

Harnessing Generalist LLMs for Diverse Objective and Subjective NLP Tasks

by

Gaurav Sahu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2024

© Gaurav Sahu 2024

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner: Jimmy Huang
Professor, School of Information Technology,
York University

Supervisor(s): Olga Vechtomova
Professor, Dept. of Management Science and Engineering,
Cheriton School of Computer Science (cross-appointment),
University of Waterloo

Internal Members: Jesse Hoey
Professor, Cheriton School of Computer Science,
University of Waterloo
Robin Cohen
Professor, Cheriton School of Computer Science,
University of Waterloo

Internal-External Member: Randy Harris
Professor, Dept. of English Language and Literature,
Cheriton School of Computer Science (cross-appointment),
University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Research presented in Chapter 3. This chapter is based on the work [Sahu et al. \(2022\)](#), where Dr. Bahdanau was the primary investigator. As the first author, I developed the methodology, designed the experimental setup, and wrote the paper with meaningful suggestions from Dr. Rodriguez, Dr. Laradji, Dr. Vazquez, Dr. Bahdanau, and Ms. Atighehchian.

Research presented in Chapter 4. This chapter is based on the work [Sahu et al. \(2023a\)](#), where Dr. Laradji and Dr. Vechtomova were the primary investigators. As the first author, I developed the methodology, designed the experimental setup, and wrote the paper with meaningful feedback from Dr. Vechtomova, Dr. Laradji, and Dr. Bahdanau.

Research presented in Chapters 5 and 6. These chapters are based on the works [Sahu et al. \(2023b\)](#) and [Sahu and Laradji \(2024\)](#), where Dr. Laradji was the primary investigator. Dr. Vechtomova and Dr. Laradji provided meaningful suggestions during the writeup of [Sahu et al. \(2023b\)](#), while Dr. Laradji provided feedback on write up of [Sahu and Laradji \(2024\)](#). As the first author, I developed the LLM-based methodology for semi-supervised text summarization discussed in Chapter 5, and MixSumm, the more robust LLM-based method employing data augmentation for few-shot text summarization, presented in Chapter 6. I also designed the experimental setup for the two works.

Research presented in Chapter 7. This chapter is based on the work [Sahu and Vechtomova \(2024\)](#), where Dr. Vechtomova was the primary investigator. As the first author, I developed the framework for modeling artistic preferences, designed the experimental setup, and wrote the paper with meaningful feedback from Dr. Vechtomova. Dr. Vechtomova aided in finding relevant literature for this work and suggested capturing Imagery, Banality, and NPMI as features in the presented framework, in addition to providing meaningful suggestions and guidance in the research meetings throughout the course of the project.

Research presented in Chapter 8. This chapter is based on the work [Sahu et al. \(2024\)](#), where Dr. Laradji was the primary investigator. As the first author, I developed AgentPoirot, the proposed agent, designed the experimental setup, and wrote the paper with meaningful feedback from Dr. Laradji, Dr. Mudumba, Dr. Pal, Dr. Chapados, Dr. Vazquez, Dr. Lacoste, Dr. Zantedeschi, Dr. Taslakian, Dr. Drouin, Mr. Puri, and Mr. Rodriguez. Dr. Mudumba further helped by generating the first 30 out of 100 datasets in InsightBench while I generated the rest. In addition, Mr. Puri and Mr. Rodriguez helped develop an early demonstration for AgentPoirot.

Abstract

Recent advances in natural language processing (NLP), particularly in the subspace of large language modeling, have led to a major paradigm shift. Large language models (LLMs), like the GPT and LLaMA family of models, are trained on a massive Internet corpus covering data from a gamut of diverse domains. In addition, the billions of parameters in these models also invoke emergent capabilities in them, leading to strong improvements across diverse NLP tasks without much task-specific tuning; however, effectively harnessing the knowledge of these generalist models for real-world data still remains a major challenge as the LLMs can produce inconsistent, biased, and unsatisfactory outputs. In this thesis, we propose task-specific strategies for effectively leveraging LLMs for a number of challenging NLP tasks, such as (low-resource) text classification, text summarization, modeling artistic preferences of creative individuals, and automated data analysis.

Our results suggest that LLMs can serve as excellent data generators and data labelers for well-defined single-step tasks like classification and summarization, crucially in data-scarce settings, where models trained on LLM-generated data achieved competitive performance to oracle models trained on a much larger labeled training data. On the other hand, for more subjective tasks like modeling artistic preferences among creative individuals, we demonstrate that while LLMs might not be able to discern between the likes and dislikes of artists, they can be effective in extracting key linguistic and poetic properties from text that can later be employed to infer artistic preferences among different individuals. Lastly, we also evaluate the effectiveness of LLMs in multi-step tasks that require the LLM to perform multiple tasks in tandem without compromising performance for individual tasks. Overall, our work draws critical insights into the strengths and shortcomings of LLMs for a wide range of subjective and objective NLP tasks and includes meaningful suggestions for the research community to harness LLMs for those tasks effectively.

Acknowledgements

I am fortunate to have had the opportunity to work with some of the best minds in the field of natural language processing. First, I would like to express my deepest gratitude to my supervisor, Prof. Olga Vechtomova, for constant guidance, support, encouragement, and patience throughout my Ph.D. journey. I have learned a lot from her, not only about research but also from our deep discussions on art. I also want to thank the incredible team of researchers at ServiceNow Research, who have made a meaningful impact on my research journey.

I would also like to thank my committee members, Prof. Jesse Hoey, Prof. Robin Cohen, and Prof. Randy Harris, and my external examiner, Prof. Jimmy Huang, for agreeing to evaluate my thesis and providing great suggestions to improve it.

A special token of thanks to Dara for her unwavering support, encouragement, and love throughout the years and to Prof. Robin for her invaluable advice during my Ph.D. journey.

Thanks to all my friends who made me feel at home—know that I sincerely appreciate the time I spent with you. Lastly, I would like to mention a very special group of people—the Runners’ Choice Marathon Team, Waterloo. YOU ARE THE BEST! I have found some of the most inspiring individuals in this group, and I have learned so much from your discipline, grit, camaraderie, and humility! Those trail runs, those hill repeats, those long runs, those “easy” runs have all been a critical ingredient in keeping my mind and body intact during the final year of my Ph.D. So, a big, big thank you for letting me be a part of your community.

Dedication

Lovingly dedicated to my parents, whose boundless love has been the cornerstone of my journey, whose quiet strength has illuminated my path, and whose unwavering faith in me has been my most profound inspiration. In moments of doubt and triumph alike, you have been my steadfast anchors—believing in me even when I couldn't, encouraging my dreams with gentle wisdom and selfless support. This achievement is not mine alone, but a testament to the love, sacrifice, and resilience you have embodied throughout my life. For your endless love and inspiration, I am forever grateful.

Table of Contents

Examining Committee Membership	ii
Author’s Declaration	iii
Statement of Contributions	iv
Abstract	v
Acknowledgements	vi
Dedication	vii
List of Figures	xiv
List of Tables	xx
1 Introduction	1
1.1 Research Contributions	4
1.2 Progression of the Thesis	4
2 Background and Related Work	8
2.1 Evolution of Generative Models in NLP	9
2.1.1 n -gram Models	9

2.1.2	Hidden Markov Models (HMMs)	11
2.1.3	Recurrent Neural Networks (RNNs)	14
2.1.4	Generative Adversarial Networks (GANs)	15
2.1.5	Transformer-based Models	17
2.2	Related Work	26
2.2.1	Text Classification	26
2.2.2	Text Summarization	29
2.2.3	Modeling Artistic Preferences	32
2.2.4	Automated Data Analysis	32
3	Leveraging LLMs for Data Augmentation in Text Classification	35
3.1	Methodology	37
3.2	Experimental Setup	38
3.2.1	Datasets	38
3.2.2	Setup	38
3.2.3	Training and Evaluation	39
3.3	Results	41
3.3.1	Analysis of GPT Outputs	44
3.3.2	Can GPT Models Understand Close Intents?	46
3.3.3	Comparison with GPT3Mix	46
3.4	Limitations	47
3.5	Conclusion	47
3.6	Key Takeaways	48
4	Advanced Data Augmentation with LLMs for Text Classification	50
4.1	Methodology	52
4.1.1	Step 1: Generating examples	53
4.1.2	Step 2: Improving Fidelity	54

4.2	Does PromptMix Generate Borderline Examples?	54
4.3	Experimental Setup	56
4.3.1	Datasets	56
4.3.2	Few-shot Setup	56
4.3.3	Training and Evaluation	57
4.4	Results	59
4.4.1	On the Effect of Relabeling Step	59
4.4.2	Borderline Examples Aid Knowledge Distillation	61
4.4.3	PromptMix v/s Other Augmentation Methods	61
4.4.4	Descriptions are Impactful	62
4.4.5	Open-source v/s Closed-source models	62
4.5	Limitations	63
4.6	Conclusion	64
4.7	Key Takeaways	64
5	Semi-Supervised Extractive Text Summarization with LLMs	65
5.1	Methodology	68
5.1.1	Stage 1: Training the Teacher Model	68
5.1.2	Stage 2: Generating Pseudo-labels using the Teacher Model	68
5.1.3	Stage 3: LLM Relabeling of Teacher’s Pseudo-labels	69
5.1.4	Stage 4: LLM Scoring of Pseudo-labels	69
5.2	Experimental Setup	70
5.2.1	Datasets	71
5.2.2	Training and Evaluation Details.	72
5.2.3	Baselines	74
5.3	Results	74
5.3.1	Comparison of Pseudo-label Selection Strategies	75
5.3.2	Effect of Relabeling	76

5.3.3	On Sample Efficiency of Different Approaches	77
5.4	Limitations	78
5.5	Conclusion	78
5.6	Key Takeaways	79
6	LLM-based Data Augmentation for Low-resource Text Summarization	80
6.1	Notations	82
6.2	Methodology	83
6.3	Experimental Setup	84
6.3.1	Datasets	84
6.3.2	Evaluation.	85
6.3.3	Implementation Details	86
6.3.4	Baselines	87
6.4	Results	88
6.4.1	<i>MixSumm</i> Generates Diverse Documents.	88
6.4.2	Effect of Clustering Documents	90
6.4.3	DA v/s SSL Methods	90
6.4.4	Knowledge Distillation from LLaMA-3	91
6.5	Limitations	91
6.6	Conclusion	92
6.7	Key Takeaways	92
7	Computational Modeling of Artistic Inspiration	94
7.1	Methodology	95
7.2	Step 1: Identifying The Key Poetic and Stylistic Characteristics	96
7.2.1	Poetic Imagery	97
7.2.2	Word Energy	97
7.2.3	Level of Abstraction	98

7.2.4	Valence	99
7.2.5	Surprisal	101
7.2.6	Contextual Entropy	101
7.2.7	Normalized Pointwise Mutual Information (NPMI)	102
7.2.8	Banality	103
7.3	Step 2: Training the Calibration Network	103
7.4	Experimental Setup	104
7.4.1	Dataset	104
7.4.2	Experiments	107
7.5	Results	112
7.5.1	Quantitative Results	112
7.5.2	Qualitative Examples	115
7.6	Limitations	115
7.7	Conclusion	115
7.8	Key Takeaways	116
8	LLMs for Advanced Data Analytics	118
8.1	Experimental Setup	121
8.1.1	Datasets	121
8.1.2	Evaluation	122
8.2	Methodology: AgentPoirot	124
8.2.1	Baselines	124
8.2.2	Implementation Details	125
8.3	Results	126
8.4	Limitations	130
8.5	Conclusion	130
8.6	Key Takeaways	131

9	Future Work	132
9.1	Progressing to Multi-Modal LLMs	132
9.2	Attribution in LLMs	134
9.3	Beyond Standard Benchmarks	139
10	Conclusion	142
10.1	Implications of Our Work	142
10.2	Recommendations for NLP Researchers	143
	References	145
	APPENDICES	174
A	Worked Out Examples	175
A.1	HMM Worked Out Example	175
A.2	Transformer Worked Out Example	180
B	Miscellaneous	184
C	Prompts Used	187

List of Figures

1.1	Progression of topics in this thesis.	5
2.1	Hidden Markov Model, and its probabilistic parameters (example). X – states; y – possible observations; a – state transition probabilities; b – output probabilities. Source — Wikipedia.	12
2.2	Recurrent Neural Network (RNN) architecture. Here, $x^{(t)}$ represents the input at time step t , $h^{(t)}$ denotes the hidden state at time step t , and $y^{(t)}$ is the output at time step t . The recurrent unit processes the input sequence iteratively, updating the hidden state based on the current input and the previous hidden state. Image source: Goodfellow et al. (2016)	13
2.3	Sequence-to-Sequence (Seq2Seq) model architecture. The encoder processes the input sequence and generates a fixed-length vector representation, which is then decoded by the decoder to produce the output sequence. The model learns to map input sequences to output sequences, making it suitable for tasks like machine translation. In this example, the source sentence in French is being translated to English. Image source: PyTorch Seq2Seq Tutorials (2024)	14
2.4	Comparison of latent spaces between a deterministic autoencoder and a variational autoencoder (VAE). The VAE learns a regularized, continuous latent space (right) by imposing a standard normal Gaussian prior, while the deterministic autoencoder learns an unregularized, potentially discontinuous latent space (left). Note: the points in the latent space are illustrative and do not represent actual data.	16
2.5	GAN architecture for text generation. The generator G generates text $\tilde{x}$ from random noise z , attempting to fool the discriminator D , which tries to distinguish between real text x and generated text $\tilde{x}$. The red dashed arrow indicates the gradient flow during training.	17

2.6	Transformer architecture and multi-head attention mechanism. The Transformer model consists of an encoder-decoder architecture with self-attention layers. The multi-head attention mechanism is a key component of the Transformer, allowing the model to focus on different parts of the input sequence simultaneously. Image source: (Vaswani et al., 2017).	18
2.7	Illustration of the Next Sentence Prediction (NSP) task in BERT. The model is trained to predict whether the second sentence follows the first sentence in the original text. The input sentences are separated by a special [SEP] token, and a [CLS] token is added to the beginning of the first sentence. The [CLS] token is used as an aggregate representation of the input sequence, which is then used to predict whether the second sentence is a continuation of the first sentence. Image source: Devlin et al. (2019b).	20
2.8	Fine-tuning BERT on four different downstream tasks: natural language inference, sentiment analysis, named entity recognition, and question answering. The pre-trained BERT model is fine-tuned on task-specific datasets to adapt to the target tasks. Note: for the NER task in (d), O denotes the token for non-entity words and B-PER denotes the beginning of a person entity. Image source: Devlin et al. (2019b).	22
2.9	Illustration of how the pre-trained BERT model (on left) is fine-tuned on various downstream tasks (on right). Image source: Devlin et al. (2019b). .	23
2.10	Illustration of prompt engineering techniques: (a) In-context learning (ICL) involves providing the model with additional context to improve its performance on a specific task. (b) Few-shot learning involves training the model on a small number of examples to adapt it to new tasks. (c) Chain-of-thought prompting requires the model to provide its reasoning process before arriving at a final answer. Image source: Brown et al. (2020); Wei et al. (2022).	24
2.11	Retrieval-Augmented Generation (RAG) architecture. The model combines a generative model with a retrieval component that retrieves relevant information from a knowledge base to guide the generation process. The retrieval component provides context and information to the generative model, enabling it to generate more accurate responses. Image source: Lewis et al. (2020b).	25

3.1	Generation Process. Given a seed intent (here, <code>music likeness</code>) and $K(=10)$ available examples for that intent, we construct a prompt following the shown template. Note that the last line of the prompt is incomplete (there is no newline character.) We then feed this prompt to a GPT-3 engine, which generates some completions of the prompt. In this example, red text denotes unfaithful examples and blue text denotes faithful examples. Note: For brevity, we only show ten generated sentences.	36
3.2	Partial few-shot validation performance for different GPT-3 models and temperatures. (a) few-shot accuracy, (b) OOS recall of intent classifiers trained on augmented sets, and (c) fidelity measured as the accuracy of the oracle on the augmented sets.	42
3.3	Full few-shot validation performance for different GPT-J temperatures on different datasets. (a) few-shot in-scope accuracy of intent classifiers trained on augmented sets, and (b) fidelity (oracle accuracy) of augmented sets generated by GPT-J with different temperatures.	43
3.4	Distribution of labels as predicted by the oracle for lowest-fidelity intents in Banking77 and HWU64 datasets (“ <code>topping_up_by_card</code> ” and “ <code>music likeness</code> ,” respectively). Green areas denote the portion of generated sentences deemed fit by the oracle for the lowest-fidelity intents in the two datasets. Red and Blue areas respectively correspond to the most common and the second most common alternative intent predicted by the oracle.	45
4.1	PromptMix focuses on generating examples near the class boundary of two classes, unlike other standard augmentation approaches like EDA (Wei and Zou, 2019) and prompting-based methods without Mixup (Sahu et al., 2022; Lin et al., 2023), that only use the information of a single class for augmentation.	51
4.2	PromptMix framework. The dashed box shows the generation process for every class in the dataset. Step 1: we generate augmentations (yellow documents) by feeding the Mixup prompt proposed in Section 4.1.1 to an LLM. Step 2: we relabel <u>all</u> the augmentations using an LLM (as described in Section 4.1.2) to fix any incorrect labels from Step 1. Note: LLM_1 and LLM_2 can be identical. Refer to Figures C.2 and C.3 for detailed versions of our prompts.	52
4.3	$\alpha = \text{round}(10(x + 1))/20 \forall x \sim \beta(5, 2)$	53

