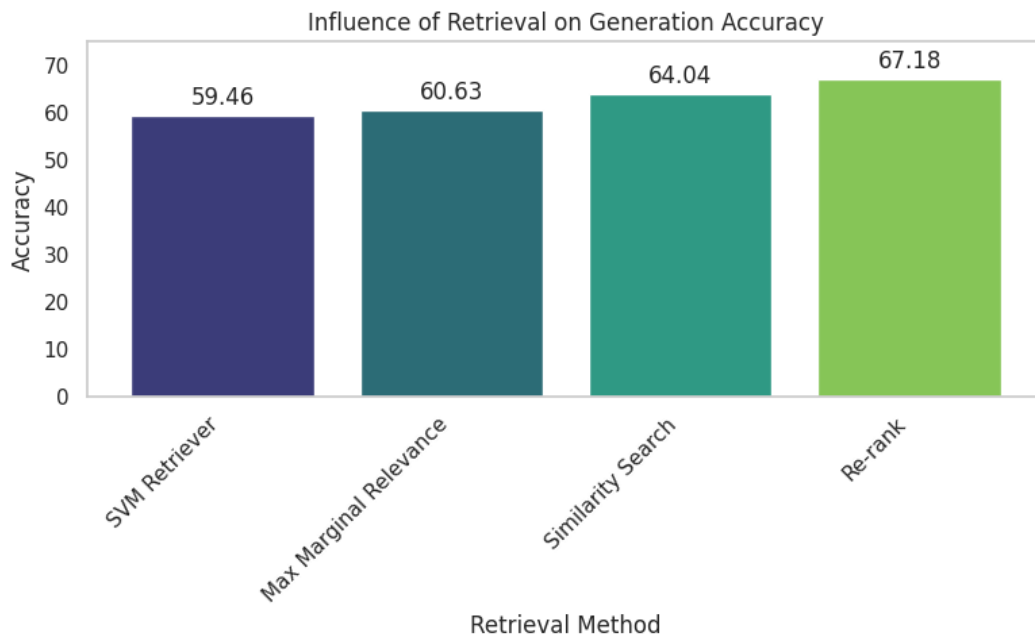


Figure 6.9: A bar graph showing the influence of retrieval method on generation accuracy

Influence of the choice of the Retrieval Method: Different retrieval strategies are used to retrieve document chunks for the RAG system. In this chapter, we compared four methods: SVM Retriever, Max Marginal Relevance, Similarity Search, and Re-rank to examine their influence on performance. Figure 6.9 shows a bar graph that presents the performance differences among these methods.

The results show that Re-rank is the most effective, with an average generation accuracy of 67.18%, followed by Similarity Search at 64.04%. The SVM Retriever and Max Marginal Relevance methods had lower accuracy, with 59.46% and 60.63% respectively, both close to 60%.

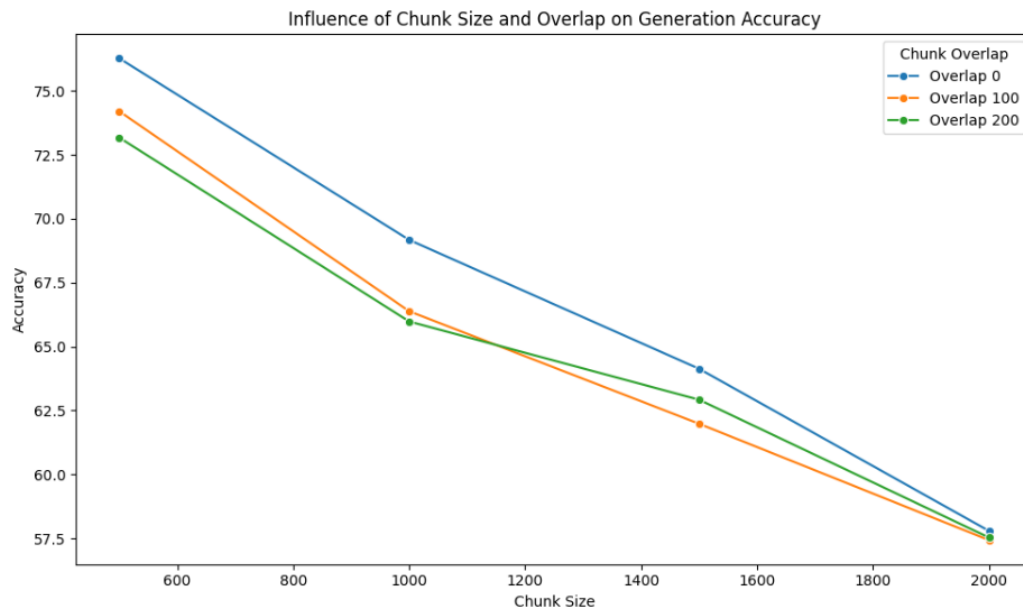


Figure 6.10: Line graphs showing the influence of Chunk Size and Overlap on generation accuracy

Influence Chunk Size and Overlap on Generation Accuracy: Before selecting document chunks for inclusion in the context, the source document must first be divided into smaller segments. In designing a RAG system, it is important to determine the appropriate number of tokens for each chunk. Figure 6.10 presents our analysis of how chunk size affects the accuracy of RAG outputs. We also examine the role of chunk overlap, which refers to the number of tokens shared between two consecutive chunks. The underlying assumption is that overlapping content may help preserve semantic continuity between adjacent chunks.

The results in Figure 6.10 show that larger chunk sizes generally lead to lower generation accuracy. As chunk size increases from 100 to 2000 tokens, accuracy decreases from 75% to 50%, suggesting that smaller chunks are more effective in RAG design. In addition, greater chunk overlap, tested at 0, 100, and 200 tokens, appears to reduce generation accuracy, particularly when chunk sizes are large.

Tool selection Accuracy: In Figure 6.11, we present the confusion matrix illustrating the performance of different LLMs in selecting the appropriate tools. This chapter focuses on two tools: the vector database and the search engine. Additionally, the confusion matrix includes a "no tool" label, which indicates cases where the LLM relied solely on its internal parametric knowledge to generate an answer, even though external information was required. It is important to note that these values reflect whether the correct tool was invoked and do not assess whether the final response was accurate. The values in the confusion matrix represent the actual number of documents in each category, not accuracy scores. These counts were later converted into accuracy values,

which are discussed next.

Among the four LLMs evaluated: LLAMA3-8B, LLAMA3-70B, Mistral3-7B, and GPT-40-mini, LLAMA3-70B achieved the highest tool selection accuracy at 97.61%, followed by GPT-40-mini and LLAMA3-8B, both with an accuracy of 90.48%. Mistral3-7B had the lowest performance, achieving 64.29%. A closer examination of the confusion matrix reveals that GPT-40-mini was the only model that frequently relied on its parametric knowledge instead of consulting external tools when necessary. Specifically, it answered ten questions using internal knowledge instead of retrieving information from the vector database and did the same for two questions that required the search engine. In contrast, the other LLMs at least attempted to use a tool, even if they selected the incorrect one.

Analyzing the predicted values for the vector database and search engine, all LLMs demonstrated high accuracy when selecting the search engine as the correct tool. Even the lowest-performing model, Mistral3-7B, correctly identified the search engine as the appropriate tool in 60 out of 63 instances. However, for questions that required the vector database, most models incorrectly defaulted to using the search engine, except for GPT-40-mini, which relied on its internal parametric knowledge. These results suggest that, given clear instructions, most LLMs can correctly select the appropriate tools. However, this evaluation only considered two tools, which may have simplified the selection process.

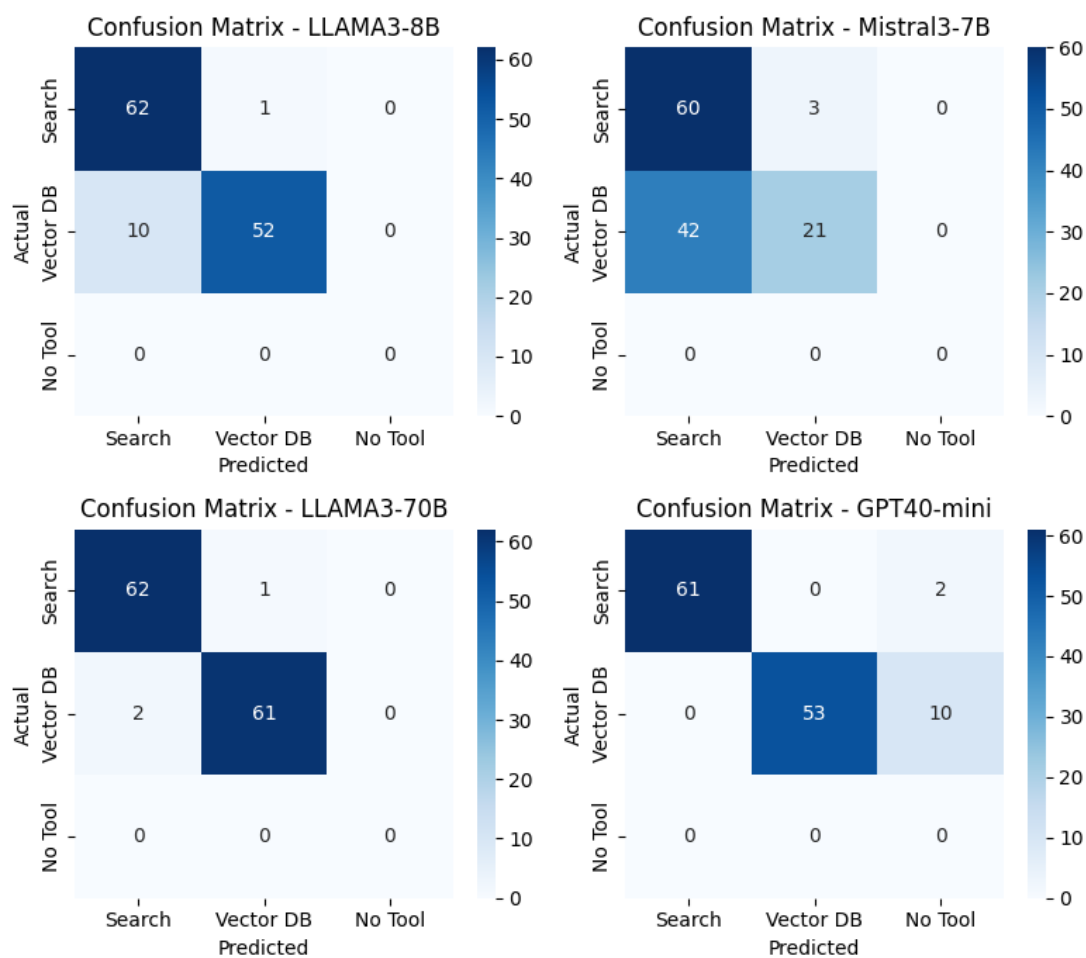


Figure 6.11: Confusion matrix for different modes of tool selection. The values in the confusion matrix and on the scale represent the number of documents.

Multimodal LLM Results: In addition to the RAG experiments, we evaluated the ability of LLMs to process and answer questions based on government documents that include both text and images. The results are based on experiments using multimodal LLMs applied to reports from the CSO. These experiments used both open and closed source models: GPT-4-Vision, Gemini, LLAVA-7B, and LLAVA-13B.

To assess the quality of the generated outputs, we employed LLM judges and applied the same evaluation criteria that was used for assessing RAG responses. While the generated responses were in text format, the inputs to these models could either be text only or a combination of text and images.

Table 6.4: Accuracy scores for different multimodal LLMs evaluated by humans and different LLM judges.

Multimodal LLM	Evaluators			
	Human	LLAMA3-8B	Mistral2-8X7B	Mistral3-7B
Gemini	83.26	92.10	81.47	81.70
LLAVA-7B	15.00	62.16	27.03	32.43
LLAVA-13B	21.60	61.80	27.19	32.13
GPT-4-Vision	91.61	99.09	99.55	98.87

Table 6.4 presents the performance of various multimodal LLMs, including accuracy scores from human evaluations. Among all models, GPT-4-Vision achieved the highest performance, with a human evaluation accuracy of 91.61% and approximately 99% accuracy from all the LLM judges. Gemini ranked second, receiving a human accuracy score of 83.26%, 92.10% from the LLAMA3-8B judge, and around 81% from both Mistral2-8X7B and Mistral2-7B.