5.1	L-Eval scores of semi-supervised v/s fully supervised models in a low-data setting. The proposed semi-supervised method (middle column for each dataset) combines PreSumm’s confidence with GPT-4’s knowledge to generate better pseudo-labels. The proposed method performs competitively on the three summarization datasets while using $6\times$ fewer labels than a fully supervised method (rightmost column for each dataset)	66
5.2	PPSL pipeline. Our approach has four main stages. Step 1: train a teacher model M on the limited labeled dataset. Step 2: generate pseudo-labels for the unlabeled set with M and shortlist 50 based on teacher confidence (see Equation 5.1). Step 3: prompt an LLM to summarize the short-listed documents. Step 4: score the pseudo-labels in Stage 3 by prompting an LLM and select the top 5. These summaries are then added to the training data for the next cycle.	67
5.3	Quality of pseudo-labels by different strategies (data-scarce setup). The y-axis denotes the ROUGE-2 scores of the top 5 pseudo-labels computed against the respective ground truths. All results are for BERT _{base} as the backbone for PreSumm and three random seeds. Refer to Section 5.3.1 for complete details.	76
5.4	ROUGE-1 curves v/s # cycles for data-scarce setting. Each cycle denotes an addition of 5 new pseudo-labels to the training set. All results use BERT _{base} as the backbone for PreSumm. The curves are averaged for three seeds (the width denotes the std). Note that we report the GPT-4 version of our method here.	77
6.1	L-Eval scores of different methods on low-resource extractive text summarization. The proposed <i>MixSumm</i> approach generates new documents by combining topics from multiple examples and outperforms other methods, including a strong LLM-based DA method (<i>MixSumm</i> w/o mixup) and a prompt-based semi-supervised approach (PPSL).	81
6.2	<i>MixSumm</i> pipeline. We first group the documents into T groups using the k -means algorithm. Then, we construct the prompt for LLaMA-3-70b-Instruct by including documents from different groups and instructing the LLM to mix information from multiple topics when generating the new documents. Finally, we train a PreSumm extractive summarizer (Liu and Lapata, 2019) on the combined seed and the synthesized dataset. For abstractive summarization, we add a DistilBART model after PreSumm.	82

7.1	Inter-annotator agreement on the EvocativeLines (small) dataset. . . .	105
7.2	Radar charts showing the 10 different preference profiles of annotators in the <i>EvocativeLines</i> dataset. Blue webs denote the web for positive lines, and red webs denote the negative lines.	111
7.3	Examples of lyric lines rated as inspiring (blue) and not inspiring (red) by annotator $\mathcal{A}_1$	117
8.1	Illustration of an agentic AI system. Image Source: https://github.com/Jenqyang/Awesome-AI-Agents	119
8.2	This diagram illustrates the scope of <i>InsightBench</i> , categorizing 100 distinct datasets into five key thematic areas.	122
8.3	<i>AgentPoirot</i> uses the “Question Generation (or QG) Prompt” (Prompt C.4 in Appendix C) that takes as input the dataset schema and the goal to generate k high-level questions. Then, it uses the “Code Generation (or CG) Prompt” (Prompt C.5 in Appendix C) to generate plots answering the high-level questions. Using the “Insight Extraction (or IE) Prompt” (Prompt C.8 in Appendix C), and the outputs from the previous step, it generates an <i>insight</i> and the <i>justification</i> of that insight. Finally, <i>AgentPoirot</i> uses the “Follow-Up Question (FQ) Prompt” (Prompt C.6 in Appendix C) to generate diverse follow-up questions and selects the top one using the “Question Selection (or QS) Prompt” (Prompt C.9 in Appendix C). This question is injected back into the CG prompt to start the cycle again.	123
8.4	Performance of different agents on <i>InsightBench</i> grouped by difficulty and dataset category.	127
8.5	Sensitivity Analysis.	128
9.1	Effect of attribution on the precision and recall of the agentic LitLLM agent.	138
C.1	Using GPT-3 as a classifier. Given a triplet of close intents, we mix and shuffle the multiple seed examples available for each of them. Then, we append an incomplete line to the prompt with just the generated sentence and feed it to GPT-3 multiple times. Among the responses, we choose the most generated in-triplet intent as the predicted intent (“music.likeness” in the above example). Note: For brevity, we don’t show all the seed examples and predictions.	188

C.2	PromptMix prompt. The demonstration highlights the two main parts of our prompt: Part 1 shows the description and examples, and Part 2 shows the mixup instruction. We use GPT3.5-turbo to obtain the completions. In this example, we highlight good generations in blue and bad generations in red . Note: for brevity, we only include two classes in the prompt.	189
C.3	Relabeling prompt. The generated sentence does not belong to the class <code>age_limit</code> as the main objective of the query entails an ATM support query, which is fixed after relabeling. Note: we relabel both good and bad examples. The color in the figure is just for demonstration purposes.	190
C.4	Different prompts used in the experiments.	205
C.5	Different prompts used in the experiments.	206
C.6	Prompt used to obtain the number of occurrences of four linguistic categories (DAVs, IAVs, SVs, and ADJs) in a lyric line.	207
C.7	Guidelines presented to the eight human labelers.	208
C.8	Prompt used to classify a lyric line with LLaMA-3-70b.	209
C.9	Prompt used to classify a lyric line into one of 29 emotion categories with LLaMA-3-70b.	210
C.10	Prompt used to measure poetic imagery of a lyric line.	211
C.11	Prompt used to measure banality of a lyric line.	212
C.12	Prompt used to measure word energy of a lyric line.	213

List of Tables

3.1	Statistics of the intent classification datasets that we use in our experiments. * indicates that we split the original data into training and validation instead of using a split provided by the dataset authors. For CLINC150, the number of out-of-scope examples in different data partitions is given in parentheses.	38
3.2	Full few-shot results on CLINC150, HWU64, Banking77, and SNIPS datasets. IA : Inscope Accuracy (mean (std)). OR : OOS-Recall (mean (std)). Towards the top of the table, we also report the test set performance (enclosed in parentheses) when all examples are used for fine-tuning (without any augmentation.)	40
3.3	Partial few-shot results on CLINC150 and SNIPS datasets. Refer to Section 3.2.3 for more details. IA : Inscope accuracy (mean (std)). OR : OOS Recall (mean (std)). A : Accuracy (mean (std)). ♠ (Lee et al., 2021).	41
3.4	The impact and the accuracy of using GPT-3 as a 3-way classifier on close intent triplets from HWU64 and Banking77 datasets. For fidelity, generated examples are rejected if the GPT-3 classifier labels them as not belonging to the seed intent. Classification accuracies are reported on the reduced validation+test sets where we only consider examples from the three confounding intents.	44
3.5	Davinci-generated sentences for closely-related intents in HWU64 and Banking77 datasets. Highlighted sub-strings indicate a difference with respect to the seed intent.	49
4.1	Validation accuracy for different values of t on TREC6. Refer to Section 4.3 for details on the experimental setup.	52

4.2	Effect of Mixup. GPT3.5-turbo generations for the prompt shown in Figure C.2. We’ve highlighted parts about age_limit in yellow and parts about atm_support in cyan. We see clear evidence of mixup at work as sentences generated using mixup contain information about both classes. .	55
4.3	Statistics of the text classification datasets we use in our experiments. * indicates that we split the original data into training and validation/testing instead of using a split provided by the dataset authors.	56
4.4	Test classification accuracy (out of 100%) on four datasets, averaged across three random seeds (with standard deviation in brackets). A1 is the accuracy of the classifier on the generated dataset, and A2 is the accuracy on the generated+reabeled dataset. Note: we use GPT3Mix results from Yoo et al. (2021), and LINDA results from Kim et al. (2021). “-” for a method shows that the particular prompt feature is inapplicable to that method. .	57
4.5	Comparing 2-shot PromptMix results with 5-shot baselines on B77. Note: CPFT, USE, CONVERT, and USE+CONVERT as reported in Zhang et al. (2021).	59
4.6	Percentage of generated examples relabeled by GPT3.5-turbo in Step 2 for different methods. Note: percentages are averaged across three runs. . . .	59
4.7	Effect of Relabeling. GPT3.5-turbo sentences that were relabeled in the Twitter Complaints and Banking77 datasets. Highlighted text denotes the difference with respect to the class of the generated sentence before relabeling.	60
4.8	Comparison of different open-source LLMs v/s GPT on TREC6. Note: Baseline refers to the 2-shot accuracy of the BERT _{base} classifier. We replace GPT3.5-turbo with GPT4 and LLama-2-70b-chat-hf to get the NN+GPT4 and NN+LLama-2-70b-chat-hf baselines, respectively.	63
5.1	Statistics of the text summarization datasets we use in our experiments. . .	71
5.2	Mean (Std.) ROUGE F-1 scores of different pseudo-labeling strategies. R-1, R-2, and R-L denote ROUGE-1, ROUGE-2, and ROUGE-L metrics, respectively. TSL results from (Zhuang et al., 2023). Refer to Section 5.2.3 for method details. Bold indicates the best-performing and <u>underline</u> denotes the second-best performing method, respectively.	73

5.3	Fully-supervised methods (first two rows) semi-supervised approaches (remaining rows). All models use BERT _{base} as PreSumm’s backbone. The number of labeled examples for fully supervised models is shown in brackets. The semi-supervised methods use 50/500 labeled and 250 unlabeled examples.	75
6.1	Summarization Results. Comparison of different text summarization models on TweetSumm, WikiHow, and ArXiv/PubMed datasets. We report ROUGE-1 (R-1), ROUGE-2 (R-2), ROUGE-L (R-L) F ₁ scores, and L-Eval scores. We report the mean $\pm$ std. performance across 5 different seeds. Refer to Appendix 6.3.3 and Section 6.3.4 for metric and implementation details. Note. TSL results are reported from (Zhuang et al., 2023). For EDA and <i>MixSumm</i> , we use a 50-shot $\mathcal{D}_{F,train}$ and generate 1000 examples as $\mathcal{D}_{A,train}$. Bold denotes the best-performing model in a given block and highlight denotes the <i>overall</i> best-performing model. For the ArXiv/PubMed dataset, we could fit only 2 documents into LLaMA-3’s context (1 from $\mathcal{D}_{F,train}$ + 1 generated), so we do not report LLaMA-3 (5-shot).	85
6.2	Validation ROUGE-2 and L-Eval scores for different values of T on the TweetSumm datasets.	86
6.3	Full qualitative comparison of EDA, <i>MixSumm</i> w/o mixup, and <i>MixSumm</i> examples. Refer to Section 6.4.1 for a detailed discussion.	89
7.1	Summary of features and their dimensions. Here, T is the number of words (or time steps) in the input sequence, and m is the total number of emotions (29, in our case).	104
7.2	Dataset statistics, broken down by annotator.	106
7.3	Search space for grid search of LSTM. Note: n is from Equation 7.20	109
7.4	Comparison of different frameworks on <i>EvocativeLines</i> . The first section shows the performance of DL-based methods, the second section shows the performance of ML-based methods, and the last section shows the performance of LLM-based methods.	110
7.5	Performance of the proposed framework with LSTM+Attn and XGBoost calibration networks on the <i>EvocativeLines (small)</i> dataset.	112
7.6	Results of interaction testing. I1 denotes the interaction between linguistic features (surprisal, entropy, banality, npmi), and I2 denotes the interaction between stylistic features (imagery, word energy, abstraction, valence).	113

8.1	Performance of different agents on <i>InsightBench</i> . All results are for 5 different seeds.	126
8.2	Comparison of different evaluation methods on the first 10 datasets of <i>InsightBench</i>	128
8.3	Mismatched examples by many-to-many compared to one-to-many evaluation scheme. Parts in bold indicate why the prediction is appropriate to compare against the ground truth insight.	129
9.1	Example use case where attribution helps understand reasoning behind an agent’s decision-making process.	140
B.1	Instances of insights from <i>InsightBench</i> for each category.	185
B.2	PandasAgent v/s AgentPoirot. Bold text shows why an insight is a close approximation of the ground truth insight.	186

Chapter 1

Introduction

Natural Language Processing (NLP) is a subfield of artificial intelligence that focuses on enabling computers to understand, interpret, and generate human language. NLP sits at the core of our day-to-day lives. From customer support chatbots to summarizing news articles, NLP applications are everywhere. It spans a wide array of tasks, such as determining whether a restaurant review is positive or negative (sentiment analysis), answering a question from a Wikipedia article (question answering), or determining the part-of-speech of a word in a sentence (POS tagging). Formally, NLP solutions designed to solve these tasks can be divided into two categories: discriminative and generative.

Discriminative methods focus on directly modeling the decision boundary between possible set of outcomes or classes. Common architectures used in discriminative NLP solutions include logistic regression, Support Vector Machines (SVMs), Conditional Random Fields (CRFs), and neural networks. For instance, CRFs are commonly used in sequence labeling tasks like named entity recognition, where they maximize the conditional probability of a sequence label given an observed sequence.

On the other hand, generative methods are designed to model the joint probability of input-output sequence pairs, enabling them to generate new, plausible data sequences based on learned patterns. Unlike discriminative methods, which focus solely on predicting labels, generative methods learn the underlying distribution of text data, allowing them to create coherent sentences, translate languages, or simulate conversations. Traditional generative models, such as n-grams and Hidden Markov Models (HMMs), rely on probabilistic approaches to generate text, though they often struggle with capturing long dependencies. More powerful deep-learning based architectures such as Autoencoders (AEs), and Sequence-to-Sequence models used recurrent neural networks (RNNs) to create

richer, more contextually aware text.

Traditionally, discriminative methods offered better performance for classification tasks than generative models, as they focus solely on the conditional relationship between inputs and outputs, which allows them to be more accurate and computationally efficient; however, it all changed with the advent of transformers.

The transformer architecture deviates from the sequential nature of RNNs by simultaneously processing all the input tokens using attention mechanism. The lack of sequential dependency in transformers allows parallelization during training and also made possible training these models on large scale internet data. The first generation of transformer-based models, such as, BERT and GPT outperformed all existing NLP methods on multiple natural language understanding tasks. These models excelled in tasks like text generation, summarization, and machine translation by producing outputs that resemble natural language and reflect intricate relationships in data. The most recent generation of transformer-based models includes highly performant *generative* language models, such as GPT-3, PaLM, LLaMA-3, and Claude, have pushed the boundaries of natural language understanding and generation. These models leverage massive datasets and high-capacity architectures to achieve unprecedented performance across a wide array of tasks, from conversational AI and code generation to creative writing and complex reasoning. By scaling both model size and training data, these advanced models capture nuanced context, learn subtle patterns, and can generalize across diverse applications, enabling them to perform at or near human level on various benchmarks. Moreover, a critical development is that these generative models now also excel at classification tasks, demonstrating strong performance in distinguishing complex categories across various domains. By fine-tuning or adapting their architectures, they have achieved high accuracy in sentiment analysis, topic categorization, and other classification tasks, rivaling dedicated classification models while maintaining the versatility to generate coherent, contextually relevant text.

While these LLMs achieve impressive results on standardized benchmarks, their performance often falls short when applied to real-life data, even for similar tasks. Benchmarks are typically curated and may not fully capture the variability, noise, and ambiguity of real-world text, leading to a gap between their benchmark success and practical reliability. For instance, LLMs might produce factually incorrect or biased outputs when faced with unseen or nuanced cases that deviate from their training distribution. Moreover, the variability in their responses to the same prompt can be troubling, as different outputs can introduce inconsistencies that complicate user trust and decision-making. This gap underscores the need for continued development to ensure that LLMs perform robustly and accurately in the dynamic and diverse contexts of real-world applications.

In this thesis, we aim to study the utility of large language models (LLMs)—artificial neural networks trained on massive datasets, for a diverse set of tasks that simulate the complexities encountered in practical applications—for two broad categories of tasks: single-step and multi-step. Single-step tasks require an LLM to perform a singular task at a time, while multi-step tasks requires the LLM to perform multiple tasks to achieve a singular goal. Critically, in multi-step tasks, an LLM is not only required to perform the individual tasks well, but it is expected to do them *in tandem* with other tasks to achieve a common goal. We explore both single-step and multi-step tasks to understand the capabilities and limitations of LLMs in different scenarios and identify strategies to enhance their performance in real-world applications.

Among the single-step tasks, we focus on text classification, text summarization, and artistic preference modeling. Each of these tasks involves performing a singular, specific function: predicting the category of a text, generating a summary of source material, or providing a score to measure the presence of a particular feature in a text (e.g., assessing the degree of poetic imagery in a lyric line, respectively). Further, we study the utility of LLMs in the challenging multi-step task of automated exploratory data analysis, which involves performing multiple individual tasks *in tandem* without compromising the efficacy for the individual tasks. For instance, to excel at automated data analysis, an LLM must excel at multiple tasks, such as identifying questions to ask about a dataset, generating code to answer those questions, and providing insights based on the code outputs. By exploring both single-step and multi-step tasks, we aim to understand the capabilities and limitations of LLMs in different scenarios and identify strategies to enhance their performance

Notably, we add additional challenging constraints to the selected tasks to simulate real-life scenarios, such as data scarcity, noise, and ambiguity. These constraints are crucial, as they allow us to evaluate the robustness and adaptability of LLMs in practical scenarios where the data is often incomplete, noisy, or biased. By testing LLMs under these conditions, we can identify their shortcomings and develop strategies to mitigate them, thereby enhancing their reliability and effectiveness in real-world applications. We also ensure that our tasks cover various complexities encountered in practical applications. For instance, our few-shot text classification setup tests the models in a data-scarce scenario with minimal or no labeled data. We also work on few-shot and semi-supervised text summarization for both short and long document summarization to assess how well LLMs can condense and convey essential information across different content lengths and formats. This dual focus allows us to investigate the models’ capabilities in handling varying degrees of complexity in text, from succinct news articles to comprehensive reports. In addition to the two objective tasks with clear task definitions, we tackle the challenge of modeling the artistic preferences of creative individuals, which is particularly challenging

due to its inherent subjectivity (for instance, what is “inspiring” to one individual may not be to another). Together, these tasks not only test the versatility of LLMs but also provide a comprehensive framework for evaluating their effectiveness in real-world scenarios. We describe below the core contributions of this thesis and its progression in the subsequent chapters.

1.1 Research Contributions

We now summarize the core contributions of this thesis below:

1. We propose novel prompt-based data augmentation strategies for few-shot and zero-shot text classification to synthesize high-quality, diverse datasets. We also demonstrate that our method facilitates the distillation of knowledge from 175B LLMs to 110M classifiers.
2. We evaluate the efficacy of employing LLMs for few-shot and semi-supervised text summarization for both short and long documents. We conduct comprehensive experiments to demonstrate that **a)** LLMs are effective data generators and pseudo-labelers for text summarization, and **b)** it is possible to achieve knowledge distillation in the generative task of text summarization as well.
3. We propose a novel framework to employ LLMs for inherently subjective tasks, such as modeling the artistic preferences of different individuals, comparing literary works of different artists, or studying the progression of artistic styles over time. We demonstrate that while LLMs may not be able to discern between the likes and dislikes of creative individuals, they can extract key linguistic and poetic features from text that can be used to infer artistic preferences among different individuals.
4. We also propose, AgentPoirot, a novel framework to employ LLMs for automated exploratory data analysis. We demonstrate that LLMs can effectively generate questions, code, and insights for a given dataset, thus automating the exploratory data analysis process.

1.2 Progression of the Thesis

We introduce the foundational concepts, including the evolution of generative NLP models and language modeling techniques, in Chapter 2. We then follow a structured approach to

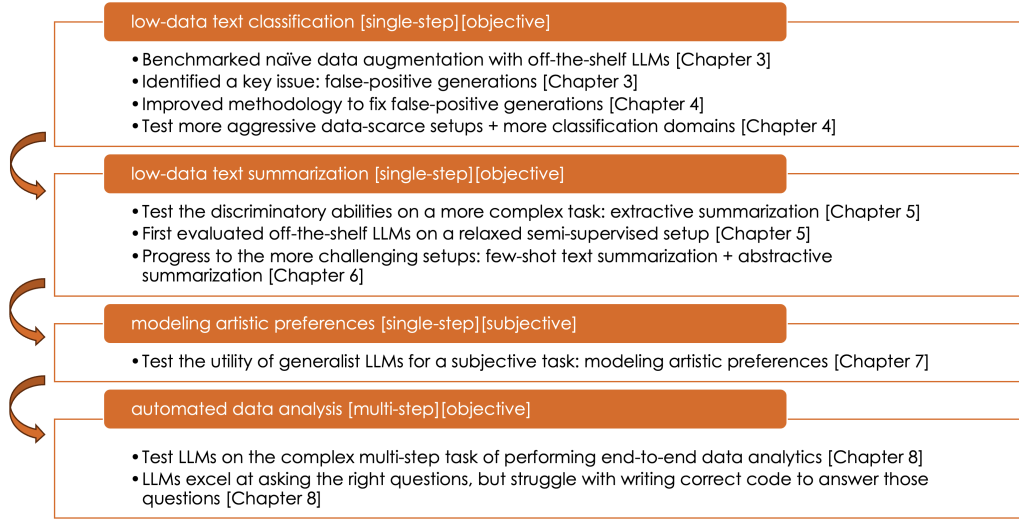


Figure 1.1: Progression of topics in this thesis.

evaluate the utility of LLMs for numerous tasks. Our tasks are categorized on two axes: *single-step* v/s *multi-step* tasks and *objective* v/s *subjective* tasks. As described earlier, we define single-step tasks as those that require an LLM to perform a singular task at a time, such as classifying whether a sentence is a question, and we define multi-step tasks as tasks that require performing multiple tasks in tandem to achieve a singular goal. For instance, performing exploratory data analysis on a dataset might require performing the following tasks: identify relevant questions to explore in the dataset, write code to generative visualizations that answer those questions, and then interpret the visualizations to derive meaningful insights. We do not count lower-level tasks, such as tokenization, computing embeddings, and attention matrices, as individual tasks in the single- and multi-step setups. Notably, multi-step tasks are more challenging than single-step tasks as they require performing multiple tasks without compromising the efficacy of individual tasks. Next, we define objective tasks as those with clearly defined outcomes and a broad consensus on the correct answer. For example, in sentiment analysis, most people would agree on whether a review expresses a positive or negative sentiment. On the other hand, we define subjective tasks as those where the value of the target variable depends on personal opinions or perspectives, leading to variability in responses (for the same input) across different individuals. For instance, a single lyrical line might feel inspiring to one person but fail to resonate with another. We chose these axes as they encompass the diversity of tasks that LLMs are expected to perform in real-world applications, and they allow us to evaluate the capabilities and limitations of LLMs in different scenarios in a principled fashion.

Given the categorization of tasks, we start by evaluating the utility of LLMs for a single-step, objective task in Chapter 3, as it is the least complex category of task. Specifically, we explore the discriminatory setup of text classification, a fundamental NLP task, with a focus on a few-shot setup without many labeled examples, as we intend to simulate real-world scenarios where labeled data is scarce. Through this study, we aim to understand if LLMs, which have already been exposed to massive data during their pretraining, could be a viable solution to alleviate data scarcity found in real-life, practical scenarios. We benchmark a naive data augmentation method with off-the-shelf LLMs against existing approaches and identify its strengths and limitations. After confirming the potential of data augmentation with generalist LLMs, we progress to developing more advanced data augmentation strategies with LLMs in Chapter 4 to address limitations of a naive LLM-based method and handle more aggressive data-scarce setups, such as zero-shot text classification.

We then proceed to progress towards our second single-step task of text summarization. We chose text summarization as our second single-step task as it increases the complexity of the objective task from a discriminatory setup to a generative setup. Moreover, text summarization is a fundamental NLP task with a wide range of applications, from condensing news articles to generating abstracts for research papers; however, to avoid an abrupt increase in complexity from simple classification to abstractive summarization, we first test whether the discriminatory capabilities of the LLM translate well to the domain of summarization. Therefore, we work on *extractive* text summarization in Chapter 5, as it casts summarization as a classification problem, where the task is to predict whether or not a given sentence in the input document should be present in the output summary. More specifically, we work in the relaxed semi-supervised setting with access to a large unlabeled pool of examples in addition to a small labeled set. We confirm that LLMs can be effectively leveraged for extractive summarization and then progress to the more challenging setup of few-shot, *abstractive* summarization in Chapter 6, where we develop a novel LLM-based pipeline that performs data augmentation and knowledge distillation to achieve competitive performance in a few-shot setup.

After evaluating the utility of LLMs on multiple single-step, objective tasks, we further increase the complexity of the tasks by moving to subjective tasks. Specifically, we test the utility of generalist LLMs for modeling artistic preferences of creative individuals in Chapter 7 as it is an inherently subjective task requiring an understanding of the artistic preferences of different individuals. We conclude that while LLMs might not be able to discern between the likes and dislikes of artists, they can be effective in extracting key linguistic and poetic properties from text that can later be employed to infer artistic preferences among different individuals.

Upon evaluating the utility of LLMs for single-step, objective, and subjective tasks, we progress to our most complex setup: multi-step tasks. In Chapter 8, we evaluate the capabilities of LLMs for automated exploratory data analysis. We demonstrate that the abilities of LLMs for single-step tasks can be translated to most multi-step tasks by appropriately contextualizing the LLM. Figure 1.1 summarizes the progression of topics in this thesis. We discuss the broader implications of our findings with prospective future directions in Chapter 9 before presenting our concluding remarks in Chapter 10.

Chapter 2

Background and Related Work

In this chapter, we cover the foundational topics relevant to the subsequent chapters. Section 2.1 provides an overview of generative models in natural language processing (NLP), highlighting their applications and significance. We discuss the evolution of these models, from traditional statistical approaches to modern deep learning architectures, and their impact on various NLP tasks. Finally, we explore the integration of generative and discriminative approaches in NLP, focusing on techniques like in-context learning, few-shot learning, fine-tuning, and retrieval-augmented generation (RAG). These hybrid models leverage the strengths of both paradigms to enhance text generation and classification tasks, enabling more nuanced and contextually aware applications. By combining generative and discriminative methodologies, researchers have achieved significant advancements in NLP, paving the way for more adaptable, intelligent, and efficient language models.

The second part of this chapter, Section 2.2, presents an overview of related work in the fields of few-shot text classification, text summarization, artistic preference modeling, and automated data analysis (Schick and Schütze, 2021; Alex et al., 2021; Bragg et al., 2021). We discuss key contributions and advancements in each domain, highlighting the challenges, methodologies, and outcomes of existing research. By examining prior work in these areas, we aim to situate our contributions and identify gaps in the current literature that our research seeks to address. This review of related work provides a foundation for understanding the state of the art in each field and sets the stage for our novel approaches and methodologies presented in the subsequent chapters.

2.1 Evolution of Generative Models in NLP

Generative models in natural language processing (NLP) are designed to learn the underlying distribution of a language and generate new, coherent text samples based on this learned distribution. These models predict the likelihood of sequences of words or phrases, enabling the generation of text that closely resembles human language. The following subsections provide an overview of generative models in NLP.

2.1.1 n -gram Models

n -gram models (Jurafsky and Martin, 2000) are one of the earliest generative models used in NLP. These models rely on a Markov assumption of order $n - 1$, meaning that the probability of occurrence of a word depends on the previous $n - 1$ words in the sequence. By estimating the conditional probability of each word given its context, n -gram models can generate text by sampling words based on their likelihood in the training data. Formally, an n -gram model can be described as follows:

$$P(w_k | w_1, w_2, \dots, w_{k-1}) \approx P(w_k | w_{k-(n-1)}, w_{k-(n-2)}, \dots, w_{k-1}) \quad (2.1)$$

where w_k represents the k -th word in the sequence. Notably, the probabilities are estimated based on the frequency of word sequences (or n -grams) in a training corpus, allowing the model to capture patterns and dependencies in the data. As a working example, let us illustrate how a bigram model predicts the probability of the word “mat” in the context of the sentence “The cat sat on the mat.” Substituting $n = 2$ in Equation 2.1, the bigram model estimates the conditional probability $P(w_k | w_1, w_2, \dots, w_{k-1}) \approx P(w_k | w_{k-1}) = P(\text{mat} | \text{the})$. Specifically, the probabilities are based on the frequency of the bigram “the mat” in the training data. For simplification, consider the following simplified, hypothetical training corpus:

“The cat sat on the mat. The dog sat on the rug. The cat chased the mouse.
The dog chased the ball.”

From the training corpus, we now extract all bigrams (pairs of consecutive words) and count their occurrences. Below is the list of extracted bigrams, along with their frequencies:

Bigram	Frequency
the cat	2
cat sat	1
sat on	2
on the	2
the mat	1
the dog	2
dog sat	1
sat on	2
on the	2
the rug	1
the cat	2
cat chased	1
chased the	2
the mouse	1
the dog	2
dog chased	1
chased the	2
the ball	1

Next, we compute the unigram probabilities for the training corpus, which is shown in the Table below.

Unigram	Frequency
the	8
cat	2
sat	2
on	2
mat	1
dog	2
chased	2
mouse	1
rug	1
ball	1

Finally, we compute the conditional probability $P(\text{mat} \mid \text{the})$ using the formula:

$$P(\text{mat} \mid \text{the}) = \frac{\text{Count}(\text{the mat})}{\text{Count}(\text{the})}$$

From the tables above:

$$\text{Count}(\text{the mat}) = 1$$

$$\text{Count}(\text{the}) = 8$$

Plugging in the values:

$$P(\text{mat} \mid \text{the}) = \frac{1}{8} = 0.125$$

The probability $P(\text{mat} \mid \text{the}) = 0.125$ indicates that, given the word “the,” there is a 12.5% chance that the next word is “mat” based on the training corpus. This probability helps the bigram model predict subsequent words in a sentence by evaluating the likelihood of each possible word following the current word. Similarly, we can compute the probability of “mat” using a trigram model ($n = 3$), where the probability of “mat” depends on the previous two words in the sequence, as follows:

$$P(\text{mat} \mid \text{on the}) = \frac{\text{Count}(\text{on the mat})}{\text{Count}(\text{on the})}$$

Overall, while n -gram models are simple to apply on observable word sequences and computationally efficient, they often struggle with capturing long-range dependencies and complex language structures due to their limited context window; however, despite their limitations, n -gram models have been foundational in NLP and have paved the way for more sophisticated generative models.

2.1.2 Hidden Markov Models (HMMs)

Hidden Markov Models (HMMs) ([Rabiner, 1989](#)), rely on more sophisticated statistical approaches to model the probability of word sequences. HMMs have two fundamental concepts: *states* and *symbols*. *States* represent the underlying, unobservable conditions or phases of the system being modeled. These states are considered “hidden” because they are not directly accessible or observable during the model’s operation. Instead, each

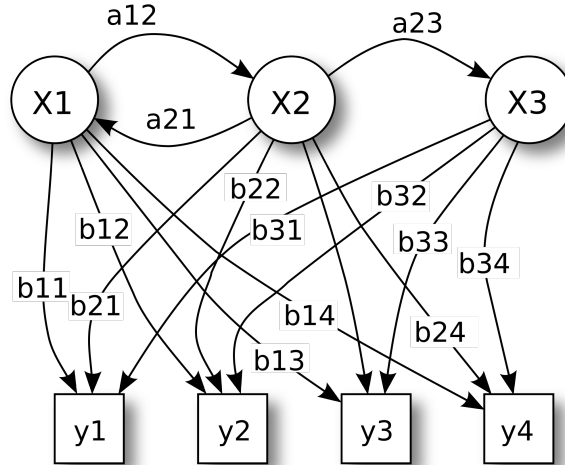


Figure 2.1: Hidden Markov Model, and its probabilistic parameters (example). X – states; y – possible observations; a – state transition probabilities; b – output probabilities. Source — [Wikipedia](#).

state generates observable outputs, allowing the model to infer the sequence of states based on the observed data. For example, in part-of-speech tagging, the hidden states could correspond to grammatical categories such as nouns, verbs, adjectives, etc., while the observable symbols are the actual words in a sentence. *Symbols*, also known as observations or emissions, are the visible outputs generated by the hidden states. These symbols are the actual data points that can be observed and measured. In the HMM framework, each state has a probability distribution over the possible symbols it can emit. Continuing with the part-of-speech tagging example, the symbols would be the words themselves, such as “cat,” “sat,” “on,” and “mat,” which are emitted by the corresponding grammatical states. Given the two concepts of states and symbols, the core mechanism of an HMM involves transitions between hidden states and the generation of observable symbols. In HMMs, we are interested in computing two sets of probabilities:

- **Transition Probabilities:** The probabilities of moving from one hidden state to another. These probabilities capture the dynamics of how states evolve over time.
- **Emission Probabilities:** The probabilities of a hidden state emitting a particular symbol. These probabilities link the hidden states to the observable data.