The LLAVA models were the lowest-performing multimodal LLMs in our experiments. Although LLAVA represented the SoTA among open-source multimodal models at the time, the results reveal a clear performance gap between open-source models like LLAVA and closed-source models such as GPT-4-Vision and Gemini in multimodal document understanding.

Table 6.5: Correlation and agreement metrics (MCC and CK) between different LLM judges and the human evaluator

Dataset	Model	MCC	CK
IRISH	LLAMA3-8B	0.4157	0.3717
	Mistral2-8X7B	0.6264	0.6208
	Mistral3-7B	0.4285	0.3904
HSE	LLAMA3-8B	0.0945	0.0873
	Mistral2-8X7B	0.1199	0.1187
	Mistral3-7B	0.1728	0.1545

Correlation of Different LLM judges with Human Evaluators: Table 6.5 shows the correlation metrics (MCC and CK) among different LLM judges: LLAMA3-8B, Mistral2-8X7B, and Mistral3-7B with human evaluations. On the IRISH dataset, Mistral2-8X7B had the strongest alignment with human judgments, achieving an MCC of 0.6264 and a CK of 0.6208. Both LLAMA3-8B and Mistral3-7B showed slightly lower but comparable results.

In contrast, correlation scores were lower on the HSE dataset. On this dataset, Mistral3-7B performed best with an MCC of 0.1728 and a CK of 0.1545, while LLAMA3-8B had the lowest agreement. These results suggest that LLM judges align more closely with human evaluations on the IRISH dataset than on the HSE dataset.

To measure agreement among the LLM judges themselves, Krippendorff's Alpha was used, resulting in 0.5796 for the IRISH dataset and 0.4300 for the HSE dataset. The moderate correlation with human evaluations and the relatively high agreement among the judges indicate that these LLM judges can be used to evaluate outputs from other LLMs.

6.3 Limitations

Many of our experiments and demonstrations used open-source models like those from LLAMA and Mistral families. While these models may not match the performance of proprietary ones like GPT-4, accessing the latter's APIs can be expensive. Smaller versions of these models may also fail to produce desired outputs during interactions with LLM agents. Given the rapidly evolving landscape of LLM and continuous updates, the outputs and behaviors observed in this chapter may vary in future versions of the models. Some of the evaluations in our experiments were conducted by other LLM judges, which, while providing a proxy measure of system output accuracy, may not fully replicate human evaluation.

Lastly, the models used for RAG, LLM agents, and LLM judges generate probabilistic outputs, which can still be incorrect even when supplemented with external information. Therefore, it is important to keep this in mind when relying on their outputs, as they may contain errors or false information, and their accuracy should be further verified.

6.4 Conclusion

This chapter examined the potential of SoTA LLMs, including ChatGPT, Mistral, and LLAMA for summarization and question answering of public sector documents. Several datasets were curated, either through human annotation or synthetic generation to assess different LLM capabilities in this domain. Extensive experiments were conducted to develop effective RAG systems for this application. The findings highlight that the choice of embedding models, re-rankers, retrievers, and other RAG components significantly impact performance. The integration of RAG substantially improved LLM accuracy, with models like LLAMA2-13B achieving up to 90% generation accuracy.

Earlier versions of LLMs faced challenges in following complex instructions and generating well-structured outputs when used as LLM agents. Recent models demonstrated significant improvements in formatting responses and selecting the appropriate tools. It is worth noting that this research was limited to evaluating only two tools in this context: a vector database and a search tool.

A significant performance difference was observed between proprietary multimodal LLMs, such as GPT-4 and Gemini, and open-source counterparts like LLAVA. Although subsequent versions of open-source models might have narrowed this gap, closed-source models demonstrated better performance during the evaluation period.

Although the methods applied in this chapter reduced hallucination rates and improved LLMs' ability to respond to queries beyond their parametric knowledge, the generated answers remain probabilistic. Therefore, implementing output guardrails is necessary to filter out potential hallucinations. These findings underscore the importance of further research into hallucination detection techniques, which are the focus of chapter 7.

Chapter 7

Hallucination Detection in Large Language Model outputs

LLMs like ChatGPT are increasingly applied across various domains and applications. However, they can produce confident yet incorrect responses, commonly referred to as hallucinations, which pose significant risks in scenarios where their outputs are used to make decisions in sensitive applications. Methods such as RAG and LLM agents reduce hallucination rates by incorporating external knowledge that was not part of the model's training data, as discussed in chapter 6. Despite improvements through these techniques, hallucinations can still occur. Therefore, identifying LLM outputs containing these hallucinations is essential to prevent the use and dissemination of misleading outputs from LLMs.

The contributions of this chapter focus on the adaptation and evaluation of two methods for detecting hallucinations in LLM-generated outputs. The first approach adapts and assesses the use of hallucination and faithfulness scores to classify hallucinations in LLM responses generated by a RAG system for public sector documents. The second approach proposes weakly supervised learning for hallucination detection in LLM responses, where hallucination classifiers are trained using labels generated by other LLMs instead of human annotators. This chapter is based on the manuscript: *Weakly Supervised Distillation Learning for Hallucination Detection*.

7.1 Methodology

This section describes the methodology for two scenarios of hallucination detection proposed in this chapter: scenario 1, based on hallucination and faithfulness scores, and scenario 2, based on weakly supervised learning. Scenario 1 relies on RAG outputs generated using the LLM RAG generators discussed in chapter 6, with hallucinations in

LLM responses classified by a human annotator. Scenario 2 detects hallucinations from LLM responses without using reference contexts, applying weakly supervised learning on the Shared-task on Hallucinations and Related Observable Overgeneration Mistakes (SHROOM) dataset [MZV⁺24]. The details of the datasets in both scenarios are provided in section 7.1.3, and the methodology is discussed next.

7.1.1 Hallucination and Faithfulness Scores for Hallucination Detection

In this subsection, we discuss the methodology for scenario 1, where the hallucination score quantifies how much the LLM-generated response contradicts the given context, while the faithfulness score evaluates whether the claims in the response can be verified based on the provided information in the context.

To compute these scores, this chapter uses an automatic approach using LLM judges with detailed instructions given to the judges described in two prompts. The first prompt shown in Figure 7.1 is designed to guide the LLM judges to detect hallucinations by identifying contradictions in the generated response relative to the provided context.

The LLM judge is given a question (or instructions given to the LLM), an LLM-generated response, and a set of context sentences, which are derived by segmenting the context used during RAG into individual sentences. The LLM judge then reviews each context sentence independently to determine whether the generated response is consistent with the given information. If the response aligns with the context, the LLM judge outputs "Yes", otherwise it outputs "No".

Hallucination Score Prompt

Question: {QUESTION}

Answer: {MODEL_ANSWER}

Context (each sentence is a separate entry): {CONTEXT SENTENCES}

For each sentence in the context, decide whether the provided answer is consistent with the sentence. Answer with "Yes" if the answer is consistent and "No" if the answer contradicts the sentence.

Figure 7.1: Hallucination Score Prompt. Values in curly brackets are replaced with actual values at inference.

The hallucination score is calculated by determining how often the LLM's response

contradicts the provided context. Let N_h be the total number of context sentences extracted from the original context, and let C_h represent the number of sentences that contradict the generated answer. A contradiction occurs when the LLM judge labels as "No". The hallucination score is then calculated using Equation 7.1.

$$\text{Hallucination Score} = \frac{C_h}{N_h} \quad (7.1)$$

The hallucination score measures the proportion of context sentences that contradict the generated responses. A higher hallucination score indicates a greater inconsistency between the response and the context, reflecting a higher level of hallucination in the LLM's response.

The second prompt, shown in Figure 7.2, is used to evaluate the faithfulness of LLM responses, which assesses whether the claims made in the generated answer are supported by the given context. The process consists of two steps: First, an LLM judge extracts individual claims from the response of the LLM under evaluation. Then, it evaluates each claim against the context and determines whether it is supported, contradicted, or unverifiable. The LLM judge is instructed to respond with "Yes" if the claim is supported by the context, "No" if it contradicts the context, and "Unverifiable" if there is insufficient evidence.

Faithfulness Score Prompt

Context: {CONTEXT SENTENCES}

Extracted Claims from Answer: {CLAIMS}

For each claim, decide whether it is supported by the context. Answer with "Yes" if the claim is supported by the context, "No" if the claim contradicts the context, and "Unverifiable" if there is insufficient evidence.

Figure 7.2: Faithfulness Score Prompt. Values in curly brackets are replaced with actual values at inference.

The faithfulness score quantifies the extent to which the claims in the LLM-generated response are supported by the given context. Let N_f represent the total number of claims extracted from the response, and let C_f denote the number of truthful claims. A claim is considered truthful if it is explicitly supported by the context (when the LLM judge outputs "Yes"). The faithfulness score is calculated using Equation 7.2.

$$\text{Faithfulness Score} = \frac{C_f}{N_f} \quad (7.2)$$

The faithfulness score represents the proportion of claims in the response that align with the provided context. A higher faithfulness score indicates that most claims are supported by the context, suggesting that the model has produced a more reliable and accurate response.

The hallucination and faithfulness scores provide complementary evaluations of LLM outputs. The hallucination score detects contradictory information in a response, while the faithfulness score assesses whether the claims made are accurate based on the given context. A low hallucination score combined with a high faithfulness score indicates that the model generates reliable responses with minimal factual inconsistencies.

7.1.2 Weakly Supervised Hallucination Detection

The hallucination and faithfulness scores (scenario 1) described in section 7.1.1 rely on having access to a reference context to be used as a ground truth for verification. While this assumption is true for RAG applications, it is not applicable to LLMs accessed via APIs or those generating responses based entirely on their parametric knowledge, as the ground truth context is not always available for verification.

This subsection describes the methodology for scenario 2, which develops and evaluates classifiers that detect hallucinations in LLM-generated responses by directly classifying the LLM response as hallucinated or not, without relying on reference contexts. Instead of relying on human-curated datasets, this approach uses noisy labels generated by other LLMs to create training labels for hallucination detection.

The hallucination detection for scenario 2 in this chapter is formulated as a binary classification task with two possible classes: $C = \{\text{"Not Hallucination"}, \text{"Hallucination"}\}$. Since the dataset does not contain human curated labels, noisy labels are first generated using k LLMs, denoted as: $\lambda = \{\lambda_1, \dots, \lambda_k\}$. This process follows a structured prompt shown in Figure 7.3. The prompt contains detailed instructions outlining the task for each model, including the original input, the correct (gold) answer, and the LLM-generated response.

Each language model λ_j assigns a noisy label to each input x_i , where $\lambda_j(x) \in \{0, 1\}$. A label of 1 indicates that the input is hallucinated, while 0 indicates it is not. Given the k synthetic labels produced by the k LLMs, a single, less noisy aggregated label is computed, as different LLMs may label the same input x_i differently. This is done using ensemble methods: WAWA, Majority Vote, and Dawid Skene, described in detail in section 4.1.1.1. Unlike the training data, which is labeled using LLMs, the test dataset used for evaluation is labeled by human annotators.

Hallucination Label Generation Prompt

Given the target task (Target), the input passed to the model for generation (Input), the intended reference "gold" text that the model ought to generate (True answer) the actual model production (Generated Answer).

For task, DM means Definition modelling, MT means machine translation and PG means paraphrase generation.

Task: {task}

Input: {source}

True Answer: {target}

Generated Answer: {model_answer}

Your task is to analyze each Generated Answer comparing it with the True Answer to the Input and determine whether it is hallucinated or non-hallucinated. Hallucinated text often includes factual inaccuracies, irrelevant or loosely related content to the given topic, fabrications of specific details, or unnecessary repetition.

Non-hallucinated text, on the other hand, should be coherent, relevant to the given topic, factually accurate, and free from unnecessary repetition.

Choose from the following labels: ['hallucination', 'not hallucination']

Figure 7.3: Hallucination Label Generation Prompt. Values in curly brackets are replaced with actual values at inference.

7.1.3 Data

In this chapter, two datasets were used for the two scenarios of hallucination detection, and the class distributions of their test sets are shown in Table 7.1. Both test sets were labeled by human annotators.

Table 7.1: Class distribution of test datasets labeled by human annotators for the two hallucination detection scenarios: scenario 1 (based on hallucination and faithfulness scores) and scenario 2 (based on weakly supervised learning).