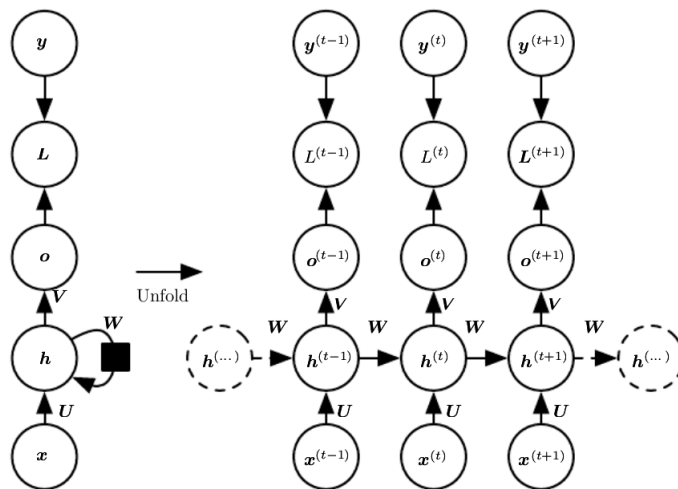


Figure 2.2: Recurrent Neural Network (RNN) architecture. Here, $x^{(t)}$ represents the input at time step t , $h^{(t)}$ denotes the hidden state at time step t , and $y^{(t)}$ is the output at time step t . The recurrent unit processes the input sequence iteratively, updating the hidden state based on the current input and the previous hidden state. Image source: [Goodfellow et al. \(2016\)](#)

By modeling the sequence of states and their corresponding emissions, HMMs can effectively capture the temporal dependencies and generate meaningful interpretations of the observed symbols. This dual structure allows HMMs to perform tasks such as sequence prediction, pattern recognition, and anomaly detection in various applications, including speech recognition, bioinformatics, and natural language processing. Figure 2.1 shows the structure of an HMM, including its probabilistic parameters, and to further illustrate the working of HMMs, we provide a worked-out example in Appendix A.1 to predict the probability of the word “mat” in the sequence “The cat sat on the mat” and its part-of-speech tag.

Despite their effectiveness in modeling sequential data, HMMs are limited by their Markov assumption and struggle with capturing long-range dependencies and complex linguistic patterns. Nevertheless, HMMs have been instrumental in advancing generative modeling techniques and have laid the groundwork for more sophisticated models like recurrent neural networks (RNNs) and modern Transformer-based architectures.

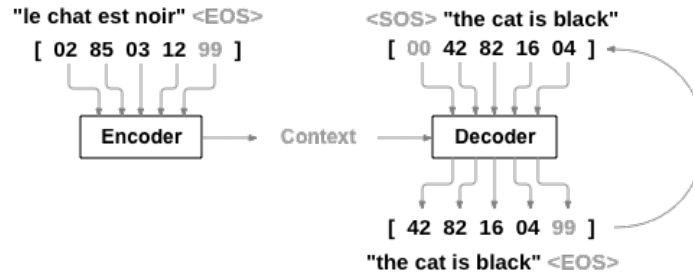


Figure 2.3: Sequence-to-Sequence (Seq2Seq) model architecture. The encoder processes the input sequence and generates a fixed-length vector representation, which is then decoded by the decoder to produce the output sequence. The model learns to map input sequences to output sequences, making it suitable for tasks like machine translation. In this example, the source sentence in French is being translated to English. Image source: [PyTorch Seq2Seq Tutorials \(2024\)](#)

2.1.3 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) ([Rumelhart et al., 1986](#)) are a class of neural networks designed to model sequential data by capturing temporal dependencies between elements in a sequence. Unlike traditional feedforward neural networks, which process input data in a fixed order, RNNs maintain an internal state that allows them to retain information about previous inputs as they process subsequent elements. This recurrent structure enables RNNs to model sequences of varying lengths and learn complex patterns in sequential data. This recurrent unit is the core component of an RNN, which processes input data at each time step and updates its hidden state based on the current input and the previous hidden state. This hidden state serves as a memory mechanism that encodes information about the sequence up to that point. By iteratively applying the recurrent unit to each element in the sequence, RNNs can capture long-range dependencies and context in the data. Figure 2.2 illustrates the architecture of an RNN.

RNNs have been widely used in natural language processing tasks such as language modeling, machine translation, and sentiment analysis, where the sequential nature of text data makes them well-suited for capturing linguistic structures and temporal dependencies. Although RNNs have been effective in modeling sequential data, they suffer from challenges like vanishing gradients, which limit their ability to capture long-range dependencies in sequences. To address these limitations, more advanced architectures like Long Short-Term Memory (LSTM) networks ([Hochreiter and Schmidhuber, 1997](#)) and Gated Recurrent

Units (GRUs) (Cho, 2014) have been developed, offering improved memory and gradient flow in recurrent networks.

A typical generative model based on RNNs is the Sequence-to-Sequence (Seq2Seq) model (Sutskever et al., 2014), which consists of an encoder-decoder architecture that maps an input sequence to an output sequence. The encoder processes the input sequence and encodes it into a fixed-length vector representation (also known as latent vector), which is then decoded by the decoder to generate the output sequence. Seq2Seq models have been instrumental in tasks like machine translation, text summarization, and dialogue generation, where the model learns to generate coherent and contextually relevant text based on the input sequence. Figure 2.3 illustrates the architecture of a Seq2Seq model, where the source sentence in French is being translated into English. Seq2Seq models have significantly advanced the field of generative modeling by enabling the generation of complex sequences and capturing semantic relationships between input and output data. However, they often struggle to obtain a continuous representation of the input sequence. This limitation led to the development of models like the Variational Autoencoder (VAE) for text (Kingma and Welling, 2013; Bowman et al., 2015), which regularize the latent space to ensure it is both continuous and smooth. Specifically, a variational autoencoder (VAE) is a stochastic neural generative model that consists of an encoder-decoder architecture. The key idea is to learn an approximate posterior distribution $q_\phi(\mathbf{z}|x)$ that closely resembles the true posterior $p_\theta(\mathbf{z}|x)$, which is generally intractable. The VAE objective minimizes the evidence lower bound (ELBO), which has the following form:

$$\text{ELBO} = E_{\mathbf{z} \sim q_\phi} [\log p_\theta(x|\mathbf{z})] - \text{KL}(q_\phi(\mathbf{z}|x) || p_\theta(\mathbf{z})), \quad (2.2)$$

where x is the input, $\mathbf{z}$ is a latent variable, $q_\phi(\mathbf{z}|x)$ is the approximate posterior distribution, and $p_\theta(\mathbf{z})$ is the prior. The KL divergence term, $\text{KL}(q_\phi(\mathbf{z}|x) || p_\theta(\mathbf{z}))$, encourages the approximate posterior to align with the prior, which is usually chosen as a standard Gaussian. Notably, the parameters ϕ and θ are jointly learned. VAE improves over a vanilla Seq2Seq by learning a continuous latent space as opposed to a deterministic input-output mapping; this allows us to sample diverse outputs from the model at inference time. Figure 2.4 illustrates the difference between the latent spaces learned by a deterministic autoencoder (using vanilla seq2seq architecture) and a variational autoencoder.

2.1.4 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) are a class of generative models that leverage a game-theoretic framework to learn the underlying distribution of

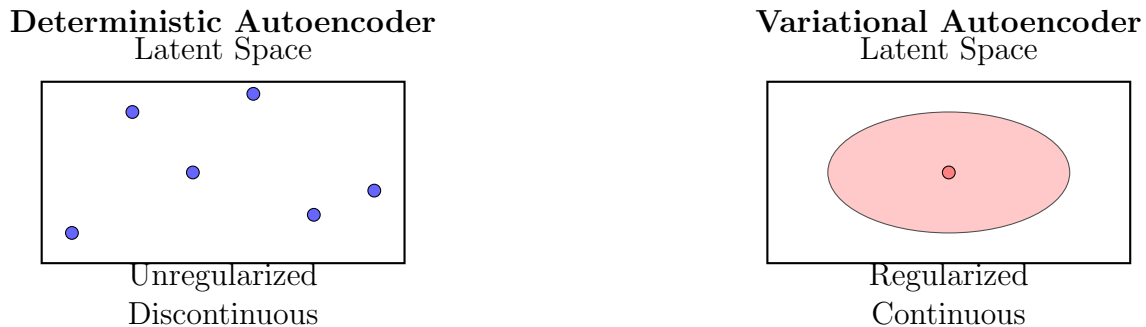


Figure 2.4: Comparison of latent spaces between a deterministic autoencoder and a variational autoencoder (VAE). The VAE learns a regularized, continuous latent space (right) by imposing a standard normal Gaussian prior, while the deterministic autoencoder learns an unregularized, potentially discontinuous latent space (left). **Note:** the points in the latent space are illustrative and do not represent actual data.

a dataset and generate new samples. GANs consist of two neural networks: a generator and a discriminator. The generator learns to produce synthetic data samples, while the discriminator learns to distinguish between real and generated samples. The two networks are trained simultaneously in a minimax game, where the generator aims to generate realistic samples that fool the discriminator, while the discriminator aims to correctly classify real and generated samples. Formally, GAN optimizes a minimax objective given by the following equation:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))], \quad (2.3)$$

where x is an input sample, z is a latent variable, and $p_{data}(x)$ is the true data distribution we wish the generator G to mimic.

Although GANs were originally proposed for image generation tasks, they have been successfully applied for some NLP use cases like dialog modeling (Khan et al., 2020), multi-modal fusion (Sahu and Vechtomova, 2021), and lyrics generation (Vechtomova and Sahu, 2023). Figure 2.5 illustrates the architecture of a GAN for text generation, where the generator produces text from random noise, and the discriminator distinguishes between real and generated text samples. Typically, the discriminator is a binary classifier that outputs a probability score indicating the likelihood of the input being real or fake, whereas the generator architecture can vary based on the task. For instance, in text generation, the generator can be an RNN or LSTM model that learns to generate coherent text samples from random noise. Despite the success of GANs for image tasks, it has been challenging

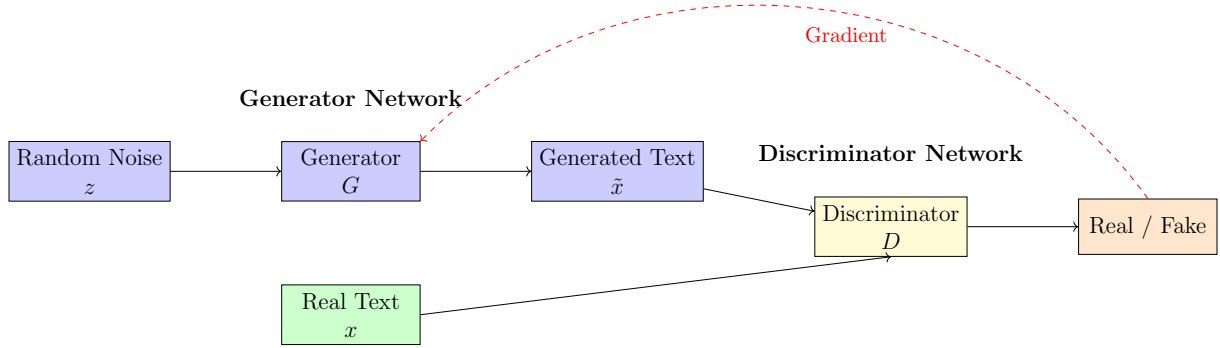


Figure 2.5: GAN architecture for text generation. The generator G generates text $\tilde{x}$ from random noise z , attempting to fool the discriminator D , which tries to distinguish between real text x and generated text $\tilde{x}$. The red dashed arrow indicates the gradient flow during training.

to widely employ them for text generation due to the discrete nature of text data and the difficulty of modeling language semantics.

2.1.5 Transformer-based Models

Transformer-based models (Vaswani et al., 2017) have revolutionized the field of natural language processing by introducing a novel architecture based on self-attention mechanisms. Transformers are designed to capture long-range dependencies in sequential data and have achieved state-of-the-art performance on various NLP tasks. The core components of the Transformer model are self-attention layers, which allow the model to focus on different parts of the input sequence simultaneously. This mechanism enables Transformers to capture complex linguistic patterns, semantic relationships, and context in text data. Figure 2.6a illustrates the architecture of a Transformer model, which consists of an encoder-decoder architecture with multiple layers of self-attention and feedforward neural networks.

The self-attention mechanism in Transformers computes attention weights for each input token based on its relationship with other tokens in the sequence. This attention mechanism allows the model to assign different importance to different parts of the input sequence, capturing dependencies and relationships between words effectively. Formally, the self-attention mechanism computes the attention weights for a query vector $\mathbf{q}$ based

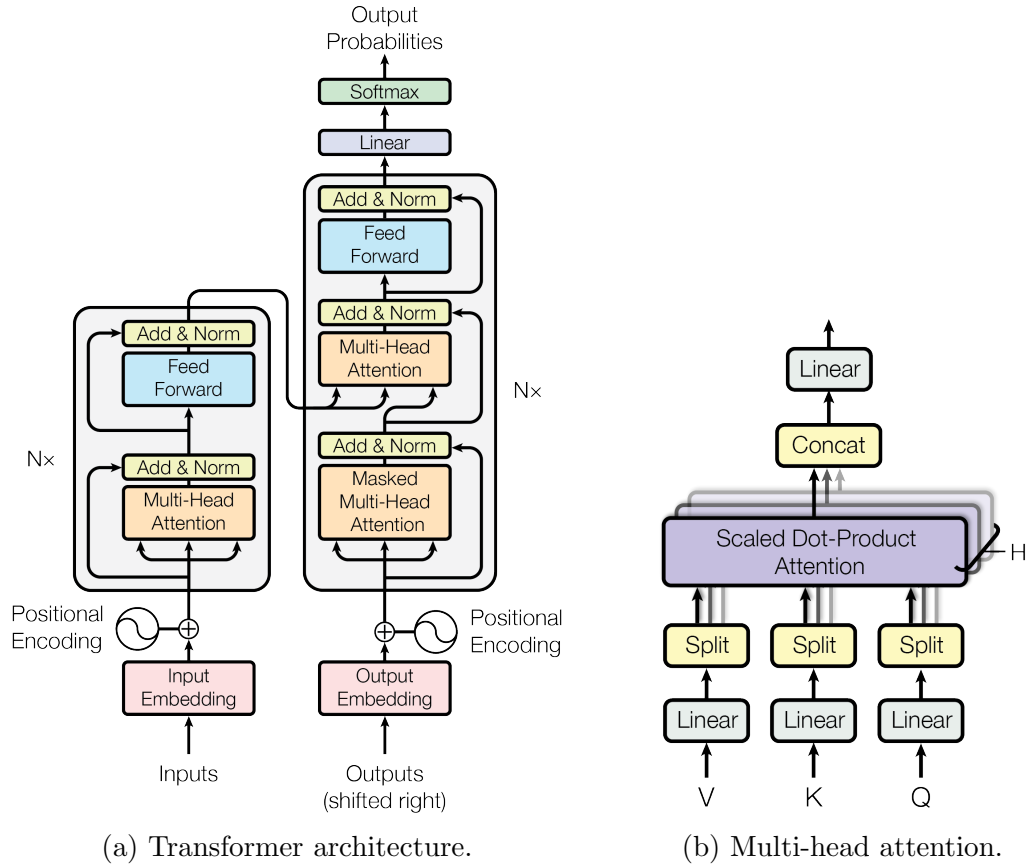


Figure 2.6: Transformer architecture and multi-head attention mechanism. The Transformer model consists of an encoder-decoder architecture with self-attention layers. The multi-head attention mechanism is a key component of the Transformer, allowing the model to focus on different parts of the input sequence simultaneously. Image source: (Vaswani et al., 2017).

on key vectors $\mathbf{k}$ and value vectors $\mathbf{v}$ as follows:

$$\text{Attention}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = \text{softmax} \left(\frac{\mathbf{q}\mathbf{k}^T}{\sqrt{d_k}} \right) \mathbf{v}, \quad (2.4)$$

where d_k is the dimensionality of the key vectors. The attention weights are computed by taking the dot product of the query and key vectors, scaling the result by $\sqrt{d_k}$, applying a softmax function to obtain a probability distribution, and then multiplying the values by the attention weights to get the output. The self-attention mechanism allows the model to capture dependencies between tokens in the input sequence and learn contextual representations that are crucial for various NLP tasks. It is an improvement over traditional attention mechanisms, as it enables the model to consider all tokens in the sequence simultaneously rather than attending to one token at a time. To understand the self-attention mechanism, consider the following example: given a sentence like “The cat sat on the mat,” the self-attention mechanism in a Transformer model would compute attention weights for each word in the sentence based on its relationship with other words. For instance, the word “cat” might receive higher attention weights from words like “the” and “sat,” indicating its importance in the context of the sentence. We include a worked-out example of the self-attention mechanism as well in Appendix A.2 to further illustrate the concept.

Multiple self-attention blocks stacked together make up the multi-head attention block, which enables the model to jointly attend to information from different representation subspaces. Figure 2.6b illustrates the multi-head attention mechanism, where the input is projected into multiple subspaces, and attention is computed independently for each subspace. This mechanism enhances the model’s ability to capture diverse patterns and relationships in the data, leading to improved performance on various NLP tasks.

Pre-training of Transformers. Since Transformers can process all tokens in parallel, they are computationally efficient and can be trained on large-scale datasets. Typically, they are pre-trained on vast amounts of text data using unsupervised learning objectives, such as masked language modeling (MLM) and next sentence prediction (NSP). The following subsections provide an overview of pre-training objectives used in Transformer models.

Language Modeling. Language modeling is a fundamental pre-training objective for NLP, where any model that learns to predict the next word in a sequence based on the context provided by the preceding words is a language model. For instance, n -gram models, RNNs, and LSTMs can be trained as language models by predicting the next word

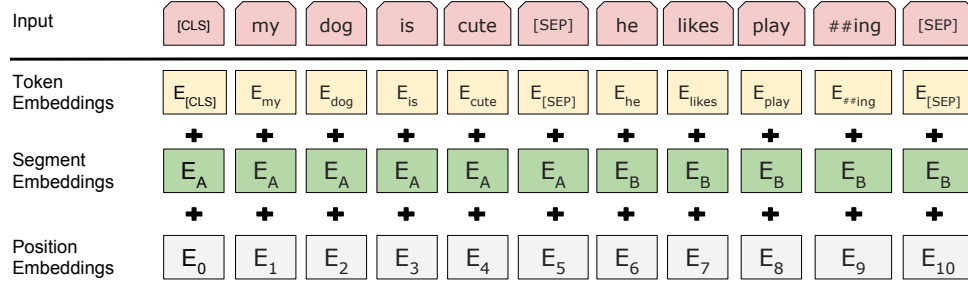


Figure 2.7: Illustration of the Next Sentence Prediction (NSP) task in BERT. The model is trained to predict whether the second sentence follows the first sentence in the original text. The input sentences are separated by a special [SEP] token, and a [CLS] token is added to the beginning of the first sentence. The [CLS] token is used as an aggregate representation of the input sequence, which is then used to predict whether the second sentence is a continuation of the first sentence. Image source: [Devlin et al. \(2019b\)](#).

in a sentence given the previous words. Formally, language modeling can be defined as predicting the probability of a word w_t given the context of the preceding words in the sequence, as shown in Equation 2.5.

$$p(x) = \prod_{t=1}^T P(w_t | w_1, w_2, \dots, w_{t-1}), \quad (2.5)$$

where w_t represents the t -th word in the sequence, and T is the total number of words in the sequence. In the context of Transformers, language modeling is also referred to as autoregressive language modeling or causal language modeling (CLM), where the model learns to predict the next token in the sequence based on the preceding tokens. Some notable language models trained on CLM include the GPT family of language models ([Radford et al., 2019](#); [Brown et al., 2020](#)).

Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). In addition to language modeling, Transformers are often pre-trained using masked language modeling (MLM) and next-sentence prediction (NSP) objectives. In MLM, a certain percentage of the input tokens are randomly masked and replaced with a special [MASK] token, and the model is trained to recover the original tokens based on the context provided by the unmasked, surrounding words. This objective encourages the model to learn bidirectional representations of the input sequence and capture dependencies between tokens in both directions.

In NSP, the model is provided with pairs of sentences, and it learns to predict whether the second sentence follows the first sentence in the original text. Specifically, the two sentences are separated by a special [SEP] token, and a special [CLS] token is added to the beginning of the first sentence. The CLS token is used as an aggregate representation of the entire input sequence (comprising of the two sentences), which is then used to predict whether the second sentence is a continuation of the first sentence.

By pre-training on large-scale text corpora using objectives like MLM and NSP, Transformers can learn rich representations of language and semantics, which can be fine-tuned on downstream tasks with limited labeled data. Some notable pre-trained Transformer models trained on MLM and NSP objectives include BERT (Devlin et al., 2019b), RoBERTa (Liu, 2019), and T5 (Raffel et al., 2019).

Fine-tuning of Transformers. Pre-training Transformers on a large corpus of text data allows them to learn general language representations that capture syntactic, semantic, and contextual information. Additionally, they also gain world knowledge that could be leveraged for a wide range of downstream tasks; however, to adapt these pre-trained models to specific tasks or domains, they need to be fine-tuned on task-specific datasets. For instance, a pre-trained Transformer model like BERT can be fine-tuned on a sentiment analysis dataset to predict the sentiment of movie reviews or on a question-answering dataset to answer questions based on a given context. Fine-tuning involves updating the parameters of the pre-trained model on the task-specific dataset using supervised learning, where the model learns to make predictions based on the labeled data. Fine-tuning leverages the pre-trained representations learned during pre-training and adapts them to the target task, enabling the model to generalize well on new data, and it has been a key strategy for leveraging the knowledge of pre-trained Transformer models in various NLP applications, including text classification, named entity recognition, and machine translation. Figure 2.8 illustrates the process of fine-tuning a pre-trained BERT model on various downstream tasks, where for each task, the model is fine-tuned on task-specific datasets to adapt to the target domain. For instance, for natural language inference (NLI), the model is fine-tuned on a dataset of sentence pairs to predict whether one sentence entails, contradicts, or is neutral with respect to the other sentence; for sentiment analysis, the model is fine-tuned on a dataset of text samples with sentiment labels; for named entity recognition (NER), the model is trained to predict the named entities like person names, locations, and organizations in a given text; and for question answering (QA), the model is fine-tuned on a dataset of question-answer pairs to predict the start and end positions of the answer span in the input context. Figure 2.9 provides an overview of how the pre-trained BERT model is fine-tuned on various downstream tasks, showcasing the versatility

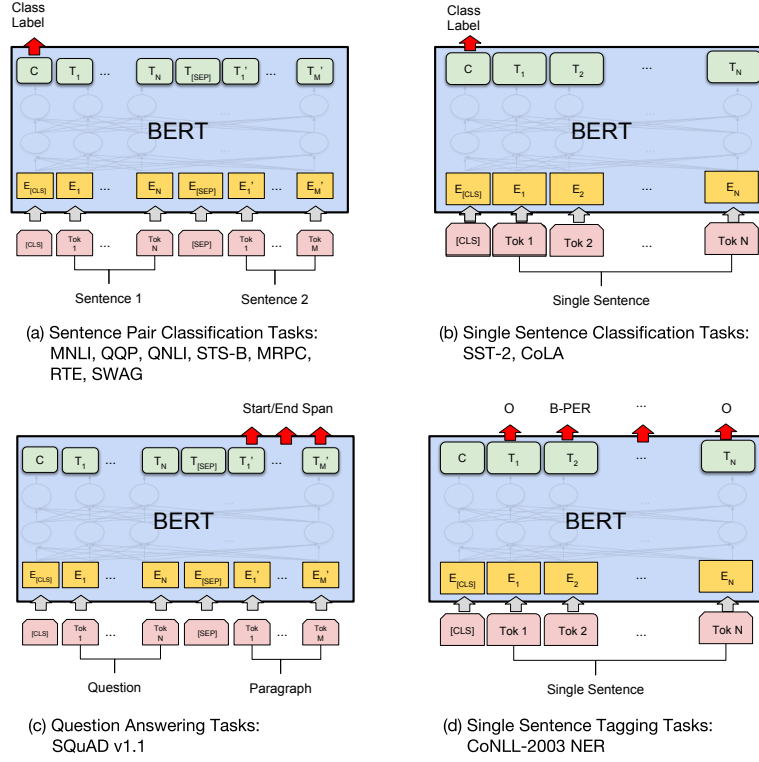


Figure 2.8: Fine-tuning BERT on four different downstream tasks: natural language inference, sentiment analysis, named entity recognition, and question answering. The pre-trained BERT model is fine-tuned on task-specific datasets to adapt to the target tasks. **Note:** for the NER task in (d), O denotes the token for non-entity words and B-PER denotes the beginning of a person entity. Image source: [Devlin et al. \(2019b\)](#).

and adaptability of pre-trained Transformer models in NLP applications.

During the initial boom of Transformer models, the pre-training + fine-tuning strategy was instrumental in achieving state-of-the-art performance on various NLP tasks; however, with the rapid rate of development in hardware technology and training algorithms, we are currently witnessing a shift towards training even larger models from scratch. This has led to the emergence of models like GPT-3 (estd. 175B parameters), GPT-4 (est 1.8T parameters), LLaMA-3 (70B parameters), LLaMA-3.1 (405B parameters), Claude-3.5-Sonnet, and others, which have billions of parameters and have demonstrated superior performance on a wide range of challenging NLP benchmarks *without* any task-specific fine-tuning ([Rein et al., 2023](#); [Hendrycks et al., 2020](#); [Chen et al., 2021a](#); [Srivastava et al.,](#)

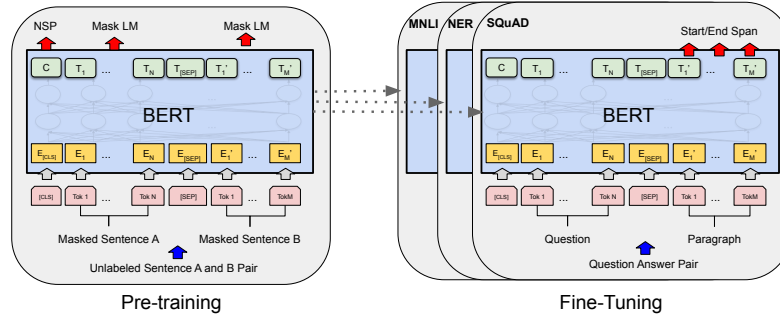


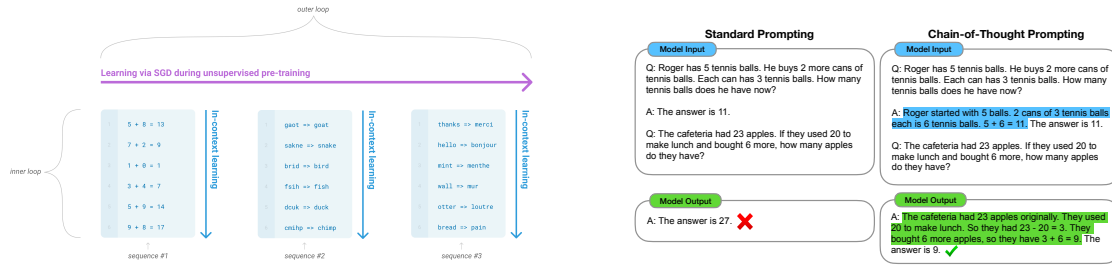
Figure 2.9: Illustration of how the pre-trained BERT model (on left) is fine-tuned on various downstream tasks (on right). Image source: [Devlin et al. \(2019b\)](#).

2022; Suzgun et al., 2022). We refer to such models as *generalist LLMs* as they perform well across a wide range of tasks without any task-specific fine-tuning.

The rise of generalist LLMs has opened up new possibilities for NLP research and applications, enabling the development of more versatile and adaptable models. As these models become more and more capable, we are experiencing a paradigm shift towards new techniques and methodologies to leverage the capabilities of these large-scale, generalist models effectively. For instance, techniques like prompt engineering and retrieval augmented generation (Lewis et al., 2020b) have been proposed to enhance the performance and generalization capabilities of generalist LLMs. We provide a brief overview of these techniques in the following subsections.

Prompt Engineering

Prompt engineering is a technique that involves designing task-specific prompts or instructions and then passing them to an LLM to guide its generation process. The prompts provide the model with context and constraints to generate the desired output, enabling it to perform specific tasks or answer questions effectively. Prompt engineering has been shown to improve the performance of generalist LLMs on a wide range of tasks, including question answering, text generation, and language understanding. By carefully designing prompts that provide the model with relevant information and constraints, researchers can tailor the behavior of the LLM to suit the target task or application. For instance, in sentiment analysis, a prompt like “classify the sentiment of the following text as positive, negative, or neutral” can guide the model to predict the sentiment of a given text accurately.



The three settings we explore for in-context learning

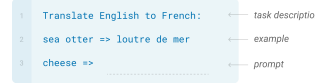
Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.



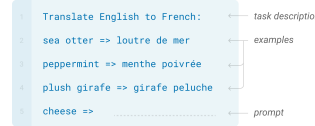
One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.



Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.



Traditional fine-tuning (not used for GPT-3)

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.



(c) Few-shot learning example.

Figure 2.10: Illustration of prompt engineering techniques: (a) In-context learning (ICL) involves providing the model with additional context to improve its performance on a specific task. (b) Few-shot learning involves training the model on a small number of examples to adapt it to new tasks. (c) Chain-of-thought prompting requires the model to provide its reasoning process before arriving at a final answer. Image source: [Brown et al. \(2020\)](#); [Wei et al. \(2022\)](#).

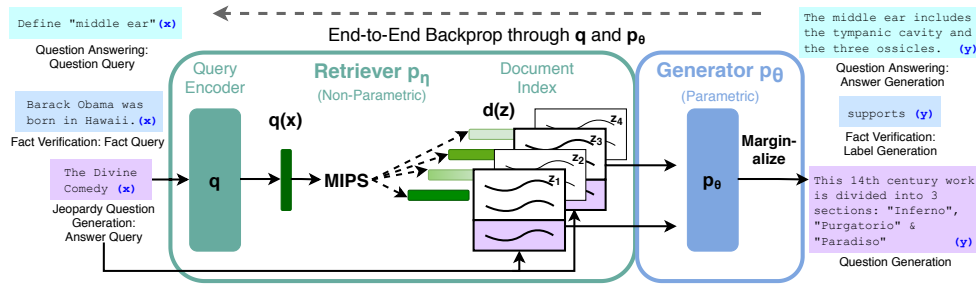


Figure 2.11: Retrieval-Augmented Generation (RAG) architecture. The model combines a generative model with a retrieval component that retrieves relevant information from a knowledge base to guide the generation process. The retrieval component provides context and information to the generative model, enabling it to generate more accurate responses. Image source: [Lewis et al. \(2020b\)](#).

Prompt engineering encompasses a wide range of modern-day NLP solutions, such as zero-shot learning, few-shot learning, and chain-of-thought (CoT) reasoning ([Brown et al., 2020](#); [Wei et al., 2022](#)). *Few-shot learning* is a machine learning paradigm that focuses on performing tasks with a very limited number of labeled examples, typically ranging from one to a few per class ([Brown et al., 2020](#)). Few-shot learning has enabled models to adapt to new tasks such as text classification, translation, and question answering with significantly fewer annotated samples ([Gao et al., 2020](#)). This approach not only reduces the need for extensive data collection and labeling but also enhances the flexibility and scalability of NLP systems across diverse domains ([Qiu et al., 2020](#)). Consequently, few-shot learning has become a pivotal strategy in developing more efficient and adaptable NLP applications, especially in scenarios where data scarcity is a major constraint. *Zero-shot learning* is a special case of few-shot learning where the LLM is tasked to perform a task without any labeled examples in the prompt. Lastly, *Chain-of-thought reasoning* refers to a specialized form of prompt engineering where the LLM is instructed to provide its reasoning process before arriving at a final answer. In addition to enhancing the interpretability and trustworthiness of LLMs, CoT reasoning has also been shown to improve the performance of LLMs on downstream tasks ([Wei et al., 2022](#)). Figure 2.10 illustrates the different types of prompt-based methods.

Retrieval-Augmented Generation (RAG)

Retrieval-augmented generation (RAG) ([Lewis et al., 2020b](#)) combines the strengths of generative models and retrieval-based methods to solve knowledge-intensive tasks, such as

open-domain question answering (Min et al., 2021; Adlakha et al., 2022). In RAG, a generative model, like a large-scale LLM, is augmented with a retrieval component that retrieves relevant information from a knowledge base or external corpus to guide the generation process. The retrieval component provides the generative model with context and information that can help it generate more accurate responses. Figure 2.11 illustrates the architecture of a retrieval-augmented generation model, where the generative model is combined with a retrieval component to enhance its performance on a question-answering task. In the next section, we discuss relevant work in the field of few-shot text classification, text summarization, artistic preference modeling, and automated data analysis, highlighting key contributions and advancements in each domain.

2.2 Related Work

We now provide an overview of related work for the tasks considered in this thesis.

2.2.1 Text Classification

Text classification is a fundamental task in NLP where, given a natural language utterance, the goal is to assign it to one or more predefined categories or classes. Text classification has numerous applications in day-to-day life, such as sentiment analysis, spam detection, topic categorization, and intent classification, among others. Traditional text classification methods relied on feature engineering and shallow machine learning algorithms like Support Vector Machines (SVM) and Naive Bayes classifiers (Cortes and Vapnik, 1995; Lewis, 1998). However, with the advent of deep learning, neural network-based models have become the de facto standard for text classification tasks. Convolutional Neural Networks (CNNs) (Kim, 2014), Recurrent Neural Networks (RNNs) (Hochreiter and Schmidhuber, 1997), and Transformer models (Devlin et al., 2019a) have been widely used for text classification tasks, achieving state-of-the-art performance on various benchmarks. The most recent advancements in the development of large-scale LLMs, like GPT and LLaMA, have now enabled research in more challenging scenarios data-scarce scenarios, such as few-shot and zero-shot text classification. We focus our study on data-scarce scenarios to measure the utility of LLMs for the task. Specifically, we study the four text classification tasks: intent classification, question classification, and review analysis.

Intent classification is a key NLP task that involves identifying the underlying intention behind a user’s input, such as a query, command, or statement. This process is crucial

for applications like chatbots, virtual assistants, and customer service automation, where understanding user intent is essential for providing relevant and accurate responses. *Question classification* is another important NLP task that involves categorizing questions into predefined classes based on their semantic content. Question classification is essential for question-answering systems, information retrieval, and conversational agents, where understanding the type of question is crucial for providing accurate answers. Lastly, *review analysis* involves analyzing and categorizing user reviews based on their sentiment, opinion, or content. Review analysis is widely used in e-commerce, social media, and customer feedback analysis to understand user sentiment, identify trends, and improve products or services. Traditional text classification methods relied on rule-based systems or supervised machine learning algorithms that utilized handcrafted features (Jurafsky and Martin, 2000) to classify utterances into predefined categories. However, modern approaches like recurrent neural networks (RNNs), BERT, and GPT-3 benefit from their ability to learn rich contextual embeddings from large datasets and can effectively classify user intent.