Dataset	Not Hallucination	Hallucination
RAG Responses (Scenario 1)	352	74
SHROOM (Scenario 2)	563	435

In scenario 1, the dataset consists of RAG generated responses from LLMs discussed in chapter 6: LLAMA2-7B, LLAMA2-13B, and Mistral2-7B. These responses were evaluated by a human annotator to determine their accuracy and whether they contained hallucinations or not. The dataset comprises a total of 142 data points for each of the LLM RAG generators, all with human annotations. Among these, 113 responses from

LLAMA2-7B were classified as not hallucinated, while 129 responses from LLAMA2-13B and 110 from Mistral2-7B were also identified as not hallucinated.

In scenario 2, the dataset is the SHROOM [MZV⁺24], developed for hallucination detection, that has questions spanning different NLP tasks: Definition Modeling (DM), Machine Translation (MT) and Paragraph Generation (PG). It includes both development and test sets. The training set consists of 30,000 data points without human annotations. To create labeled training data, we used LLMs to generate synthetic labels for this dataset. In contrast, the test set contains human-annotated labels. It includes 999 instances, of which 563 are classified as non-hallucinated and 435 as hallucinated.

7.1.4 Experimental Setup and Metrics

This section describes the experimental setup for the two scenarios of hallucination detection. For scenario 1 (the hallucination and faithfulness score approach) experiments, LLM judges were selected from SoTA models within the 7B to 8B parameter range available at the time. The LLMs used are: LLAMA3-8B, Mistral3-7B, and Qwen2-7B. These models were not fine-tuned for judging; all outputs were generated during inference using prompting only.

For scenario 2 of hallucination detection (weakly supervised learning), the classifiers were trained using a selection of models, including: Zephyr-7B, Mistral2-7B, BERT, and RoBERTa. These models were chosen to provide a mix of smaller architectures like BERT which provide faster inference while also assessing the effectiveness of large LLMs like Zephyr-7B and Mistral2-7B, which were among the leading models in the 7B category at the time of experimentation. Larger LLMs were not included in the training phase due to hardware constraints.

For inference using prompting (scenario 2), which requires fewer computational resources than fine-tuning, larger models were tested including: LLAMA2-13B, LLAMA2-70B, and GPT-3.5-Turbo. GPT-3.5-Turbo was specifically chosen as a proprietary non-open-source model known for its strong performance and cost-efficient inference. The performance of experiments in both scenarios was assessed using F1 score, accuracy, and GM, as detailed in section 4.1.3.

7.2 Results and Discussion

This section presents the results of the two hallucination detection approaches proposed in this chapter: hallucination and faithfulness scores (scenario 1) and weakly supervised learning (scenario 2).

7.2.1 Hallucination and Faithfulness Scores for Hallucination Detection

This subsection presents the discussion of results for scenario 1. The hallucination score, described in section 7.1.1, measures the proportion of context sentences that contradict the generated response. The score ranges from 0 to 1; responses with a score below 0.5 are predicted as not hallucinated, while those with a score of 0.5 or higher are predicted as hallucinated.

The faithfulness score measures how many claims in the LLM response are supported by the given context. A score above 0.5 means the response is considered faithful and predicted as not hallucinated. For more details on how these scores are calculated, see section 7.1.1. The performance metrics in Table 7.3 are calculated by comparing the predicted class, based on the thresholds defined for the hallucination and faithfulness scores, with the human-annotated labels.

Table 7.3: Performance metrics of different LLM judges for hallucination detection on RAG responses

LLM RAG Generator	LLM Judge	Hallucination Scores			Faithfulness Scores		
		Accuracy	F1	GM	Accuracy	F1	GM
LLAMA2-13B	LLAMA3-8B	67.50	79.03	81.13	59.53	72.47	50.46
	Mistral3-7B	79.29	88.00	83.82	95.48	97.67	42.81
	Qwen2-7B	42.64	56.96	62.95	68.45	74.29	39.62
Mistral2-7B	LLAMA3-8B	63.48	74.76	76.86	79.34	91.77	64.61
	Mistral3-7B	85.54	91.75	87.93	90.57	94.97	36.42
	Qwen2-7B	52.58	64.55	68.54	63.73	73.64	38.26
LLAMA2-7B	LLAMA3-8B	56.91	69.99	73.38	57.79	70.87	36.31
	Mistral3-7B	83.24	90.40	84.62	91.34	95.28	38.51
	Qwen2-7B	21.97	26.54	34.10	62.31	71.88	47.43

Table 7.3 shows the evaluation results for different LLM RAG generators: LLAMA2-13B, LLAMA2-7B, and Mistral2-7B. These models were evaluated by three LLM judges: LLAMA3-8B, Mistral3-7B, and Qwen2-7B. The Hallucination Scores and Faithfulness Scores columns represent two different approaches to hallucination detection, detailed in section 7.1.1.

Using the hallucination score approach on LLAMA2-13B RAG responses, Mistral3-7B achieved the highest GM of 83.83%, followed by LLAMA3-8B with 81.13%, while Qwen2-7B had the lowest with 62.95%. For the faithfulness score approach, LLAMA3-8B achieved the highest GM of 50.46%, followed by Mistral3-7B with 42.81%, and Qwen2-7B again the lowest. Although the differences are smaller for faithfulness, all GM values

are clearly lower compared to those from the hallucination score.

On Mistral2-7B RAG responses, Mistral3-7B again achieved the highest GM on the hallucination score approach at 87.93%, followed by LLAMA3-8B at 76.86%, and Qwen2-7B at 68.54%. For the faithfulness score, LLAMA3-8B had the highest GM at 64.61%, significantly outperforming Mistral3-7B and Qwen2-7B, which scored 36.42% and 38.26% respectively.

On LLAMA2-7B RAG responses, using the hallucination score approach, Mistral3-7B achieved the highest GM of 84.62%, followed by LLAMA3-8B at 73.38%, and Qwen2-7B at 34.10%. In contrast, for the faithfulness score, Qwen2-7B performed best with a GM of 47.43%, ahead of Mistral3-7B and LLAMA3-8B, which scored 38.51% and 36.31%, respectively.

Among all LLM judges, Mistral3-7B consistently achieved the highest GM for the hallucination score criteria across all RAG generators. In contrast, results for the faithfulness score varied, with LLAMA3-8B obtaining the highest GM in two of the three cases. Overall, the GM values for the faithfulness approach were lower compared to those of the hallucination approach. This suggests that the hallucination score is more effective at detecting factual inconsistencies in LLM generated responses compared to the faithfulness score.

This could be due to the differences in how the two scores are computed. In the faithfulness prompt, the model is first required to extract claims from the response and then verify whether those claims are supported by the context. In contrast, the hallucination score follows a more straightforward approach, where each sentence in the context is directly checked for consistency with the response. This makes hallucination detection a relatively simpler task compared to first identifying claims and then verifying their validity.

7.2.2 Weakly Supervised Learning for Hallucination Detection

In this subsection, we present the results for scenario 2, which is based on the weakly supervised hallucination detection approach. Model performance was evaluated according to the adaptation method (prompting or fine-tuning), the type of training data used, and the ensembling strategy applied. The methodology for this evaluation is detailed in section 7.1.2, using the SHROOM dataset [MZV⁺24].

Table 7.4: Comparison of different models for the hallucination detection task under zero-shot learning with prompting

Model	Method	Aggregation	Accuracy	F1
LLAMA2-13B	Prompting	None	56.41	36.07
Zephyr-7B	Prompting	None	67.64	60.55
GPT-3.5-Turbo	Prompting	None	68.34	68.25
LLAMA2-70B	Prompting	None	75.45	75.35

Zero-shot Prompting: Table 7.4 presents a comparison of different LLMs and their performance in detecting hallucinations using prompting in a zero-shot setting. The results show that LLAMA2-70B performed best with an accuracy of 75.45%, followed by GPT-3.5-Turbo at 68.34%, Zephyr-7B at 67.64%, and LLAMA2-13B with the lowest accuracy of 56.41%. These results suggest that within the same model family, larger models tend to achieve better hallucination detection accuracy when using prompting.

An example of this is the 19% accuracy difference between LLAMA2-70B and LLAMA2-13B, which share the same architecture but differ in size (70 billion parameters versus 13 billion parameters). However, model size does not solely determine performance across different model types. For instance, Zephyr-7B, with only 7 billion parameters, outperforms LLAMA2-13B (67.64% versus 56.41%). This may be due to Zephyr-7B's additional tuning through direct preference optimization [TBL⁺23].

Table 7.5: Performance Measures for different Ensemble Techniques

Model	Adaptation Technique	Training Data	Aggregation	Accuracy	F1
Ensemble	Prompting	None	WAWA	73.45	70.94
Ensemble	Prompting	None	Majority Vote	73.45	70.94
Ensemble	Prompting	None	Dawid-Skene	75.65	75.65

Ensemble Techniques: The influence of different ensembling methods on downstream performance was evaluated, with results shown in Table 7.5. Among the methods tested, the Dawid-Skene approach achieved the highest accuracy at 75.65%, while majority vote and WAWA both reached an accuracy of 73.45%.

When comparing ensemble methods to individual model performance, the findings suggest that ensembling leads to improved accuracy. However, for the best performing model, LLAMA2-70B, the improvement was minimal, with accuracy increasing from 75.45% to 75.65%, a slight gain of only 0.2 percentage points.

Table 7.6: Accuracy for different models across different generation tasks: MT, DM and PG

Model	Adaptation Technique	MT	DM	PG	Overall
LLAMA2-70B	Prompting	82.09	74.87	74.87	75.45
GPT-3.5-Turbo	Prompting	75.94	65.51	61.20	68.34
Zephyr-7B	Prompting	82.62	51.34	69.60	67.64
RoBERTa	Fine-tuning	74.33	66.31	62.40	68.34
Mistral2-7B	Fine-tuning	83.96	78.07	74.40	79.17
Zephyr-7B	Fine-tuning	85.03	78.07	73.60	79.36

Task Performance: Table 7.6 provides a comparative analysis of different model performances for detecting hallucinations in the three tasks: DM, MT, and PG. The results highlight variability in performance across these tasks for different models. Models achieve the highest accuracy in the MT task at 85.03%, followed by DM at 78.07%, while the lowest performance is observed in PG at 79.36%. Among the evaluated models, LLAMA2-70B is the best performing model across all tasks when used in a prompting setting.

In the MT task, LLAMA2-70B and Zephyr-7B show similar performance in the prompting setting, though Zephyr-7B experiences a drop in accuracy in the DM task. During fine-tuning, Zephyr-7B and Mistral2-7B perform comparably in DM, while Zephyr-7B outperforms Mistral2-7B in MT (85.03% versus 83.96%). On the other hand, Mistral2-7B performs slightly better than Zephyr-7B in PG (74.40% versus 73.60%).

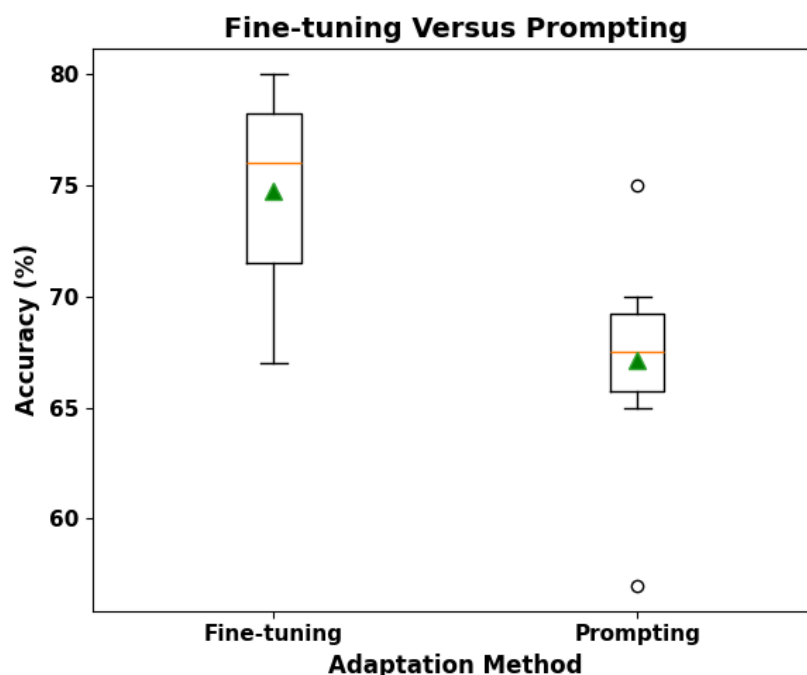


Figure 7.4: Box plots illustrating the accuracy of fine-tuning and prompting adaptation methods.

Prompting versus fine-tuning: Figure 7.4 shows that fine-tuning on weakly labeled data improves model performance. Fine-tuned models achieved a median accuracy of 74.01%, surpassing the median accuracy of 70.98% for models that used only prompting.

From Table 7.6, the fine-tuned versions of Mistral2-7B and Zephyr-7B outperformed models that relied solely on prompting, including LLAMA2-70B and LLAMA2-13B. Zephyr-7B achieved the highest accuracy of 79.66% and an F1 score of 79.37%, which is a 12% increase over its zero-shot prompting accuracy of 67.63%. The fine-tuned Zephyr-7B also outperformed LLAMA2-70B, the top prompting model (79.66% versus 75.45%). These findings show that fine-tuning, even with weakly labeled data, can enhance model performance beyond what is possible with prompting alone.

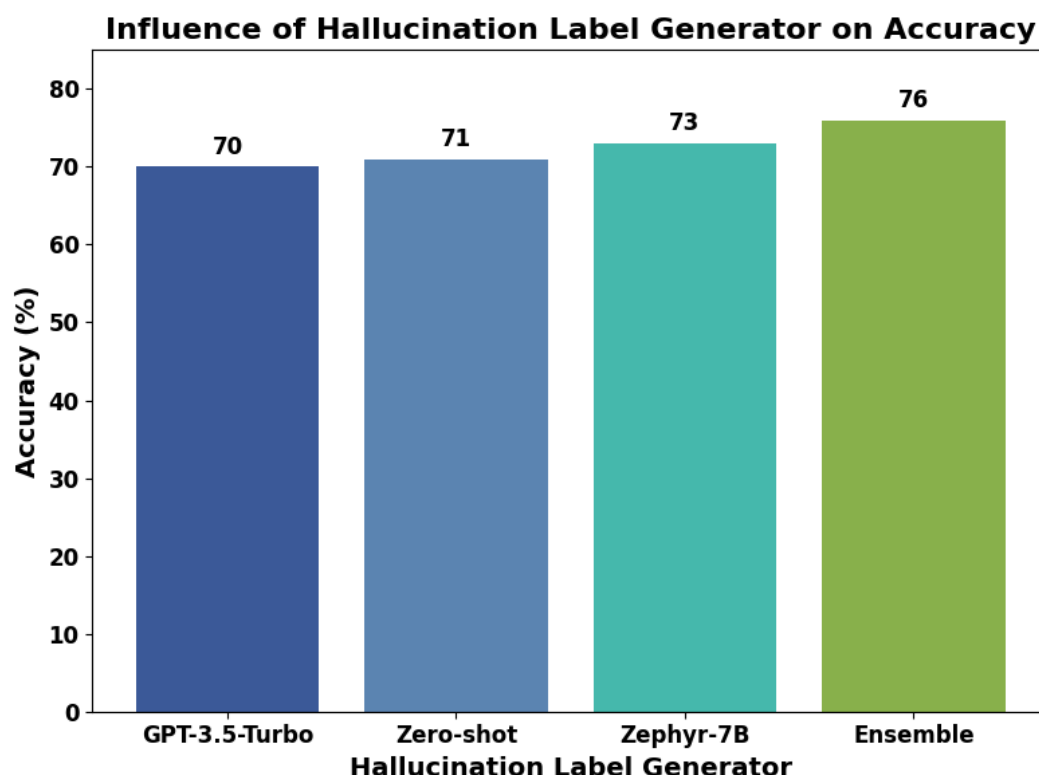


Figure 7.5: Accuracy for Model fine-tuned on Data generated by different Hallucination Label Generators

Hallucination Label Generator: We conducted additional evaluations to examine how different hallucination label generators influence the performance of fine-tuned models. Figure 7.5 shows the accuracy of models fine-tuned on data labeled by various generators. The lowest accuracy was observed with zero-shot prompting alone at 70.98%. Accuracy slightly improved with labels from GPT-3.5-Turbo at 71.04%, followed by Zephyr-7B at 72.64%. The best performance was achieved using labels generated by an ensemble of models, reaching 75.75%. These results suggest that hallucination label generators with higher a zero-shot performance tend to produce training data that leads to better fine-tuned models.

Even when a model generates its own training labels and is then fine-tuned on that data, performance improves. For example, Zephyr-7B reaches 72.84% accuracy in a zero-shot setting, but when fine-tuned on its self-generated labels, the accuracy increases to 76.85%, a gain of 4.01%. A similar trend is seen with Mistral2-7B, which, when trained on labels from GPT-3.5-Turbo, performs better than GPT-3.5-Turbo itself (75.35% versus 68.33%). This improvement is likely due to the fine-tuning process helping the model learn task-specific patterns, rather than relying on any extra information in the training data.

7.3 Conclusion

In this chapter, we analyzed two scenarios for hallucination detection: one designed for scenarios where a source context is available, such as in RAG and LLM agents (scenario 1), and another that identifies hallucinations directly from LLM responses (scenario 2).

In scenario 1, we evaluated two scoring methods: the hallucination score, which checks how many context sentences contradict the generated response, and the faithfulness score, which measures how many claims in the response are supported by the context. The results showed that the hallucination score was more effective at detecting inconsistencies, even though both scores aim to identify the same problem using different methods. Among the LLM judges evaluated, Mistral3-7B achieved the highest GM across all LLM RAG generators using the hallucination score approach. However, in the faithfulness score approach, LLAMA3-8B performed best in two out of the three LLM RAG generator responses.

In the scenario 2, fine-tuning improved performance even when using noisy labels generated by other LLMs. One observation was that when an LLM generated training data and was subsequently fine-tuned on the same data, its performance surpassed that of its original zero-shot prompting. These findings suggest that weak labels from LLMs can serve as a viable alternative for training hallucination classifiers, particularly when human annotations are unavailable or expensive to obtain. However, even after filtering out potential hallucinations, LLM responses must still be verified for accuracy, especially in sensitive applications.

Chapter 8

Conclusion and Future Work

The key contributions of this Thesis lie in the adaptation of LLMs for analyzing digital media and government publication documents. The rapid growth of unstructured data due to digital transformation raised the question of whether NLP advancements such as LLMs, which have largely benefited the private sector, could also be effectively applied in the public sector. Although public sector organizations have considerable resources, they often allocate limited funding to ML training and infrastructure. This Thesis examined how LLMs can be implemented cost efficiently in these domains, the challenges involved in their adoption, and potential solutions.

Three applications in the public sector were studied to address the research questions raised in this Thesis. The first focused on classifying news articles to support the construction of EPU index for monitoring economic trends. The second examined the analysis of citizen feedback in a government organization, assessing the effectiveness of LLMs in classification, summarization, and question answering. The third investigated the use of LLMs for summarizing and responding to queries about extensive government documents, adapting SoTA approaches such as RAG, LLM agents, and multimodal LLMs. The conclusions derived from the experiments conducted in these applications are discussed next.

8.1 Conclusions

Chapter 4 adapts LLMs in the public sector for economic monitoring, using news articles as indicators to measure EPU. This chapter is focused on the identification stage, where automated approaches based on LLMs are proposed for classifying news articles that describe EPU and identifying different EPU categories. The experiments conducted in the chapter aimed to address the second research objective: assessing whether cost effective labeling sources such as synthetic labels, could be used to fine-tune LLMs for

downstream tasks instead of relying on human annotations.

Experimental results showed that while models trained on human labeled data achieved better classification performance, the difference in performance between these models and those fine-tuned with LLM generated labels was relatively small. Given the high cost and time constraints of human annotation, LLM generated data presents a practical and cost-effective alternative for developing EPU classifiers.

Additionally, certain LLMs, particularly those with a large number of parameters, exhibited strong zero-shot performance. However, their high inference costs make them impractical for continuous deployment. A more feasible approach is to utilize these models as label generators, producing training labels that can be used to fine-tune smaller, more efficient models. Furthermore, aggregation techniques enhanced model performance and reduced noise in the generated labels. These findings support the research question by demonstrating that synthetic labels can provide a viable starting point for adapting LLMs in this application.

Chapter 5 investigated automated methods for analyzing unstructured citizen feedback. The approaches studied include unsupervised ML techniques such as topic models, to identify patterns within the data. Additionally, the chapter evaluated ML techniques for automatically classifying textual feedback into different categories, as well as generating summaries and answering questions related to citizen feedback. The experiments conducted in this chapter aimed to address the third research question, which focused on how LLMs can improve the processing and utilization of unstructured citizen feedback in government organizations.

For category citizen feedback classification, the results demonstrated that pre-trained LLMs significantly outperformed traditional ML models, with the latest generative models from the Qwen and LLAMA LLM families achieving the highest performance. The performance gap between fine-tuning and prompting was considerably large for classification tasks. However, for question answering and summarization, while fine-tuning remained the more effective approach, the difference in performance between prompting and fine-tuning was smaller compared to classification. The high performance achieved by these LLMs indicates their effectiveness in classifying citizen feedback and summarizing lengthy responses for better clarity. However, for question answering, RAG or LLM agent-based approaches may be more suitable than fine-tuning, as questions are dynamic, making it challenging to encode all relevant information during fine-tuning.

In chapter 6, various techniques including RAG, LLM agents, and multimodal LLMs, were evaluated for their effectiveness in summarization and question answering of public sector documents. Chapter 7 expands on this work by studying hallucination

detection in LLM outputs using two approaches: one that verifies responses against a reference context, and another that detects hallucinations based only on the LLM responses. The experiments conducted in these chapters aimed to address the research questions: whether LLMs can function as language assistants to process and streamline information retrieval from government documents and websites, and the primary challenges AI professionals encounter when adapting current models to specific applications effectively.

Several datasets were curated through human annotation and synthetic generation to evaluate different LLM capabilities in this domain. Extensive experiments were conducted to develop effective RAG systems, revealing what choices of embedding models, re-rankers, retrievers, and other RAG components significantly influence downstream performance. The integration of RAG substantially enhanced LLM accuracy. Among the multimodal LLMs assessed, a clear performance gap was observed between proprietary models like GPT-4 and Gemini and open-source alternatives such as LLAVA. For tool use selection, LLMs had a high detection accuracy on the choice of tool for a given question, though the experiments conducted in this Thesis were only limited to two tools: vector database and search tool.

Although the methods applied in this chapter reduced hallucination rates and improved LLMs' ability to respond to queries beyond their parametric knowledge, the generated responses are still probabilistic and may contain errors. This remains one of the biggest barriers to adoption of LLMs in these domains. To mitigate potential inaccuracies, implementing output guardrails is essential. However, even after filtering hallucinations, LLM generated responses must be verified for accuracy, particularly in sensitive applications where misinformation could have significant consequences.

8.2 Future Work

Chapter 6 evaluated multimodal LLMs for processing documents, particularly statistical reports that combine visual and textual elements. However, this Thesis focused on processing simple one-page or short multi-page reports without incorporating multimodal information during retrieval. Future research could explore integrating visual components into RAG systems, particularly at the retrieval stage, to enhance the selection of relevant document chunks by considering both visual and textual content.

Additionally, in this Thesis, LLM agents were assessed based on their ability to select appropriate tools. As their adoption grows, future research could extend beyond tool selection to analyze the agent's reasoning trajectory, evaluating whether the actions taken were appropriate and whether the retrieved information was accurate. This could

improve system reliability and facilitate debugging for more effective deployment.

In chapter 7, hallucination detection methods were evaluated, primarily leveraging prompt-based techniques or fine-tuning with either human annotations or synthetic labels. However, implementing these approaches in deployed applications can be costly and time intensive, as they require additional LLM calls, increasing latency. Future research could focus on detecting hallucinations directly from LLM outputs without relying on external classifiers. This could involve analyzing token probabilities or internal model states linked to hallucinations. Examining statistical differences in token distribution between hallucinated and factual statements may offer insights into developing automatic hallucination detection methods that directly utilize these distributional variations.