Traditional Solutions. As the capabilities of LLMs improve, research is steadily gravitating towards data-scarce scenarios that are more reflective of real-world applications. Traditionally, data augmentation has been a widely employed NLP technique for autonomously obtaining more labeled data and reducing the financial and time costs associated with manual data labeling. For instance, simple edit-based methods such as Easy Data Augmentation (EDA) (Wei and Zou, 2019) perform random insertions, deletions, swaps, and synonym replacements on the original text to generate new training examples. On the other hand, in back-translation (Sennrich et al., 2016), a sentence is translated into another language and then back-translated to the original language to generate new training examples. Notably, this technique relies on the fact that there may be multiple valid translations for a given sentence, and the back-translation process can introduce diversity in the generated examples.

LLM-based Solutions. Some works finetune language models on a relatively small amount of data and then use them as data generators to synthesize new training examples. For instance, Kumar et al. (2019) evaluate different feature space data augmentation techniques, such as upsampling, perturbation, and linear interpolation, for varying levels of data scarcity (ranging from 5% data availability to 20%); however, their performance gains are minor compared to a no-augmentation setting. Kumar et al. (2020) consider 10-shot, 50-shot, and 100-shot text classification setups where they fine-tune pretrained language models like BERT, BART (Lewis et al., 2020a), and GPT-2 (Radford et al., 2019) as data

generators on k -shot data. Next, they condition the fine-tuned data generators to synthesize new training examples for individual classes in the dataset. Similarly, Lee et al. (2021) explore a *partially* few-shot scenario where they first obtain a data generator by fine-tuning a T5 model on a small amount of data to then generate novel labeled examples. This method improves over classical data augmentation techniques of easy data augmentation and back translation, but they work on the assumption that there is enough labeled data to fine-tune an LM in the first place. In contrast, we propose data augmentation strategies that sidestep LM-finetuning and leverage off-the-shelf LLMs to generate new training examples, making them more scalable and cost-effective. GPT3Mix (Yoo et al., 2021) is one of the earliest works using prompt-based data augmentation to generate new training examples for few-shot text classification tasks. Specifically, it lists the names of all possible classes (e.g., “The sentiment is one of ‘positive’ or ‘negative’”) in the prompt, and the LLM picks a random class from the list to generate a new labeled example. Although GPT3Mix is effective, it does not scale to intent classification setups, which often feature hundreds of intents, many of which have nuanced differences (see Section 3.3.3). Therefore, we choose different prompts that encourage the LM to extrapolate between some seed examples. Additionally, our methods can handle unfaithful GPT generations that have incorrect labels and instances where two classes have very subtle differences.

Some non-prompting-based approaches include Wei et al. (2021), who propose curriculum data augmentation, where they first train a model on limited k -shot data and then incrementally introduce augmented data as training progresses. They use triplet loss (Schroff et al., 2015), which minimizes the distance between data points with the same label and maximizes for differently labeled examples. Tunstall et al. (2022) propose SetFit that first fine-tunes sentence transformers on a small number of text pairs in a contrastive Siamese manner (Dong and Shen, 2018) and then generates rich text embeddings for training a classification head. Finally, Kim et al. (2021) propose LINDA, which is first trained on one million sentence pairs randomly drawn from English Wikipedia. Later, they use the Mixup algorithm Zhang et al. (2018) to interpolate between two English sentences of varying lengths. LINDA significantly improves the performance of a BERT_{base} model across multiple few-shot classification setups. In our work, we consider more extreme few-shot setups (2-shot, zero-shot) and perform more controlled interpolation that promotes the generation of examples near class boundaries. We present more details on our methods in Chapters 3 and 4.

Knowledge Distillation. One of our proposed augmentation methods (discussed in Chapter 4) also achieves a positive by-product of achieving knowledge distillation (Bucila et al., 2006; Hinton et al., 2015; West et al., 2022), which refers to training a smaller student

model to mimic the behavior of a larger teacher model. In particular, the objective function aims to match the output distribution of the student model to that of the teacher model. By doing so, knowledge of the teacher model is effectively distilled into a much smaller student model, allowing a similar level of performance as the bigger teacher model at a lower computational cost. We can describe the minimization objective for knowledge distillation as follows:

$$\mathcal{L}_{KD} = \min_{\theta_s} \mathbb{E}_{(x,y) \sim \mathcal{D}} [\text{KL} (P_{\text{teacher}}(y|x) || P_{\text{student}}(y|x))], \quad (2.6)$$

where $P_{\text{teacher}}(y|x)$ and $P_{\text{student}}(y|x)$ are the output distributions of the teacher and student models, respectively, KL denotes the Kullback-Leibler divergence (we can use other divergence measure as well), $\mathcal{D}$ is the dataset, and θ_s are the parameters of the student model. [Shridhar et al. \(2022\)](#) distill a GPT3 (6B) model into a GPT-2 model for a Chain-of-thought (CoT) reasoning task, and [Liang et al. \(2021\)](#) propose MixKD to encourage the student model to mimic the teacher’s behavior on not only the available training examples but also on interpolated examples. Finally, [Sun et al. \(2020\)](#) distill knowledge through intermediate layers of the teacher via a contrastive objective. In comparison, our approach allows knowledge distillation of a massive teacher model like GPT3.5 (175B parameters) into significantly smaller student models like DistilBERT and BERT (67M and 110M parameters, respectively).

2.2.2 Text Summarization

Text Summarization refers to the process of generating a concise and coherent summary of a given text document while preserving its key information and meaning. Text summarization has numerous applications in various domains, such as news aggregation, document summarization, and summarization of web search results. There are two main approaches to text summarization: extractive and abstractive summarization. Extractive summarization involves selecting and rearranging sentences or phrases from the source text to create a summary, while abstractive summarization involves generating new sentences that capture the essence of the original text. Traditional approaches to extractive text summarization include classification-based approaches, where sentences are ranked based on their relevance to the document, and graph-based methods, where sentences are represented as nodes in a graph, and the most important sentences are selected based on their centrality in the graph. More recent approaches for extractive summarization include neural network-based models like BERT that have better semantic understanding to model the relative

importance of sentences in the source text. Abstractive summarization methods, on the other hand, use sequence-to-sequence models like BART and T5 to generate summaries by learning to paraphrase and rewrite the input text. These models have shown significant improvements in the quality and coherence of generated summaries, making them suitable for a wide range of summarization tasks. Additionally, their ability to generate human-like summaries with minimal input has made them central to applications in document summarization, news aggregation, and aiding research by distilling large amounts of information into digestible formats.

The earliest works on extractive text summarization use unsupervised and supervised algorithms to extract sentence and word-level features. Examples of unsupervised approaches include using graph-based ranking techniques like LexRank (Erkan and Radev, 2004) and external knowledge from Wikipedia (Sankarasubramaniam et al., 2014), and fuzzy-logic systems (Kyoomarsi et al., 2008; Suanmali et al., 2009). While these methods improve coherency and reduce redundancy, they rely on surface-level features and cannot handle dangling anaphoras. In contrast, our methods (discussed in Chapters 5 and 6) combine richer sentence representations of pretrained language models like BERT and DistilBERT (Sanh et al., 2019) with off-the-shelf LLMs like GPT-4 and LLaMA-3 that can aid in anaphora resolution (Shen and Elhoseiny, 2023). Prior supervised approaches use probabilistic support vector machines (Wong et al., 2008), Bayes rule (Neto et al., 2002), conditional random fields (Svore et al., 2007), and more recently, pretrained language models like BERT (Liu and Lapata, 2019; Pilault et al., 2020; Zhong et al., 2020) for the task. While supervised methods provide better representation for sentence selection, they need large human-annotated datasets for training, which can be financially expensive to gather. To address this problem, Zhuang et al. (2023) explore the semi-supervised setting and propose a novel objective function to improve the pretraining of BERT for dialog summarization. Their method significantly improves over the traditional pretraining process; however, it lacks flexibility and is heavily tailored for dialog summarization. In our experiments, we focus on the generalizability of the framework and develop a generic prompt-based strategy that can be easily adapted across different domains. Additionally, our method uses fewer labeled data samples and generates higher-quality pseudo-labels.

With the recent advancements in generative language modeling, a new body of literature using GPT-like models for summarization has emerged. For instance, Fabbri et al. (2020) use round-trip back-translation to improve BART’s abstractive summarization performance, whereas, Dou et al. (2021) propose GSum, a fully supervised transformer-based architecture that can use a guidance signal from an external source for improved abstractive text summarization. Zhang et al. (2023b) propose an extract-then-generate method using ChatGPT where they use in-context learning to generate extractive-summary-guided

abstractive summaries. However, since most of these works operate in a fully-supervised setting, the method suffers from scalability issues for large datasets. [Goyal et al. \(2022\)](#) employ zero-shot prompting on GPT-3 for open-ended news summarization and show that humans overwhelmingly prefer GPT-3 summaries over human summaries. [Pu and Demberg \(2023\)](#) use prompting on GPT-3 for controllable text summarization and show that while GPT-3 can follow simple constraints in the prompt, like length, it shows a noticeably lower degree of change in styles compared to human-written summaries. [Liu et al. \(2024a\)](#) and [Zhang et al. \(2024b\)](#) benchmark the zero-shot performance of LLMs on instruction-controlled summarization and news summarization. [Chintagunta et al. \(2021\)](#) use GPT-3 as a data annotator for 210-shot medical dialog summarization and show significant gains equivalent to using 6400 human-written labels. More recently, [Liu et al. \(2024b\)](#) fine-tuned BART on LLM-generated summaries instead of human-generated summaries and showed that LLMs are excellent references. Notably, these works prompt GPT-3 directly for summarization in their experiments. Except for the last two works, none of them use LLMs as data generators in data-scarce setups. Additionally, they all use a closed-source LLM in their experiments. [Mishra et al. \(2023\)](#) propose LLM pseudo-labeling for semi-supervised dialog summarization that aims to mitigate data-scarcity issues; however, in addition to operating in a semi-supervised setup, we also explore the more challenging few-shot summarization setup. We present more details on our methods for summarization in Chapters 5 and 6 that are more sample-efficient than existing literature in terms of labeled data requirements.

Similar to text classification, our methods achieve knowledge distillation as a by-product of text summarization as well. Recent work using LLMs as data generators for distilling a large teacher model’s knowledge into smaller models for training instruction-tuned models and chain-of-thought reasoning while reducing human annotation load ([Ho et al., 2023](#); [Shum et al., 2023](#); [Meng et al., 2023](#); [Liu et al., 2023a](#); [Peng et al., 2023](#)). [Bonifacio et al. \(2022\)](#) use few-shot prompting to construct training datasets with query-document pairs for information retrieval. Our approach combines the best prompting practices based on these works to generate diverse examples by mixing examples from different classes or groups; and specify concrete criteria when using LLMs for generation and evaluation as in [Pu and Demberg \(2023\)](#) and [Liu et al. \(2024b\)](#). Furthermore, we conduct extensive experiments to test the capabilities of an open-source LLM, LLaMA-3-70b, for few-shot text summarization, instead of using closed-source LLM like GPT-3 and GPT-4. Our method also evaluates our LLM-based DA strategy on extremely long documents, which has been overlooked by past studies.

2.2.3 Modeling Artistic Preferences

We explore the subjective task of modeling artistic preferences in the context of short lyric lines. Modeling artistic preferences involves understanding various linguistic and stylistic properties of lyric lines that can help in modeling the subjective tastes of different creative individuals. It is a challenging task that requires capturing the complex and nuanced aspects of creative works.

In the space of text evaluation, LLM-based metrics such as LLM-Rubric (Hashemi et al., 2024), GPTScore (Fu et al., 2024), and G-Eval (Liu et al., 2023b) have facilitated the evaluation of natural language text. For objective or well-defined tasks like text classification or text summarization, these metrics are far better aligned with human judgments than traditional NLP metrics like BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), or METEOR (Banerjee and Lavie, 2005); however, these LLM-based metrics primarily assess general language quality rather than creative attributes. There are also evaluation approaches that measure aspects like diversity and fluency but do not capture the multifaceted nature of creative expression. For instance, Hashimoto et al. (2019) proposed an evaluation framework balancing diversity and fluency, while Pillutla et al. (2024) introduced MAUVE, an automatic measure of how close generated text is to human text. Traditional approaches to measuring creativity relied on rule-based systems like Consensual Assessment Technique (CAT) (Amabile, 1982) that utilizes the collective judgment of experts to assess creativity, while the Torrance Test of Creative Thinking (TTCT) (Baer, 2011) measures divergent-thinking ability. Chakrabarty et al. (2024) extended these ideas to the Torrance Test of Creative Writing (TTCW) to evaluate creativity in outputs from both humans and large language models (LLMs); however, TTCT has faced criticism for not reflecting the diverse forms that creativity can take, and the other evaluation methods treat the generated text as final artifacts and might not be well-suited to AI-generated outputs intended to inspire artists in their initial phase of creative exploration. In our work, we develop multidimensional evaluation metrics that capture various linguistic and stylistic characteristics of generated text and investigate how these metrics can predict the personal preferences of individuals with diverse artistic skills and interests. We include more details on our methodology in Chapter 7.

2.2.4 Automated Data Analysis

Automated Data Analysis refers to the process of using computational tools and algorithms to analyze and extract insights from large datasets. Automated data analysis has numerous applications in various domains, such as business intelligence, healthcare, finance, and

scientific research. Traditional data analysis methods involve manual data processing, visualization, and statistical analysis, which can be time-consuming and error-prone. With this task, we slowly transition to measuring the utility of LLMs for more complex, real-world tasks requiring a wide variety of skills, such as autonomously handling diverse data sources, coming up with meaningful areas to explore in the dataset, applying complex data transformations, and providing actionable insights in real-time. Automated data analysis agents have the potential to revolutionize the way data is analyzed and interpreted, making it easier for non-experts to derive insights from complex datasets. We now provide an overview of the ongoing efforts in the field.

Text-to-Analytics Agents. The evolution of LLMs has shown advances in automated data analytics through their capabilities in zero-shot and few-shot learning, as well as their responsiveness to human instructions (OpenAI, 2022, 2023a). These models are adept at extracting actionable insights from extensive unstructured datasets (Gravitas, 2023). Chen et al. (2023b) explore the application of GPT variants within a data visualization context, highlighting the strengths and limitations of these models. More recent LLM-based data analysis agents include Code Interpreter (OpenAI, 2024) and Pandas Agent (LangChain, 2024) that are capable of processing multiple data formats and answering questions about them. Ma et al. (2023) propose InsightPilot, an advanced automated tool that leverages LLMs to enhance data exploration by automatically identifying goals and generating targeted intentional queries. Vacareanu et al. (2024) showed that LLMs can also perform regression tasks, enhancing their predictive abilities. Additional studies assess the data analysis capabilities of GPT-4 and propose an end-to-end framework for automating data processes (Cheng et al., 2023; Wang et al., 2023, 2024; Hong et al., 2024). We describe our proposed data analysis agent in Chapter 8 that is capable of performing end-to-end data analysis tasks, including data exploration, visualization, and insight generation and outperforms existing methods on a challenging benchmark.

LLM Evaluation Frameworks Existing LLM evaluation frameworks focus on handling structured outputs and predominantly rely on pre-formatted prompts to assess code completion (Wu et al., 2023; Zhang et al., 2023c, 2024a; Yao et al., 2023). While recent advancements have seen autonomous agents specializing in intricate data science tasks, including analysis, visualization, and modeling (Qian et al., 2023, 2024; Zhang et al., 2023c), evaluations for these methods often depend on extensive human effort (Cheng et al., 2023) or use more powerful LLMs to assess the output (Dubois et al., 2023; Belyi et al., 2024). Another work uses the “Capture the Flag” principle, where insights are planted into a dataset as flags to evaluate whether models can uncover them (Laradji et al., 2023). G-Eval has

also used to perform reference-free evaluation of free-form texts in terms of factuality and coherence. In this work, we propose an open-source variant of G-Eval to score how well the predicted insights are aligned with the ground-truth insight.

Chapter 3

Leveraging LLMs for Data Augmentation in Text Classification

In this chapter, we focus on a key challenge in text classification setups – creating task-oriented conversational agents that require extensive gathering and labeling of training data. Standard data-gathering options include manual authoring and crowd-sourcing. Unfortunately, both of these options are tedious and expensive. For instance, crowdsourcing is a financially expensive procedure incurring a documented high cost of human labor (Sheng et al., 2008; Rashtchian et al., 2010; Rajpurkar et al., 2016; Khot et al., 2018). *Data augmentation* is a widely used strategy to alleviate this problem of data acquisition, and with the recent surge in the development of generative large language models (LLMs), a large body of literature has emerged that employs LLMs to generate additional data for various tasks.