The models evaluated in this Thesis primarily consisted of traditional ML techniques, pre-trained LLMs, and models that had undergone instruction tuning or preference alignment. Towards the final stages of this Thesis, a new generation of reasoning-focused models such as OpenAI's o1 and Deepseek-R1 [GYZ⁺25] emerged. These models integrate reinforcement learning techniques to enhance logical reasoning by incorporating CoT processes even during pre-training, resulting in improved reasoning abilities and overall performance compared to many models examined in this Thesis. Future research could explore whether substituting the LLMs used in this Thesis with these newer models yields substantial performance gains in the applications studied.

Acronyms

AI Artificial Intelligence. xi, 2, 7, 34, 36, 37, 44, 45, 48, 49

ALBERT A Lite BERT. 45

API Application Programming Interface. 28, 36, 46, 95, 115, 120

AUC Area Under the Curve. vi, 60, 63–66, 78, 85

AUPRC Area Under the Precision Recall Curve. vi, 60, 63–66, 78, 85

AutoML Automated Machine Learning. 44

BART Bidirectional and Auto-Regressive Transformer. 8, 29

BERT Bidirectional Encoder Representations from Transformers. 8, 26, 27, 29, 41, 45, 46, 48, 55, 56, 59, 62–66, 71, 77, 78, 85, 86, 89, 122

BERTScore Bidirectional Encoder Representations from Transformers Score. vi, 78, 88, 89

BLEU Bilingual Evaluation Understudy. vi, 100, 105, 106

BLOOM BigScience Large Open-science Open-access Multilingual Language Model. 8, 28, 29

BM25 Best Matching 25. 35

BoW Bag of words. 8, 12, 41

C4 Colossal Clean Crawled Corpus. 29

CBOW Continuous Bag of Words. 13, 14

CE Cross Entropy. 56–58, 73, 85, 86

ChatGPT Chat Generative Pre-trained Transformer. 2, 34, 43, 45, 48, 115, 117

CK Cohen’s Kappa. vi, 60, 61, 114, 115

CLIP Contrastive Language Image Pretraining. 36, 37

- CNN** Convolution Neural Network. 41, 44
- ColBERT** Contextualized Late Interaction over BERT. 94
- CoT** Chain of Thought. 8, 31, 36, 133
- CSO** Central Statistics Office. 99, 104, 105, 113
- DistilBERT** Distilled BERT. 45, 77, 85
- DistilRoBERTa** Distilled RoBERTa. 77, 85, 86
- DM** Definition Modeling. vii, 121, 122, 126
- Doc2Vec** Document to Vector. 8, 14
- DP** Data Programming. iv, vi, 53, 54, 61–64, 67
- DPO** Direct Preference Optimization. 34
- ELECTRA** Efficiently Learning an Encoder that Classifies Token Replacements Accurately. 8, 27
- EMR** Exact Match Ratio. 60, 64–66
- EPU** Economic Policy Uncertainty. iv, vi, xi, 2, 3, 6, 38–41, 51–53, 55–67, 130, 131
- FAISS** Facebook AI Similarity Search. 93
- FLAN** Fine-tuned LAnguage Net. 32
- GBM** Gradient Boosting Machines. 16, 17, 40, 43, 77, 85, 86, 89
- GloVe** Global Vectors for Word Representation. 14
- GM** Geometric Mean. iv, vi, 60, 62–66, 78, 85–87, 122–124, 129
- GPT** Generative Pre-trained Transformer. 6–8, 28–30, 33, 34, 37, 41, 45, 46, 48, 115, 116
- GPT-4** GPT version four. 132
- GPT-40-mini** GPT version four mini. 112
- GPU** Graphics Processing Unit. 24, 59, 77, 78
- GRU** Gated Recurrent Unit. 8, 19, 40, 41, 44
- HSE** Zephyr model with 7 billion parameters. vi, 92, 98, 99, 104–106, 114, 115

- IDF** Inverse Document Frequency. 12
- InstructGPT** Instruction Tuned Generative Pretrained Transformer. 8, 33, 34
- IRISH** Documents from Irish publications website. vi, 92, 98, 99, 102, 104–106, 114, 115
- IT** Information Technology. 45
- KNN** K-Nearest Neighbors. 43, 44
- LDA** Latent Dirichlet Allocation. 40, 43, 46, 70, 71
- LLAMA** Large Language Model Meta AI. 8, 32, 34, 37, 41, 46, 47, 59, 77, 89, 103
- LLAMA2-13B** LLAMA version two with 13 billion parameters. 103, 105, 106, 115, 121–123, 125, 127
- LLAMA2-70B** LLAMA version two with 70 billion parameters. 122, 125–127
- LLAMA2-7B** LLAMA version two with 7 billion parameters. 77, 88, 89, 103, 105, 106, 121–124
- LLAMA3-1B** LLAMA version three with 1 billion parameters. 77, 85
- LLAMA3-3B** LLAMA version three with 3 billion parameters. 77, 85
- LLAMA3-70B** LLAMA version three with 70 billion parameters. 103, 112
- LLAMA3-8B** LLAMA version three with 8 billion parameters. 60–62, 103–106, 112, 114, 115, 122–124, 129
- LLAVA-13B** LLAVA with 13 billion parameters. 103, 113, 114
- LLAVA-7B** LLAVA with 7 billion parameters. 103, 113, 114
- LLM** Large Language Models. iv, vi, vii, xi, 2–8, 14, 22, 24, 26–28, 30–56, 59, 61–63, 65, 67, 68, 73, 75–77, 85, 88–92, 94–108, 111–125, 129–133
- LM** Language Model. 9
- LRP** Label Ranking Precision. 60, 64–66
- LSTM** Long Short Term Memory. 8, 19, 40, 44, 55, 56, 59, 62–66
- Majority Vote** Majority Vote. 53–55, 61, 62
- MaxSim** Maximum Cosine Similarity. 94
- MCC** Matthews Correlation Coefficient. vi, 60, 61, 114, 115

- Mistral-24B** Mistral 24 Billion parameters. 60–63, 67
- Mistral2-7B** Mistral version two with 7 billion parameters. 77, 88, 89, 103–106, 114, 121–124, 126–128
- Mistral2-8X7B** Mistral2-8X7B. 103, 105, 106, 114
- Mistral3-7B** Mistral version two with 7 billion parameters. 60–62, 103–106, 112, 114, 115, 122–124, 129
- ML** Machine Learning. xi, 3, 7, 8, 10, 15, 17, 30, 36–46, 51, 67, 68, 72, 73, 77, 84–86, 89, 130, 131, 133
- MLM** Masked Language Modeling. 26, 27
- MT** Machine Translation. vii, 122, 126
- NLP** Natural Language Processing. xi, 1, 2, 5, 7–10, 13–15, 17, 18, 20, 22, 26, 29, 32, 37, 38, 42, 44, 45, 47, 49, 68, 122, 130
- NSP** Next Sentence Prediction. 27
- NTP** Next Token Prediction. 26, 28, 31, 34, 76, 91
- OECD** Organisation for Economic Co-operation and Development. 48
- OPT** Open Pretrained Transformer. 8, 28, 29
- OPT-125M** OPT with 125 million parameters. 77, 85
- OPT-350M** OPT with 350 million parameters. 77, 85
- PaLM** Pathways Language Model. 32
- PG** Paragraph Generation. vii, 122, 126
- QA** Question Answering. 47
- Qlora** Quantized Low-Rank Adaptation. 78
- Qwen2-0.5B** Qwen version two with 0.5 billion parameters. 77, 85
- Qwen2-14B** Qwen version two with 14 billion parameters. 60–62
- Qwen2-3B** Qwen version two with 3 billion parameters. 77, 85
- Qwen2-7B** Qwen version two with 7 billion parameters. 60–62, 77, 85–87, 122–124

- RAG** Retrieval Augmented Generation. iv–vii, xi, 3, 4, 6–8, 35, 36, 38, 45, 47–50, 90–92, 94, 95, 98, 99, 103–105, 107–111, 113, 115, 117, 118, 120, 121, 123, 124, 129–132
- RBF** Radial Basis Function. 15
- ReACT** Reasoning + Acting. 8, 36
- ReLU** Rectified Linear Unit. 18, 19
- RLHF** Reinforcement Learning from Human Feedback. 7, 33, 34
- RNN** Recurrent Neural Network. iv, 7, 8, 17–22, 25, 44, 48
- RoBERTa** Robustly Optimized BERT Pretraining Approach. 8, 27, 29, 41, 45, 46, 48, 55, 56, 59, 62–66, 77, 85, 89, 122, 126
- ROC** Receiver Operating Curves. iv, 64
- ROOTS** Responsible Open-science Open-access Text Sources. 29
- ROUGE** Recall Oriented Understudy for Gisting Evaluation. vi, 78, 88, 89, 100, 105, 106
- SBERT** Sentence Bidirectional Encoder Representations from Transformers. 8
- SHROOM** Shared-task on Hallucinations and Related Observable Overgeneration Mistakes. 118, 121, 122, 124
- SoTA** State of The Art. 6, 28, 29, 32, 37, 59, 77, 93, 103, 114, 115, 122, 130
- STM** Structural Topic Model. 46, 70, 72, 81
- SVM** Support Vector Machines. 8, 15, 40, 43–45, 77, 85, 86, 89, 93, 110
- T5** Text-to-Text Transfer Transformer. 8, 29, 32
- TF** Term Frequency. 12, 71, 72
- TF-IDF** Term Frequency Inverse Document Frequency. 8, 12, 44, 71
- TIN** Tax Identification Number. 79, 82, 83
- ToT** Tree of Thought. 8, 31, 36
- URA** Uganda Revenue Authority. 3, 68, 69, 78
- USA** United States of America. 58
- VAT** Value Added Tax. 81, 83

WAWA Worker Agreement with Aggregate. 53, 55, 61, 62, 120, 125

Word2Vec Word to Vector. 8, 12–14, 44

WS weak supervision. iv, vi, 62–64

XLNet Extra Long Network. 8, 27

Zephyr-7B Zephyr model with 7 billion parameters. 41, 88, 89, 122, 125–128

References

- [AAA⁺23] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. The Falcon Series of Open Language Models. *ArXiv Preprint arXiv:2311.16867*, 2023.
- [AADH⁺22] Marah Alhalabi, Bashaver Al Dhaghari, Nada Hussein, Hurma Ehtesham, Janette Laconsay, Adel Khelifi, Ahmed Salem, and Mohammed Ghazal. M-Government Smart Service Using AI Chatbots: Evidence from the UAE. In *2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference*, pages 325–330. IEEE, 2022.
- [ADL⁺22] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: A Visual Language Model for Few-shot Learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 23716–23736. Curran Associates, Inc., 2022.
- [AG17] Andrés Azqueta-Gavaldón. Developing News-Based Economic Policy Uncertainty Index with Unsupervised Machine Learning. *Economics Letters*, 158:47–50, 2017.
- [AGDGP23] Andrés Azqueta-Gavaldón, Marina Diakonova, Corinna Ghirelli, and Javier J Pérez. Sources of Economic Policy Uncertainty in the Euro Area: A Ready-to-use Database. Occasional paper, Bank of Spain, August 2023.
- [AGHOS20] Andres Azqueta-Gavaldon, Dominik Hirschbühl, Luca Onorante, and Lorena Saiz. Economic Policy Uncertainty in the Euro Area: an Unsupervised Machine Learning Approach. Working Paper Series 2359, European Central Bank, 2020.
- [AGS24] Francesco Audrino, Jessica Gentner, and Simon Stalder. Quantifying Uncertainty: A New Era of Measurement through Large Language Models.

- Technical Report 24-68, Swiss Finance Institute, 2024.
- [Aiz03] Akiko Aizawa. An Information Theoretic Perspective of TF-IDF Measures. *Information Processing & Management*, 39:45–65, 2003.
- [AM19] Omar Saeed Al-Mushayt. Automating E-Government Services with Artificial Intelligence. *IEEE Access*, 7:146821–146829, 2019.
- [AMGCM20] Ana Alvarenga, Florinda Matos, Radu Godina, and João CO Matias. Digital Transformation and Knowledge Management in the Public Sector. *Sustainability*, 12:2071–1050, 2020.
- [Ant24] Anthropic. The Claude3 Model Family: Opus, Sonnet, Haiku. Technical report, Anthropic, 2024. Claude-3 Model Card.
- [APG⁺23] Nicola Arici, Luca Putelli, Alfonso Emilio Gerevini, Luca Sigalini, Ivan Serina, et al. LLM-based Approaches for Automatic Ticket Assignment: A Real-world Italian Application. In *Proceedings of the Seventh Workshop on Natural Language for Artificial Intelligence*, volume 3535, pages 1–20, Rome, Italy, 2023. CEUR-WS.org.
- [ASN⁺23] Muhammad Cendekia Airlangga, Andi Sulasikin, Yudhistira Nugraha, Nur Laily Romadhotul Husna, Muhamad Erza Aminanto, Fajar Kurniawan, and Juan Intan Kanggrawan. Understanding Citizen Feedback of Jakarta Government Super App: Leveraging Deep Learning Models. In *2023 IEEE International Smart Cities Conference*, pages 1–6. IEEE, 2023.
- [AT14] Mucahit Altintas and A Cuneyd Tantug. Machine Learning-based Ticket Classification in Issue Tracking Systems. In *Proceedings of the International Conference on Artificial Intelligence and Computer Science*, pages 195–207. World Conferences, 2014.
- [ATJA23] Gerald Agaba, Quynh Tieu, Maria Jouste, and Brian Arinaitwe. The Uganda Revenue Authority Pay-As-You-Earn (PAYE) Data. Technical report, UNU-WIDER Helsinki, 2023.
- [AZFSK22] Syed S Ali Zaidi, Muhammad Moazam Fraz, Muhammad Shahzad, and Sharifullah Khan. A Multi-approach Generalized Framework for Automated Solution Suggestion of Support Tickets. *International Journal of Intelligent Systems*, 37:3654–3681, 2022.
- [BAD⁺12] Ross C Brownson, Peg Allen, Kathleen Duggan, Katherine A Stamatakis, and Paul C Erwin. Fostering More Effective Public Health by Identifying Administrative Evidence-based Practices: A Review of the Literature. *American Journal of Preventive Medicine*, 43:309–319, 2012.

- [BBD16] Scott R. Baker, Nicholas Bloom, and Steven J. Davis. Measuring Economic Policy Uncertainty. *The Quarterly Journal of Economics*, 131:1593–1636, 2016.
- [BCB15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations*, pages 1–15. ICLR, 2015.
- [BGJM17] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.
- [BHA⁺21] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the Opportunities and Risks of Foundation Models. *ArXiv Preprint arXiv:2108.07258*, 2021.
- [Bis06] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer Normalization. *ArXiv Preprint arXiv:1607.06450*, 2016.
- [BKYSG21] Vinayshekhar Bannihatti Kumar, Mohan Yarramsetty, Sharon Sun, and Anukul Goel. SupportNet: Neural Networks for Summary Generation and Key Segment Extraction from Technical Support Tickets. In *Proceedings of the 4th Workshop on e-Commerce and NLP*, pages 164–173, Online, 2021. Association for Computational Linguistics.
- [BM19] Anat Biletzki and Anat Matar. Ludwig Wittgenstein. <https://plato.stanford.edu/entries/wittgenstein/>, 2019. Edited by Edward N. Zalta, Stanford Encyclopedia of Philosophy.
- [BMR⁺20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language Models are Few-shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.
- [BNJ03] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.
- [BPC⁺24] Simone Bonechi, Giulia Palma, Mario Caronna, Massimiliano Ugolini, Alessandra Massaro, and Antonio Rizzo. Enhancing Customer Support

- in Banking: Leveraging AI for Efficient Ticket Classification. *Procedia Computer Science*, 246:128–137, 2024.
- [Bre01] Leo Breiman. Random Forests. *Machine Learning*, 45:5–32, 2001.
- [BSA⁺23] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O’Brien, Eric Hallahan, Mohammad Aflah Khan, Shivan-shu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR, 2023.
- [BZK⁺20] Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. The Pushshift Reddit Dataset. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 14, pages 830–839. Association for the Advancement of Artificial Intelligence, 2020.
- [CGH24] Tzu hao Chen and Mila Gasco-Hernandez. Uncovering the Results of AI Chatbot Use in the Public Sector: Evidence from US State Governments. *Public Performance & Management Review*, 0:1–26, 2024.
- [CGTB24] Emilien Caudron, Nathan Ghesquière, Wouter Travers, and Alexandra Balahur. Adaptation of Large Language Models for the Public Sector: A Clustering Use Case. In *International Conference on Applications of Natural Language to Information Systems*, pages 327–340. Springer, 2024.
- [CHL⁺24] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling Instruction-Finetuned Language Models. *Journal of Machine Learning Research*, 25:1–53, 2024.
- [CHLS24] Daniel CHANG, Nan HU, Peng LIANG, and Morgan SWINK. Understanding the Impact of Trade Policy Effect Uncertainty on Firm-Level Innovation Investment: A Deep Learning Approach. *Journal of Operations Management*, 70:316–340, 2024.
- [CHM⁺23] Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free Dolly: Introducing the World’s First Truly Open Instruction-Tuned LLM. <https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm>, 2023. Accessed: 2023-06-30.

- [CHY24] Chung-Chi Chen, Yu-Lieh Huang, and Fang Yang. Semantics Matter: An Empirical Study on Economic Policy Uncertainty Index. *International Review of Economics & Finance*, 89:1286–1302, 2024.
- [CJ23] Davide Chicco and Giuseppe Jurman. The Matthews Correlation Coefficient (MCC) Should Replace the ROC AUC as the Standard Metric for Assessing Binary Classification. *Bio Data Mining*, 16:4, 2023.
- [CLLM20] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *International Conference on Learning Representations*, pages 1–18. PMLR, 2020.
- [CND⁺23] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. PaLM: Scaling Language Modeling with Pathways. *Journal of Machine Learning Research*, 24:1–113, 2023.
- [CV95] Corinna Cortes and Vladimir Vapnik. Support Vector Networks. *Machine Learning*, 20:273–297, 1995.
- [Dai24] Ziqing Dai. Applications and Challenges of Large Language Models in Smart Government from Technological Advances to Regulated Applications. In *Proceedings of the 2024 3rd International Conference on Frontiers of Artificial Intelligence and Machine Learning*, pages 275–280, New York, NY, USA, 2024. Association for Computing Machinery.
- [DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.
- [DLD⁺24] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, and Zhi-fang Sui. A Survey on In-Context Learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128, Miami, Florida, USA, 2024. Association for Computational Linguistics.

- [DMG23] Zolzaya Dashdorj, Tsetsentsengel Munkhbayar, and Stanislav Grigorev. Deep Learning Model for Mongolian Citizens Feedback Analysis Using Word Vector Embeddings. *ArXiv Preprint arXiv:2302.12069*, 2023.
- [DPHZ23] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs. In *Advances in Neural Information Processing Systems*, volume 36, pages 10088–10115. Curran Associates, Inc., 2023.
- [dPPdGC⁺24] Eduardo Soares de Paiva, Fernando Sola Pereira, David da Guia Carvalho, Nilson Romero Michiles Junior, Rennis Sousa de Oliveira, Stella Mendes Meireles Bonifacio, Andre Luiz Monteiro da Rocha, Hamilton Luiz Rodrigues de Oliveira, Felipe de Abreu Moreira Cezar, and Helio Theodoro Junior. Continued Pre-training of LLMs for Portuguese and Government Domain: A Proposal for Product Identification in Textual Purchase Descriptions. In *AAAI-2024 Workshop on Public Sector LLMs: Algorithmic and Sociotechnical Design*, pages 1–7. AAAI, 2024.
- [DS79] Alexander Philip Dawid and Allan M Skene. Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm. *Journal of the Royal Statistical Society: Applied Statistics*, 28:20–28, 1979.
- [DSB21] Li Dong, Matthew C Spencer, and Amir Biagi. A Semi-Supervised Multi-Task Learning Approach to Classify Customer Contact Intents. *ArXiv Preprint arXiv:2106.07381*, 2021.
- [DVCA23] Huijue Kelly Duan, Miklos A Vasarhelyi, Mauricio Codesso, and Zamil Alzamil. Enhancing the Government Accounting Information Systems Using Social Media Information: An Application of Text Mining and Machine Learning. *International Journal of Accounting Information Systems*, 48:100600, 2023.
- [Eng24] Philipp Engelbrechtsmüller. Fine-tuning Large Language Models for Ticket Classification at Doka. Master’s thesis, Johannes Kepler University Linz, 2024.
- [FB16] Ana Catarina Forte and Pavel B Brazdil. Determining the Level of Clients’ Dissatisfaction from their Commentaries. In *Computational Processing of the Portuguese Language: 12th International Conference*, pages 74–85. Springer, 2016.
- [FBH23] Asbjørn Følstad and Nina Bjerkreim-Hanssen. User Interactions With a Municipality Chatbot—Lessons Learnt From Dialogue Analysis. *International Journal of Human–Computer Interaction*, 40:1–14, 2023.

- [FBR⁺19] Joao Filgueiras, Luís Barbosa, Gil Rocha, Henrique Lopes Cardoso, Luís Paulo Reis, Joao Pedro Machado, and Ana Maria Oliveira. Complaint Analysis and Classification for Economic and Food Safety. In *Proceedings of the Second Workshop on Economics and Natural Language Processing*, pages 51–60. Association for Computational Linguistics, 2019.
- [Fel14] Daniel L Feldman. Commentary: Public Value Governance or Real Democracy. *Public Administration Review*, 74:504–505, 2014.
- [Fir57] J. Firth. A Synopsis of Linguistic Theory 1930-1955. In *Studies in Linguistic Analysis*. Philological Society, Oxford, 1957.
- [FK14] Ahmad Fauzan and Masayu Leylia Khodra. Automatic Multi-label Categorization Using Learning to Rank Framework for Complaint Text on Bandung Government. In *2014 International Conference of Advanced Informatics: Concept, Theory and Application*, pages 28–33. IEEE, 2014.
- [Fri01] Jerome H Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, 29:1189–1232, 2001.
- [FSL22] Leon Feng, Jnana Senapati, and Bill Liu. TaDaa: Real Time Ticket Assignment Deep Learning Auto Advisor for Customer Support, Help Desk, and Issue Ticketing Systems. *ArXiv Preprint arXiv:2207.11187*, 2022.
- [FSLRJ24] Xinyu Fu, Thomas W Sanchez, Chaosu Li, and Juliana Reu Junqueira. Deciphering Public Voices in the Digital Era: Benchmarking ChatGPT for Analyzing Citizen Feedback in Hamilton, New Zealand. *Journal of the American Planning Association*, pages 1–14, 2024.
- [GBB⁺20] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. *ArXiv Preprint ArXiv:2101.00027*, 2020.
- [GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [GKT19] Matthew Gentzkow, Bryan Kelly, and Matt Taddy. Text as Data. *Journal of Economic Literature*, 57:535–74, 2019.
- [Gro22] Maarten Grootendorst. BERTopic: Neural Topic Modeling with a Class-Based TF-IDF Procedure. *ArXiv Preprint arXiv:2203.05794*, 2022.

- [GS13] Justin Grimmer and Brandon M. Stewart. Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts. *Political Analysis*, 21:267–297, 2013.
- [GTDS⁺24] Joseph Gesnouin, Yannis Tannier, Christophe Gomes Da Silva, Hatim Tapory, Camille Brier, Hugo Simon, Raphael Rozenberg, Hermann Woehrel, Mehdi El Yakaabi, Thomas Binder, et al. LLAmandement: Large Language Models for Summarization of French Legislative Proposals. *ArXiv Preprint arXiv:2401.16182*, 2024.
- [GVPP22] Phani Krishna Kollapur Gandla, Rajesh Kumar Verma, Chhabi Rani Panigrahi, and Bibudhendu Pati. Ticket Classification Using Machine Learning. In *International Conference on Advanced Computing and Intelligent Engineering*, pages 487–501. Springer, 2022.
- [GYZ⁺25] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *ArXiv Preprint arXiv:2501.12948*, 2025.
- [Har54] Zellig S. Harris. Distributional Structure. *Word*, 10:146–162, 1954.
- [Har12] Jamie Harding. Choice and Information in the Public Sector: A Higher Education Case Study. *Social Policy and Society*, 11:171–182, 2012.
- [HF03] Louis M Hsu and Ronald Field. Interrater Agreement Measures: Comments on Kappan, Cohen’s κ , Scott’s π , and Aickin’s α . *Understanding Statistics*, 2:205–219, 2003.
- [HHI21] Noor Aklima Harun, Sharin Hazlin Huspi, and Noorminshah A Iahad. Question Classification Framework for Helpdesk Ticketing Support System Using Machine Learning. In *2021 7th International Conference on Research and Innovation in Information Systems*, pages 1–7. IEEE, 2021.
- [HkLX⁺24] Jiawei Han, Jian kang Lu, Ying Xu, Jin You, and Bingxin Wu. Intelligent Practices of Large Language Models in Digital Government Services. *IEEE Access*, 12:8633–8640, 2024.
- [HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9:1735–1780, 1997.
- [Ind20] Agustín Indaco. From Twitter to GDP: Estimating Economic Activity from Social Media. *Regional Science and Urban Economics*, 85:1–32, 2020.