There are two particularly promising paradigms for data augmentation in natural language processing that use pretrained language models (LMs) (Peters et al., 2018; Devlin et al., 2019a). The first family of methods fine-tunes an LM on task-specific data and generates new examples using the fine-tuned LM (Wu et al., 2019; Kumar et al., 2019, 2020; Anaby-Tavor et al., 2020; Lee et al., 2021). A limitation of these methods is that, in a real-world scenario, task-specific data is scarce and fine-tuning an LM can quickly become the bottleneck. The second family of methods sidesteps this bottleneck by employing off-the-shelf pretrained LMs such as GPT-3 (Brown et al., 2020) to directly generate text without any task-specific fine-tuning. In particular, data generation by the GPT3Mix approach by (Yoo et al., 2021) boosts performance on multiple classification tasks; however, they only consider tasks with few (up to 6) classes and easy-to-grasp class boundaries (e.g., *positive* and *negative*).

Input Prompt:

The following sentences belong to the same category music_likeness:

Example 1: i like soft rock music
Example 2: current song rating three stars
Example 3: save this song as a favorite
Example 4: remind me that i like that song
Example 5: save my opinion on the currently
 playing song
Example 6: i love the song do you
Example 7: add the song to my favorites
Example 8: store opinion on song
Example 9: the song in background is cool
Example 10: i am the living blues
Example 11:

Completions:

i dislike classical music
she is a music lover
i am a lover of painting
this is the best song ever
video that looks like the other video
save preference on my profile
express negative opinion on the song
i am a great blues follower
the song is better than i thought
this song is also fun

Figure 3.1: **Generation Process.** Given a seed intent (here, music_likeness) and $K(=10)$ available examples for that intent, we construct a prompt following the shown template. Note that the last line of the prompt is incomplete (there is no newline character.) We then feed this prompt to a GPT-3 engine, which generates some completions of the prompt. In this example, **red text** denotes unfaithful examples and **blue text** denotes faithful examples. **Note:** For brevity, we only show ten generated sentences.

In this chapter, we study the applicability of massive off-the-shelf LMs, such as GPT-3 and GPT-J (Wang and Komatsuzaki, 2021) to perform effective data augmentation for intent classification (IC) tasks. In IC, the end goal is to predict a user’s intent given an utterance, i.e., what the user of a task-oriented chatbot wants to accomplish. Data augmentation for IC is particularly challenging because the generative model must distinguish between a large number (in practice up to several hundred) of fine-grained intents that

can be semantically very close to each other. Prior methods such as GPT3Mix prompt the model with the names of all classes as well as a few examples from randomly chosen classes. We test GPT3Mix for one and observe that such approaches are poorly suitable for intent classification tasks with tens or hundreds of possible intents. Instead, in this study, we use a simple prompt structure that focuses on a single seed intent (see Figure 3.1) as it combines the intent’s name and all available examples.

Our experiments in this chapter focus on few-shot IC on four prominent datasets: CLINC150 (Larson et al., 2019), HWU64 (Xingkun Liu and Rieser, 2019), Banking77 (Casanueva et al., 2020), and SNIPS (Coucke et al., 2018). We also consider a partial few-shot setup to compare to the Example Extrapolation (Ex2) approach by (Lee et al., 2021) who use a similar prompt but fine-tune the LM instead of using it as is. The main findings of our experiments are as follows: **(1)** GPT-generated samples boost classification accuracy when the considered intents are well-distinguished from each other (like in CLINC150, SNIPS). **(2)** On more granular datasets (namely HWU64 and Banking77), we find that GPT struggles to distinguish between different confounding intents. **(3)** A small-scale study to further understand this behavior suggests that GPT could be used as a classifier to filter out unfaithful examples and enhance the quality of the generated training set. Additionally, we investigate how valuable the generated data could be if relabeled by a human. Using an oracle model, we show that **(4)** the human labeling of GPT-generated examples can further improve the performance of intent classifiers and that **(5)** LM-generated data has a higher relabeling potential compared to edit-based augmentation techniques, such as Easy Data Augmentation (EDA) (Wei and Zou, 2019).

3.1 Methodology

We consider training an intent classifier, where an intent is a type of request that the conversational agent supports; e.g., the user may want to change the language of the conversation, play a song, transfer money between accounts, etc. However, collecting many example utterances that express the same intent is difficult and expensive. Therefore, we experiment with a straightforward method to augment the training data available for an intent: creating prompts from the available examples and feeding them to a large language model such as GPT-3 (Brown et al., 2020). Figure 3.1 illustrates the process of data generation for an intent with K available examples ¹.

¹Our code is available at: <https://github.com/ElementAI/data-augmentation-with-llms>

	CLINC150	SNIPS	HWU64	Banking77
domains	10	1	18	1
intents	150	7	64	77
train	15000	13084	8954*	9002*
examples	(100)			
val.	3000	700	1076*	1001*
examples	(100)			
test	4500	700	1076	3080
examples	(1000)			

Table 3.1: Statistics of the intent classification datasets that we use in our experiments. * indicates that we split the original data into training and validation instead of using a split provided by the dataset authors. For CLINC150, the number of out-of-scope examples in different data partitions is given in parentheses.

3.2 Experimental Setup

3.2.1 Datasets

We use four intent classification datasets in our experiments with varying levels of granularity among intents. CLINC150 (Larson et al., 2019), HWU64 (Xingkun Liu and Rieser, 2019) are multi-domain datasets, each covering a wide range of typical task-oriented chatbot domains, such as playing music and setting up alarms. Importantly, the CLINC150 task also contains examples of out-of-scope (OOS) utterances that do not correspond to any of CLINC’s 150 intents. Banking77 (Casanueva et al., 2020) is a single-domain dataset with very fine-grained banking-related intents. Finally, the SNIPS (Coucke et al., 2018) dataset contains 7 intents typical for the smart speaker use case. We refer the reader to Table 3.1 for exact statistics of all used datasets.

3.2.2 Setup

The main data-scarce setup that we consider In this chapter is the *few-shot setup*, where only $K = 10$ training examples are available for every intent of interest. Additionally, to compare to example extrapolation with fine-tuned language models as proposed by (Lee et al., 2021), we consider a *partial few-shot setup*. In the latter setting, we limit the amount

of training data only for a handful of *few-shot intents*² and use the full training data for others. When data augmentation is performed, we augment the few-shot intents to have N examples, where N is the median number of examples per intent of the original data.

To precisely describe the training and test data in all settings, we will use D_{part} to refer to dataset parts, i.e., train, validation, and test. In addition, we use D_F and D_M to refer to data-scarce and data-rich intents (the latter only occur in the partial few-shot setting). This notation is defined for all parts, therefore, $D_{part} = D_{\{F,part\}} \cup D_{\{M,part\}}$, $\forall part \in \{train, val, test\}$. When GPT-3 or a baseline method is used to augment the training data, we generate $N - K$ examples per intent and refer to the resulting data as $\tilde{D}_{F,train}$. We experiment with four different-sized GPT-3 models³ by OpenAI and GPT-J by EleutherAI⁴ to obtain $\tilde{D}$. The four GPT-3 models are: Ada, Babbage, Curie, and Davinci. In order, Ada is the smallest model and Davinci is the largest. Model sizes of GPT-3 engines are not known precisely but are estimated by Eleuther AI to be between 300M and 175B parameters⁵.

3.2.3 Training and Evaluation

We fine-tune BERT-large (Devlin et al., 2019a) on the task of intent classification by adding a linear layer on top of the [CLS] token (Wolf et al., 2020). In all setups, we use the original validation set to tune the classifier’s training hyperparameters. We chose to use the full validation set as opposed to a few-shot one to avoid issues with unstable hyperparameter tuning and focus on assessing the quality of the generated data.

Full few-shot. In this setup, we treat *all* the intents as few-shot and evaluate our method on the following three scenarios: (i) **Baseline**: all the intents are truncated to $K = 10$ samples per intent, (ii) **Augmented**: $\tilde{D}_{\{F,train\}}$ is generated using GPT and models are trained on $D_{\{F,train\}} \cup \tilde{D}_{\{F,train\}}$ and (iii) **EDA-baseline**: same as above, but $\tilde{D}_{\{F,train\}}$ is generated using Easy Data Augmentation (EDA) by (Wei and Zou, 2019). For each scenario, we report the 1) overall in-scope accuracy on the complete test set D_{test} , i.e., intent classification accuracy excluding OOS samples in the test set, and 2) few-shot classification accuracy of the models on $D_{\{F,test\}}$. For CLINC150, we also report out-of-scope recall

²We use the truncation heuristic provided by (Lee et al., 2021): https://github.com/google/example_extrapolation/blob/master/preprocess_clinc150.py

³<https://beta.openai.com/docs/engines>

⁴<https://github.com/kingoflolz/mesh-transformer-jax/>

⁵<https://blog.eleuther.ai/gpt3-model-sizes/>

	CLINC150		HWU64	Banking77	SNIPS
Model	IA (96.93)	OR (42.9)	IA (92.75)	IA (93.73)	IA (98.57)
EDA	92.66 (0.40)	43.81 (2.03)	83.67 (0.48)	83.96 (0.66)	92.50 (1.61)
Baseline (Ours)	90.28(0.49)	50.18(1.14)	81.43 (0.57)	83.35 (0.59)	89.69 (1.63)
Augmented					
Ada (Ours)	91.31 (0.34)	21.69 (1.57)	79.68 (0.83)	79.30 (0.42)	94.27 (0.52)
Babbage (Ours)	92.72 (0.33)	22.99 (2.39)	81.86 (0.78)	80.31 (0.41)	94.74 (0.67)
Curie (Ours)	93.37 (0.21)	25.85 (1.49)	82.85 (0.70)	83.50 (0.44)	94.73 (0.62)
GPT-J (Ours)	93.25 (0.19)	24.02 (1.45)	81.78 (0.56)	82.32 (0.90)	95.19 (0.61)
Davinci (Ours)	94.07 (0.18)	27.36 (1.08)	82.79 (0.93)	83.60 (0.45)	95.77 (0.86)
Augmented + Relabeled					
EDA	93.43 (0.22)	48.56 (1.84)	85.58 (0.73)	84.82 (0.57)	94.91 (0.66)
Ada (Ours)	95.09 (0.16)	41.38 (1.77)	88.53 (0.61)	88.45 (0.19)	97.03 (0.18)
Babbage (Ours)	95.39 (0.17)	40.58 (1.63)	89.49 (0.32)	88.86 (0.26)	96.89 (0.49)
Curie (Ours)	95.08 (0.19)	40.09 (2.38)	89.78 (0.47)	88.30 (4.64)	96.86 (0.31)
GPT-J (Ours)	95.11 (0.13)	43.94 (1.76)	89.52 (0.54)	88.94 (0.40)	97.33 (0.38)
Davinci (Ours)	95.08 (0.13)	40.76 (1.37)	89.53 (0.45)	88.89 (0.31)	97.03 (0.38)

Table 3.2: **Full few-shot results on CLINC150, HWU64, Banking77, and SNIPS datasets.** **IA:** Inscope Accuracy (mean (std)). **OR:** OOS-Recall (mean (std)). Towards the top of the table, we also report the test set performance (enclosed in parentheses) when all examples are used for fine-tuning (without any augmentation.)

(OOS recall) on D_{test} that we compute as the percentage of OOS examples that the model correctly labeled as such.

The purpose of this setting is to estimate what further gains can be achieved if the data generated by GPT were labeled by a human. We train an oracle $\mathcal{O}$ on the full training data D_{train} . We also use $\mathcal{O}$ to assess the quality of the generated data. Namely, we compute *fidelity* of the generated data as the ratio of generated utterances that the oracle labels as indeed belonging to the intended seed intent. A higher fidelity value means that the generated samples are more faithful to the original data distribution.

Partial few-shot. In this setup, we train $\mathcal{S}$ intent classifiers, choosing different *few-shot intents* every time to obtain D_F . We then average the metrics across these $\mathcal{S}$ runs. For CLINC150, $\mathcal{S} = 10$ corresponds to the 10 different domains, whereas for SNIPS, $\mathcal{S} = 7$ corresponds to the 7 different intents. We evaluate our method on the following three scenarios introduced by (Lee et al., 2021): (i) **Baseline:** models are trained without data augmentation on $D_{\{F,train\}} \cup D_{\{M,train\}}$. (ii) **Upsampled:** $D_{\{F,train\}}$ is upsampled to have

		CLINC150			SNIPS	
		Overall		Few-shot	Overall	Few-shot
	Classifier	IA	OR	A	IA	A
Baseline [♣]	T5	97.4	-	93.7	95.2	74.0
Upsampled [♣]	T5	97.4	-	94.4	95.9	80.0
Augmented (Ex2) [♣]	T5	97.4	-	95.6	97.8	94.0
Baseline (ours)	BERT	96.28 (0.06)	39.14 (0.82)	91.36 (0.47)	95.47 (0.45)	78.38 (3.34)
Upsample (ours)	BERT	96.20 (0.05)	40.21 (0.59)	90.93 (0.19)	95.29 (0.37)	79.28 (2.05)
Augmented (Ada)	BERT	96.16 (0.05)	34.37 (0.27)	92.60 (0.15)	97.30 (0.24)	94.41 (0.72)
Augmented (Babbage)	BERT	96.39 (0.06)	35.71 (0.46)	93.66 (0.21)	97.46 (0.25)	95.31 (0.74)
Augmented (Curie)	BERT	96.41 (0.06)	36.77 (0.93)	93.90 (0.21)	97.37 (0.19)	94.79 (0.64)
Augmented (GPT-J)	BERT	96.38 (0.05)	35.91 (0.94)	93.85 (0.25)	97.59 (0.21)	96.08 (0.39)
Augmented (Davinci)	BERT	96.45 (0.03)	37.52 (0.54)	94.28 (0.24)	97.66 (0.21)	96.52 (0.35)

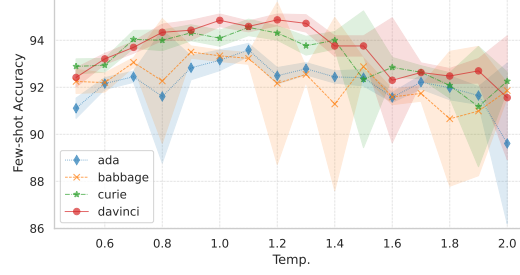
Table 3.3: **Partial few-shot results on CLINC150 and SNIPS datasets.** Refer to Section 3.2.3 for more details. **IA**: Inscope accuracy (mean (std)). **OR**: OOS Recall (mean (std)). **A**: Accuracy (mean (std)). [♣] (Lee et al., 2021).

N examples per intent. Then models are trained on upsampled $D_{\{F,train\}} \cup D_{\{M,train\}}$. (iii) **Augmented**: models are trained on $D_{\{F,train\}} \cup \tilde{D}_{\{F,train\}} \cup D_{\{M,train\}}$. For each scenario in this setup, we report the overall in-scope classification accuracy (and OOS Recall for CLINC150).

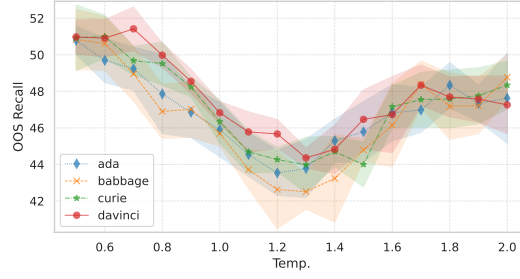
For both partial few-shot and full few-shot settings, we report means and standard deviations over 10 repetitions of each experiment.

3.3 Results

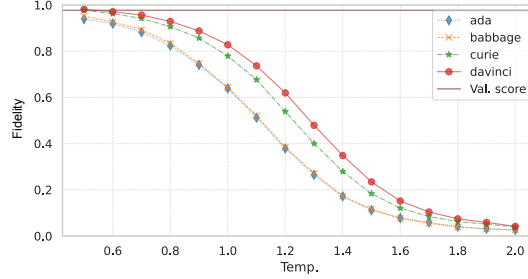
Full few-shot. Table 3.2 shows the results of our few-shot experiments. For CLINC150 and SNIPS, data augmentation with GPT-3 is very effective as it leads to respective accuracy improvements of up to approximately 3.7% and 6% on these tasks. These improvements are larger than what the baseline EDA method brings, namely 2.4% and 2.9% for CLINC150 and SNIPS. Importantly, using larger GPT models for data augmentation brings significantly bigger gains. Data augmentation results on Banking77 and HWU64 are, however, much worse, with no or little improvement upon the plain few-shot baseline. We present a thorough investigation of this behavior in Section 3.3.1. One can also see that data augmentation with GPT models lowers the OOS recall.



(a) Temperature v/s Few-shot accuracy



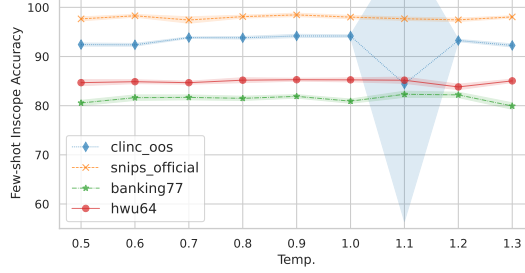
(b) Temperature v/s OOS recall



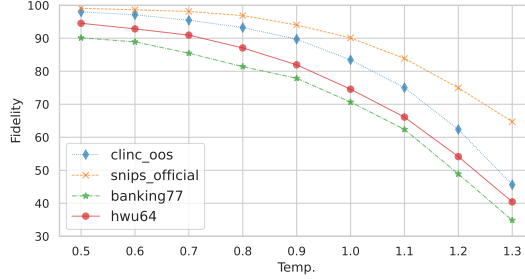
(c) Temperature v/s Fidelity

Figure 3.2: **Partial few-shot validation performance for different GPT-3 models and temperatures.** (a) few-shot accuracy, (b) OOS recall of intent classifiers trained on augmented sets, and (c) fidelity measured as the accuracy of the oracle on the augmented sets.