- [ISO25] Arata Ito, Masahiro Sato, and Rui Ota. Content-based Metric on Monetary Policy Uncertainty by using Large Language Models. *Finance Research Letters*, 75:106832, 2025.
- [JFS23] Jinhang Jiang, Mei Feng, and Karthik Srinivasan. Can Conversations on Reddit Forecast Future Economic Uncertainty? An Interpretable Machine Learning Approach. In *Proceedings of the 44th International Conference on Information Systems (ICIS 2023)*, pages 1975–1990, Hyderabad, India, 2023. Association for Information Systems.
- [JK19] Justin M Johnson and Taghi M Khoshgoftaar. Survey on Deep Learning with Class Imbalance. *Journal of Big Data*, 6:1–54, 2019.
- [JM09] Daniel Jurafsky and James H. Martin. *Language Processing: An Introduction to Natural Language Processing*. Prentice Hall, 2009.
- [JM23] Zhijing Jin and Rada Mihalcea. *Natural Language Processing for Policymaking*, pages 141–162. Springer International Publishing, Cham, 2023.
- [Joa98] Thorsten Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *European Conference on Machine Learning*, pages 137–142. Springer, 1998.
- [JPV⁺21] Zhijing Jin, Zeyu Peng, Tejas Vaidhya, Bernhard Schoelkopf, and Rada Mihalcea. Mining the Cause of Political Decision-Making from Social Media: A Case Study of COVID-19 Policies Across the US States. In *Findings of the Association for Computational Linguistics*, pages 288–301. Association for Computational Linguistics, 2021.
- [JPWK23] Yunqing Jiang, Patrick Cheong-Iao Pang, Dennis Wong, and Ho Yin Kan. Natural Language Processing Adoption in Governments and Future Research Directions: A Systematic Review. *Applied Sciences*, 13:12346, 2023.
- [JSM⁺23] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7B. *ArXiv Preprint arXiv:2310.06825*, 2023.
- [JZS15] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, page 2342–2350. JMLR.org, 2015.

- [KB24] Fernando Kleiman and Marcelo Mendes Barbosa. Management and Performance Program Chatbot: A Use Case of Large Language Model in the Federal Public Sector in Brazil. *Digital Government: Research and Practice*, 5:1–8, 2024.
- [KDJ⁺24] Filip Koukal, František Dařena, Roman Ježdík, Jan Přichystal, et al. Improving Automated Categorization of Customer Requests with Recent Advances in Natural Language Processing. *European Journal of Business Science and Technology*, 10:173–184, 2024.
- [KEJM20] Radosław Kowalski, Marc Esteve, and Slava Jankin Mikhaylov. Improving Public Services by Mining Citizen Feedback: An Application of Natural Language Processing. *Public Administration*, 98:1011–1026, 2020.
- [Ket17] Nikhil Ketkar. Introduction to Keras. In *Deep Learning with Python: A Hands-on Introduction*, pages 97–111. Springer, 2017.
- [KFFB24] Babymol Kurian, A Aafreen Fathima, T Afra Fathima, and R Shahista Begum. GovInfohub: A Dynamic Government Scheme Chatbot for Informed Engagement and Accessibility. In *2024 International Conference on Advances in Computing, Communication and Applied Informatics*, pages 1–6. IEEE, 2024.
- [Kow20] Radosław Kowalski. *Automated Analysis of Citizen Feedback for Inclusive Public Policies*. PhD thesis, UCL (University College London), 2020.
- [KS16] Hyo-Soon Kong and Eun-Jee Song. A Study on Customer Feedback of Tourism Service Using Social Big Data. *International Information Institute (Tokyo). Information*, 19:49, 2016.
- [KS19] Katherine Keith and Amanda Stent. Modeling Financial Analysts’ Decision Making via the Pragmatics and Semantics of Earnings Calls. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 493–503, Florence, Italy, 2019. Association for Computational Linguistics.
- [KTOM20] Katherine A Keith, Christoph Teichmann, Brendan O’Connor, and Edgar Meij. Uncertainty over Uncertainty: Investigating the Assumptions, Annotations, and Text Measurements of Economic Policy Uncertainty. In *Proceedings of the Fourth Workshop on Natural Language Processing and Computational Social Science*, pages 116–131. Association for Computational Linguistics, November 2020.

- [KYP⁺21] Byungjun Kim, Minjoo Yoo, Keon Chul Park, Kyeo Re Lee, and Jang Hyun Kim. A Value of Civic Voices for Smart City: A Big Data Analysis of Civic Queries Posed by Seoul Citizens. *Cities*, 108:102941, 2021.
- [KYZ21] Fatemeh Kaveh-Yazdy and Sajjad Zarifzadeh. Measuring Economic Policy Uncertainty Using an Unsupervised Word Embedding-Based Method. *ArXiv Preprint arXiv:2105.04631*, 2021.
- [KZYY18] Feyzullah Kalyoncu, Engin Zeydan, Ibrahim Onuralp Yigit, and Ahmet Yildirim. A Customer Complaint Analysis Tool for Mobile Network Operators. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 609–612. IEEE, 2018.
- [LBJ23] Zhexiong Liu, Cris Benge, and Siduo Jiang. Ticket-BERT: Labeling Incident Management Tickets with Language Models. *ArXiv Preprint arXiv:2307.00108*, 2023.
- [LHV⁺23] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, et al. The FLAN Collection: Designing Data and Methods for Effective Instruction Tuning. *ArXiv Preprint ArXiv:2301.13688*, 2023.
- [Lin04] Chin-Yew Lin. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, jul 2004. Association for Computational Linguistics.
- [Lin23] Zeteng Lin. How Generative AI can be Effectively used in Government Chatbots. *ArXiv Preprint arXiv:2312.02181*, 2023.
- [LLG⁺20] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising Sequence-to-Sequence Pre-Training For Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, 2020. Association for Computational Linguistics.
- [LM14] Quoc Le and Tomas Mikolov. Distributed Representations of Sentences and Documents. In *International Conference on Machine Learning*, pages 1188–1196. PMLR, 2014.
- [LOG⁺19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov.

- RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv Preprint arXiv:1907.11692*, 2019.
- [LSC⁺22] Huayang Li, Yixuan Su, Deng Cai, Yan Wang, and Lemao Liu. A Survey on Retrieval-Augmented Text Generation. *ArXiv Preprint arXiv:2202.01110*, 2022.
- [LSL22] Ivana Lolić, Petar Sorić, and Marija Logarušić. Economic Policy Uncertainty Index Meets Ensemble Learning. *Computational Economics*, 60:401–437, 2022.
- [LYF⁺23] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Computing. Surveys.*, 55:35, 2023.
- [MBMSC23] Hairo U Miranda-Belmonte, Victor Muñiz-Sánchez, and Francisco Corona. Word Embeddings for Topic Modeling: An Application to the Estimation of the Economic Policy Uncertainty Index. *Expert Systems with Applications*, 211:118499, 2023.
- [MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the International Conference on Learning Representations*, pages 1–12. ICLR, 2013.
- [MGH⁺23] Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query Rewriting in Retrieval-Augmented Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Singapore, December 2023. Association for Computational Linguistics.
- [MH12] Mohammed A Mahmoud and Robert E Hinson. Market Orientation in a Developing Economy Public Institution: Revisiting the Kohli and Jaworski’s Framework. *International Journal of Public Sector Management*, 25:88–102, 2012.
- [Mik12] Tomáš Mikolov. Statistical Language Models Based on Neural Networks. Presentation at Google, Mountain View, 2012. Presented on 2nd April.
- [MKBH21] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task Generalization via Natural Language Crowdsourcing Instructions. *ArXiv Preprint arXiv:2104.08773*, 2021.

- [MKK⁺23] Marios Evangelos Mamalis, Evangelos Kalampokis, Areti Karamanou, Petros Brimos, and Konstantinos Tarabanis. Can Large Language Models Revolutionize Open Government Data Portals? A Case of Using ChatGPT in Statistics. gov. scot. In *Proceedings of the 27th Pan-Hellenic Conference on Progress in Computing and Informatics*, pages 53–59. Association for Computing Machinery, 2023.
- [MLL⁺21] Jie Meng, Yan Li, Chen Liu, Yaru Dong, Zishuo Wang, and Yao Zhang. Classification of Customer Service Tickets in Power System Based on Character and Word Level Semantic Understanding. In *2021 China International Conference on Electricity Distribution (CICED)*, pages 1062–1066. IEEE, 2021.
- [MR18] Alessandro Minelli and Renato Ruffini. Citizen Feedback as a Tool for Continuous Improvement in Local Bodies. *International Journal of Public Sector Management*, 31:46–64, 2018.
- [MRS08] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [MSC⁺13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and Their Compositionality. In *Advances in Neural Information Processing Systems*, volume 26, pages 3111–3119. Curran Associates, Inc., 2013.
- [MZV⁺24] Timothee Mickus, Elaine Zosa, Raul Vazquez, Teemu Vahtola, Jörg Tiedemann, Vincent Segonne, Alessandro Raganato, and Marianna Apidianaki. SemEval-2024 Task 6: SHROOM, a Shared-task on Hallucinations and Related Observable Overgeneration Mistakes. In Atul Kr. Ojha, A. Seza Doğruöz, Harish Tayyar Madabushi, Giovanni Da San Martino, Sara Rosenthal, and Aiala Rosá, editors, *Proceedings of the 18th International Workshop on Semantic Evaluation*, pages 1979–1993, Mexico City, Mexico, 2024. Association for Computational Linguistics.
- [MZW18] Piero Molino, Huaixiu Zheng, and Yi-Chia Wang. Cota: Improving the Speed and Accuracy of Customer Support through Ranking and Deep Networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 586–595. Association for Computing Machinery, 2018.
- [NO20] Rickard Nyman and Paul Ormerod. Text as Data: A Machine Learning-based Approach to Measuring Uncertainty. *ArXiv Preprint arXiv:2006.06457*, 2020.

- [Ope22] OpenAI. Introducing ChatGPT, 2022. Accessed: 2024-07-21.
- [Ope23] OpenAI. GPT-4 technical report. Technical Report arXiv:2303.08774, OpenAI, 2023.
- [OWJ⁺22] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training Language Models to Follow Instructions with Human Feedback, 2022. URL <https://arxiv.org/abs/2203.02155>, 2022.
- [PAT⁺24] Jayr Pereira, Andre Assumpcao, Julio Trecenti, Luiz Airoso, Caio Lente, Jhonatan Cléto, Guilherme Dobins, Rodrigo Nogueira, Luis Mitchell, and Roberto Lotufo. INACIA: Integrating Large Language Models in Brazilian Audit Courts: Opportunities and Challenges. *Digital Government: Research and Practice*, pages 1–20, 2024.
- [Pen46] Lionel S Penrose. The Elementary Statistics of Majority Voting. *Journal of the Royal Statistical Society*, 109:53–57, 1946.
- [PMF⁺23] Alejandro Peña, Aythami Morales, Julian Fierrez, Ignacio Serna, Javier Ortega-Garcia, Iñigo Puente, Jorge Cordova, and Gonzalo Cordova. Leveraging Large Language Models for Topic Classification in the Domain of Public Affairs. In *International Conference on Document Analysis and Recognition*, pages 20–33. Springer, 2023.
- [PRWZ02] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics, 2002.
- [PŞAS24] Ezgi Paket, Göksu Şenerkek, Fatma Betül Akyol, and Furkan Salman. IT Service Desk Ticket Classification via Large Language Models. In *2024 9th International Conference on Computer Science and Engineering (UBMK)*, pages 135–140. IEEE, 2024.
- [PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar, 2014. Association for Computational Linguistics.
- [PSMT24] George Papageorgiou, Vangelis Sarlis, Manolis Maragoudakis, and Christos Tjortjis. Enhancing E-Government Services Through State-of-

- the-Art, Modular, and Reproducible Architecture Over Large Language Models. *Applied Sciences*, 14:8259–8276, 2024.
- [PVG⁺11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning In Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [PYC⁺24] Sakshi Patil, Mayuri Yerande, Pushkaraj Chaudhari, Aaman Bhowmick, and Shanta Sondur. Resolvio: User Feedback and Complaint Analysis Using ML. In *2024 4th Asian Conference on Innovation in Technology (ASIANCON)*, pages 1–4. IEEE, 2024.
- [QCDD21] Shafiullah Qureshi, Ba Chu, Fanny S Demers, and Michel Demers. Using Natural Language Processing to Measure COVID-19-Induced Economic Policy Uncertainty for Canada and the USA. In *International Conference on Time Series and Forecasting*, pages 107–121. Springer, 2021.
- [Qui86] J Ross Quinlan. Induction of Decision Trees. *Machine Learning*, 1:81–106, 1986.
- [R⁺19] Krzysztof Rybinski et al. A Machine Learning Framework for Automated Analysis of Central Bank Communication and Media Discourse: The Case of Narodowy Bank Polski. *Bank i Kredyt*, 1:1–20, 2019.
- [RDN⁺22] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical Text-Conditional Image Generation with CLIP Latents. *ArXiv Preprint arXiv:2204.06125*, 2022.
- [RDSW⁺16] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data Programming: Creating Large Training Sets, Quickly. In *Advances in Neural Information Processing Systems*, volume 29, pages 1–9. Curran Associates, Inc., 2016.
- [RG19] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3982–3992. Association for Computational Linguistics, 2019.
- [RKH⁺21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning Transferable Visual Models from Natural

- Language Supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.
- [RNSS18] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving Language Understanding by Generative Pre-Training. Technical report, OpenAI, 2018.
- [RSM⁺23] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct Preference Optimization: Your Language Model Is Secretly a Reward Model. In *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741. Curran Associates, Inc., 2023.
- [RSR⁺20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the Limits of Transfer Learning With a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.
- [RST19] Margaret E Roberts, Brandon M Stewart, and Dustin Tingley. STM: An R Package for Structural Topic Models. *Journal of Statistical Software*, 91:1–40, 2019.
- [RTMS24] Amar Rao, Marco Tedeschi, Kamel Si Mohammed, and Umer Shahzad. Role of Economic Policy Uncertainty in Energy Commodities Prices Forecasting: Evidence from a Hybrid Deep Learning Approach. *Computational Economics*, 64:3295–3315, 2024.
- [RWC⁺19] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. Technical report.
- [RZ⁺09] Stephen Robertson, Hugo Zaragoza, et al. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, 3:333–389, 2009.
- [Sad22] Perry Sadorsky. Using Machine Learning to Predict Clean Energy Stock Prices: How Important are Market Volatility and Economic Policy Uncertainty? *Journal of Climate Finance*, 1:100002, 2022.
- [SAP⁺24] Goli Sunil, Areefa Areefa, Kota Pragathi, Koyyada Rishitha, Sambari Praveen Kumar, Kothandaraman Dhandapani, and Rajasri Reddy. Tackling Customer Support through NLP. In *AIP Conference Proceedings*, volume 2971, pages 0200601–0200606. AIP Publishing, 2024.

- [SC15] Karen Sanders and María José Canel. Mind the Gap: Local Government Communication Strategies and Spanish Citizens' Perceptions of their Cities. *Public Relations Review*, 41:777–784, 2015.
- [SFA⁺22] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176B-parameter Open-access Multilingual Language Model. *ArXiv Preprint ArXiv:2211.05100*, 2022.
- [SHB⁺24] Vincent J Straub, Youmna Hashem, Jonathan Bright, Satyam Bhagwanani, Deborah Morgan, John Francis, Saba Esnaashari, and Helen Margetts. AI for Bureaucratic Productivity: Measuring the Potential of AI to Help Automate 143 Million UK Government Transactions. *ArXiv Preprint arXiv:2403.14712*, 2024.
- [SKSF⁺22] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734, Seattle, United States, 2022. Association for Computational Linguistics.
- [SM22] Maria Jihan Sangil and Lany L Maceda. Citizen Feedback Analytics for Inclusive Policy Design and Decision-making: A Case Study on the Implementation of Low-speed Streets. In *Proceedings of Seventh International Congress on Information and Communication Technology*, pages 313–325. Springer, 2022.
- [SVCR24] Mudragada Venkata Subbarao, Kasukurthi Venkatarao, Suresh Chittineni, and Sarella Venkata Ramana. Automated Dialogue-Based Response and Resolution of Conversational IT Tickets Using Deep Neural Networks. In *International Conference on Communications and Cyber Physical Engineering 2018*, pages 351–366. Springer, 2024.
- [SWD⁺17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *ArXiv Preprint ArXiv:1707.06347*, 2017.
- [SWR⁺21] Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask Prompted Training Enables Zero-shot Task Generalization. *ArXiv Preprint arXiv:2110.08207*, 2021.

- [TAB⁺23] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a Family of Highly Capable Multimodal Models. *ArXiv Preprint arXiv:2312.11805*, 2023.
- [TB24] Mario Truss and Stephan Boehm. AI-Based Classification of Customer Support Tickets: State-of-the-Art and Implementation with AutoML. *ArXiv Preprint arXiv:2406.01789*, 2024.
- [TBL⁺23] Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Cl  mentine Fourrier, Nathan Habib, et al. Zephyr: Direct Distillation of LM Alignment. *ArXiv Preprint ArXiv:2310.16944*, 2023.
- [TMS⁺23] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. LLAMA2: Open Foundation and Fine-tuned Chat Models. *ArXiv Preprint ArXiv:2307.09288*, 2023.
- [TMT⁺25] Ahmed Taha, Doaa Mabrouk, Ashraqt Tamer, Mariam Tarek, Fatma Alaa, Mahmoud El-Fateh, and Fady Romany. AI Web-Based Smart Ticketing System. *Advanced Sciences and Technology Journal*, 2:1–22, 2025.
- [TND⁺18] Ellen Tobback, Hans Naudts, Walter Daelemans, Enric Junqu   de Fortuny, and David Martens. Belgian Economic Policy Uncertainty Index: Improvement through Text Mining. *International Journal of Forecasting*, 34:355–365, 2018.
- [TZM23] Paul Trust, Ahmed Zahran, and Rosane Minghim. Understanding the Influence of News on Society Decision Making: Application to Economic Policy Uncertainty. *Neural Computing and Applications*, 35:14929–14945, 2023.
- [UPT24] Dmitry Ustalov, Nikita Pavlichenko, and Boris Tseitlin. Learning from Crowds with Crowd-Kit. *Journal of Open Source Software*, 9:6227–6235, 2024.
- [VSP⁺17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, volume 30, pages 1–11. Curran Associates, Inc., 2017.
- [VT24] Sherwin Varghese and James Tian. Efficient Support Ticket Resolution Using Knowledge Graphs. *ArXiv Preprint arXiv:2501.00461*, 2024.

- [WLZF22] Michael D Wang, Jie Lou, Dong Zhang, and C Simon Fan. Measuring Political and Economic Uncertainty: a Supervised Computational Linguistic Approach. *SN Business & Economics*, 2:1–17, 2022.
- [WM24] Jochen Wulf and Juerg Meierhofer. Exploring the Potential of Large Language Models for Automation in Technical Customer Service. *ArXiv Preprint arXiv:2405.09161*, 2024.
- [WMS22] Yasmen Wahba, Nazim H Madhavji, and John Steinbacher. A Hybrid Machine Learning Model for Efficient Classification of IT Support Tickets in The Presence of Class Overlap. In *Proceedings of the 32nd Annual International Conference on Computer Science and Software Engineering*, pages 151–156. IBM Corporation, 2022.
- [WWS⁺22] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022.
- [WYZG24] Shanshan Wang, Xingxing Yang, Xia Zhu, and Xiangyu Ge. The Impacts of Economy Policy Uncertainty on Peer Effects of Firms R&D Investment: Based on LDA Machine Learning and Regression Statistical Modeling Approach. *PLOS ONE*, 19:1–23, 2024.
- [XCC⁺20] Jia Xue, Junxiang Chen, Chen Chen, Chengda Zheng, Sijia Li, and Ting-shao Zhu. Public Discourse and Sentiment During the COVID-19 Pandemic: Using Latent Dirichlet Allocation for Topic Modeling on Twitter. *PLOS ONE*, 15:1–12, 2020.
- [XCG⁺24] Zhentao Xu, Mark Jerome Cruz, Matthew Guevara, Tie Wang, Manasi Deshpande, Xiaofeng Wang, and Zheng Li. Retrieval-Augmented Generation with Knowledge Graphs for Customer Service Question Answering. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2905–2909. Association for Computing Machinery, 2024.
- [XRWW23] Wangfang Xu, Wenjia Rao, Longbao Wei, and Qianqian Wang. A Normalized Global Economic Policy Uncertainty Index From Unsupervised Machine Learning. *Mathematics*, 11:3268, 2023.
- [XXL⁺15] Long Xia, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. Learning Maximal Marginal Relevance Model via Directly Optimizing Diversity Evaluation Measures. In *Proceedings of the 38th International ACM SIGIR*

- Conference on Research and Development in Information Retrieval*, pages 113–122. Association for Computing Machinery, 2015.
- [YAKMS24] Fatma Yiğit Açıkgöz, Mehmet Kayakuş, Georgiana Moiceanu, and Nesrin Sönmez. A New Approach to Assess Sustainable Corporate Reputation with Citizen Comments Using Machine Learning and Natural Language Processing. *Sustainability*, 16:9610, 2024.
- [YDY⁺19] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R. Salakhutdinov, and Quoc V. Le. XLNet: Generalized Autoregressive Pre-training for Language Understanding. In *Advances in Neural Information Processing Systems*, volume 32, pages 5730–5741. Curran Associates, Inc., 2019.
- [YHPC24] Hsiu-Hsuan Yeh, Yu-Lieh Huang, Zihao Park, and Chung-Chi Chen. Automation of Text-Based Economic Indicator Construction: A Pilot Exploration on Economic Policy Uncertainty Index. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 4213–4217. Association for Computing Machinery, 2024.
- [YTK24] Chang Hee Yun, Ai Ping Teoh, and Tze Yin Khaw. Artificial Intelligence Integration in E-Government: Insights from the Korean Case. In *2024 IEEE 3rd International Conference on Electrical Engineering, Big Data and Algorithms*, pages 1159–1164. IEEE, 2024.
- [YYZ⁺24a] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 Technical Report. *ArXiv Preprint arXiv:2412.15115*, 2024.
- [YYZ⁺24b] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc., 2024.
- [YZY⁺23] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*. ICLR, 2023.
- [ZCS⁺23] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging LLM as a Judge with MT-Bench and Chatbot Arena. In

- Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc., 2023.
- [ZDL⁺23] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. Instruction Tuning for Large Language Models: A Survey. *ArXiv Preprint arXiv:2308.10792*, 2023.
- [ZKW⁺20] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*, pages 1–43. ICLR, 2020.
- [ZKZ⁺15] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 19–27. IEEE, 2015.
- [ZMS⁺23] Alessandro Zangari, Matteo Marcuzzo, Michele Schiavinato, Andrea Gasparetto, and Andrea Albarelli. Ticket Automation: An Insight Into Current Research with Applications to Multi-level Classification Scenarios. *Expert Systems with Applications*, 225:119984, 2023.
- [ZRG⁺22] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. OPT: Open Pre-Trained Transformer Language Models. *ArXiv Preprint arXiv:2205.01068*, 2022.
- [ZSW⁺19] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-Tuning Language Models from Human Preferences. *ArXiv Preprint ArXiv:1909.08593*, 2019.
- [ZWM19] Peixiang Zhong, Di Wang, and Chunyan Miao. An Affect Rich Neural Conversational Model with Biased Attention and Weighted Cross-Entropy Loss. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7492–7500. AAAI, 2019.
- [ZZ13] Min-Ling Zhang and Zhi-Hua Zhou. A Review on Multi-label Learning Algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26:1819–1837, 2013.