Next, we observe that relabeling EDA and GPT-generated sentences by the oracle gives a significant boost to accuracies across the board, confirming our hypothesis that human inspection of generated data could be fruitful. Importantly, we note that the



(a) Temperature v/s Few-shot in-scope accuracy



(b) Temperature v/s Fidelity

Figure 3.3: **Full few-shot validation performance for different GPT-J temperatures on different datasets.** (a) few-shot in-scope accuracy of intent classifiers trained on augmented sets, and (b) fidelity (oracle accuracy) of augmented sets generated by GPT-J with different temperatures.

magnitude of improvement for EDA is less than that of GPT models. This suggests that GPT models generate more diverse data that can eventually be more useful after careful human inspection. Lastly, relabeling also improves OOS recall on CLINC150, which is due to the fact that much of the generated data was labeled as OOS by the oracle.

Partial few-shot. Table 3.3 shows the results of our partial few-shot experiments on CLINC150 and SNIPS. By augmenting the dataset with GPT-generated samples, the few-shot accuracy improves by up to 2.92% on CLINC150 and 18.14% on SNIPS compared to the baseline setting. Our method achieves competitive results compared to Ex2 (Lee et al., 2021), both in terms of absolute accuracies and the relative gains brought by data augmentation. Note that Ex2 uses T5-XL (Roberts et al., 2020) with nearly 3 billion parameters as its base intent classifier, while our method uses BERT-large with only 340

Fidelity (3 intents)	HWU64	Banking77
w/o filtering (468)	60.26	57.31
w/ filtering (371)	72.51	65.54
3-way accuracy		
Davinci	86.36	78.75
10-shot BERT-large	82.95	65.54
Full data BERT-large	94.32	95.00

Table 3.4: The impact and the accuracy of using GPT-3 as a 3-way classifier on close intent triplets from HWU64 and Banking77 datasets. For fidelity, generated examples are rejected if the GPT-3 classifier labels them as not belonging to the seed intent. Classification accuracies are reported on the reduced validation+test sets where we only consider examples from the three confounding intents.

million parameters.

3.3.1 Analysis of GPT Outputs

Effect of GPT sampling temperature. We investigate the impact of generation temperature on the quality and fidelity of generated data. We perform this investigation on the CLINC150 dataset using the partial few-shot setup. Results in Figure 3.2 show that, for all engines, the generated data leads to the highest classification accuracy when the generation temperature is around 1.0, although lower temperatures result in higher OOS recall. We also observe that the fidelity of the generated samples decreases as we increase the temperature (i.e., higher diversity, see Figure 3.2c). This suggests that higher fidelity does not always imply better quality samples, as the language model may simply copy or produce less diverse utterances at lower temperatures.

Fidelity on different datasets. Our results in Section 3.3 show that data augmentation gains are much higher on CLINC150 and SNIPS than on HWU64 and Banking77. To contextualize these results, we report the fidelity of GPT-J-generated data for all these tasks in Figure 3.3b. Across all generation temperatures, the fidelity of the generated data is higher for CLINC150 and SNIPS than for HWU64 and Banking77. For all datasets, the fidelity is higher when the generation temperature is lower; however, Figure 3.3a shows that low-temperature data also does improve the model’s performance.

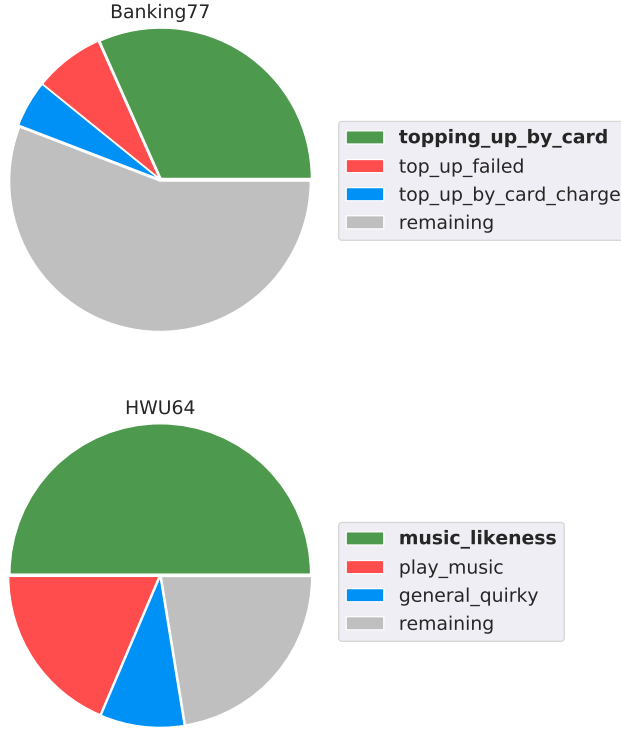


Figure 3.4: Distribution of labels as predicted by the oracle for lowest-fidelity intents in Banking77 and HWU64 datasets (“topping_up_by_card” and “music_likeness,” respectively). Green areas denote the portion of generated sentences deemed fit by the oracle for the lowest-fidelity intents in the two datasets. Red and Blue areas respectively correspond to the most common and the second most common alternative intent predicted by the oracle.

Data generation for close intents. To better understand the lower fidelity and accuracy on HWU64 and Banking77 datasets, we focus on intents with the lowest fidelities. Here, by intent fidelity, we mean the percentage of the intent’s generated data that the oracle classified as indeed belonging to the seed intent. In the Banking77 dataset, the lowest-fidelity intent is “topping_up_by_card.” For this intent, only 33% of the Davinci-generated sentences were labeled as “topping_up_by_card” by the oracle, implying that two-thirds of the sentences did not fit that intent, “top_up_failed” and “top_up_card_charge” being the two most common alternatives chosen by the oracle. Similarly, only 50% of the Davinci-generated sentences abide by the lowest-fidelity “music_likeness” intent in the HWU64

dataset, “play_music” and “general_quirky” being the most common intents among the “unfaithful” sentences. Figure 3.4 visualizes this high percentage of unfaithful generated sentences. It also shows the proportion of the two most common alternatives that the oracle preferred over the seed intent. Table 3.5 presents generated sentences for confounding intents in the HWU64 and Banking77 datasets. There are clear indications of mix-up of intents, e.g., Davinci generates, “play a song with the word honey,” which should belong to “play_music” rather than “music_likeness.” There are also instances where the LM mixes two intents; for instance, Davinci generates “Hi my app is activated on activate.co.in, but unable to top up my phone. I tried credit card, debit card and Paytm but fails,” which could belong to either “topping_up_by_card” intent (as it mentions about using credit card in the context of a top up) or “top_up_failed” (as the top up ultimately fails).

3.3.2 Can GPT Models Understand Close Intents?

We perform extra investigations to better understand what limits GPT-3’s ability to generate data accurately. We hypothesize that one limiting factor can be GPT-3’s inability to understand fine-grained differences in the meanings of utterances. To verify this hypothesis, we evaluate how accurate GPT-3 is at classifying given utterances as opposed to generating new ones. Due to the limited prompt size of 2048 tokens, we can not prompt GPT-3 to predict all the intents in the considered datasets. We thus focus on the close intent triplets from HWU64 and Banking77 datasets that we use in Table 3.5. We compare the 3-way accuracy of a prompted GPT-3 classifier to the similarly-measured 3-way performance of conventional BERT-large classifiers. We prompt GPT-3 with 10 examples per intent (see Figure C.1). For comparison, we train BERT-large classifiers on either the same 10 examples or the full training set. Table 3.4 shows that the Davinci version of GPT-3 performs in between the 10-shot and the full-data conventional classifiers. This suggests that while GPT-3’s understanding of nuanced intent differences is imperfect, it could still be sufficient to improve the performance of the downstream few-shot model. Inspired by this finding, we experimented with GPT-3’s classification abilities to improve the quality of generated data. Namely, we reject the generated utterances that GPT-3 classifies as not belonging to the seed intent. For both HWU64 and Banking77, this filtering method significantly improves the fidelity of the generated data for the chosen close intent triplets.

3.3.3 Comparison with GPT3Mix

To test our initial hypothesis that prior methods, such as GPT3Mix, are not suitable for intent classification, we experiment with the said method on the CLINC150 dataset using

Curie. Specifically, we include an enumeration of the 150 intent names in the prompt and randomly select one example for K intents. We observe a poor in-scope accuracy of 86.33% in the *Augmented* scenario⁶. Furthermore, the generated samples have low fidelity (27.96%). We also test a mixture of the GPT3Mix prompt and our prompt where we include all the K examples for the seed intent instead of 1 example per K randomly sampled intents. This mixed variant also performs poorly on CLINC150 and only achieves an in-scope accuracy of 86.05%⁷ and a fidelity of 33.56%. Our interpretation of this result is that GPT cannot handle the long list of 150 intent names in the prompt.

3.4 Limitations

Using large language models for data augmentation presents challenges, such as exhibiting social biases and generating toxic content. Therefore, samples generated using our prompting-based approach need to be considered carefully. To address such ethical concerns, human inspection would be the most reliable way to identify and filter out problematic generations. The practitioners who apply our method may also consider debiasing the language model before using it for generation (Schick and Schütze, 2021). Moreover, the generated data should be used as an additional source of data rather than the sole source, and the models trained on this data should be evaluated for fairness and bias. Finally, the generated data should be used in a controlled environment, and the models trained on this data should be monitored for any unintended consequences.

3.5 Conclusion

In this chapter, we discuss a prompt-based method to generate intent classification data with large pretrained language models. Our experiments show that generated data can be helpful as additional labeled data for some tasks, whereas, for other tasks, the generated data needs to be either relabeled or filtered to be helpful. We show that a filtering method that recasts the same GPT model as a classifier can be effective. Our filtering method, however, requires knowing the other intents that the generated data is likely to belong to instead of the seed intent. In the next chapter, we experiment with heuristics for approximately identifying the most likely actual intents for the generated utterances. This would complete a data generation and filtering pipeline that, according to our preliminary

⁶Average of 10 runs with a standard deviation of 1.17

⁷Average of 10 runs with a standard deviation of 0.59

results in Section 3.3.2 here, could be effective. Other filtering methods could also be applied, such as looking at the likelihood of the generated utterances as explored in a concurrent work by (Meng et al., 2022). An interesting research problem to explore is identifying which generated utterances most likely need a human inspection so that we make the data generation pipeline more sample efficient.

3.6 Key Takeaways

To summarize, the key takeaways from this chapter are:

- Off-the-shelf LLMs are an effective tool for generating data for intent classification and, by extension, other text classification tasks.
- LLMs struggle with distinguishing between close intents that have nuanced differences in their meanings.
- Overall, LLM-generated data has a lot of potential to improve the performance of classifiers in few-shot settings, but it needs careful contextualization of the task to be effective.

In the next chapter, we will study a more advanced data augmentation method that can handle data scarcity more effectively and can also handle augmentation for close categories while ensuring high fidelity and diversity in the generated data.

Davinci generated sentences	Seed Intent	Oracle Prediction
HWU64		
play a song with the word honey	music_likeness	play_music
you are playing music	music_likeness	play_music
'let me hear some of that jazz!'	music_likeness	play_music
i really like myspace music	play_music	music_likeness
i love the start lucky country music	play_music	music_likeness
thank you for the music	play_music	music_likeness
please play the next song	music_settings	play_music
play background music	music_settings	play_music
play the hour long loop of rock song	music_settings	play_music
need you to play that song one more time	play_music	music_settings
skip that song, its turkish	play_music	music_settings
pickup the beat or a temp track or audio plu- gin	play_music	music_settings
Banking77		
My last attempt to top up didn't seem to work, any success?	topping_up_by_card	top_up_failed
I tried to top off my wallet using my card but it says "top up failed".	topping_up_by_card	top_up_failed
I cannot top-up by my cellular phone number? How do I do that?	topping_up_by_card	top_up_failed
Can you transfer money to my Ola prepaid op- tion? Or help me top up my card to money. They never accept my card so I always have to suffer	top_up_failed	topping_up_by_card
Hi my app is activated on activate.co.in, but unable to top up my phone. I tried credit card, debit card and Paytm but fails	top_up_failed	topping_up_by_card
I try to top up my card but it's not going through. It's still on pending status. Do I need to wait or did I do something wrong	top_up_failed	pending_top_up
I tried top-up with my card but notification shows that 'Pending'. This has been happen- ing since last night. Can you tell me what's going on	top_up_failed	pending_top_up
Top up didn't go through.	pending_top_up	top_up_failed
Did my master card top-up fail?	pending_top_up	top_up_failed

Table 3.5: Davinci-generated sentences for closely-related intents in HWU64 and Banking77 datasets. Highlighted sub-strings indicate a difference with respect to the seed intent.

Chapter 4

Advanced Data Augmentation with LLMs for Text Classification

Building on the learnings of the previous chapter, we progress to operating in even more challenging zero-shot and 2-shot setups and consider a wider array of text classification tasks, such as detecting customer complaints on social media platforms and detecting the subjectivity of movie reviews.

So far, we have discussed LLM- data augmentation that either first fine-tune a generative language model on an initial seed dataset to later synthesize new training data (Wu et al., 2019; Kumar et al., 2019, 2020; Anaby-Tavor et al., 2020) or they prompt an off-the-shelf LLM to generate additional data samples (Yoo et al., 2021; Sahu et al., 2022; Lin et al., 2023); however, we have noted that the fine-tuning step quickly becomes a bottleneck in the absence of sufficient seed examples. Additionally, existing prompting-based techniques suffer from a key limitation: their prompts only focus on using information from a single class when generating augmentations and they do not incentivize the LLM to diversify the generated examples. Recently, some recent works also show that instruction-tuned LLMs like InstructGPT (Ouyang et al., 2022) and Vicuna (Chiang et al., 2023) are prone to mode collapse (Zhu et al., 2023).

To address the aforementioned limitations, we propose a two-step prompting-based technique, PromptMix. First, PromptMix instructs an LLM to generate new examples by mixing information from multiple classes. The degree of mixing is controlled by a parameter α , and using a range of α values diversifies the generated examples. Figure 4.1 illustrates the difference in the generated examples between PromptMix and prior standard augmentation approaches. While promoting mixup can improve the diversity of the

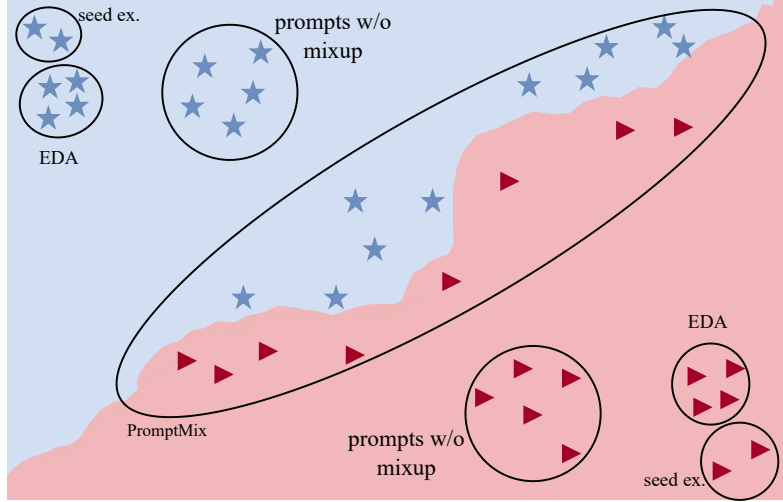


Figure 4.1: PromptMix focuses on generating examples near the class boundary of two classes, unlike other standard augmentation approaches like EDA (Wei and Zou, 2019) and prompting-based methods without Mixup (Sahu et al., 2022; Lin et al., 2023), that only use the information of a single class for augmentation.

augmented set, it also increases the risk of false positive generations (generating examples that do belong to the desired class). To address this problem, PromptMix uses a relabeling scheme in the second step to improve the faithfulness of the generated examples. In particular, it uses an LLM as a classifier to assign new labels to the generated examples. We conduct comprehensive experiments on diverse text classification tasks and find that training a classifier on these relabeled examples is highly robust, and our approach also achieves knowledge transfer from a massive LLM like GPT3.5 into much smaller models like BERT (Devlin et al., 2019a) and DistilBERT (Sanh et al., 2019). Figure 4.2 demonstrates the complete PromptMix framework.

To summarize the key contributions in this chapter: **a)** we propose PromptMix, a novel two-step prompting-based method to generate a diverse set of labeled examples for any text classification dataset; **b)** we demonstrate that generating borderline examples and relabeling them improves knowledge transfer from a massive LLM like GPT3.5 into much smaller models like DistilBERT and BERT, even without abundant seed examples. We also show that 2-shot PromptMix outperforms multiple data augmentation baselines that

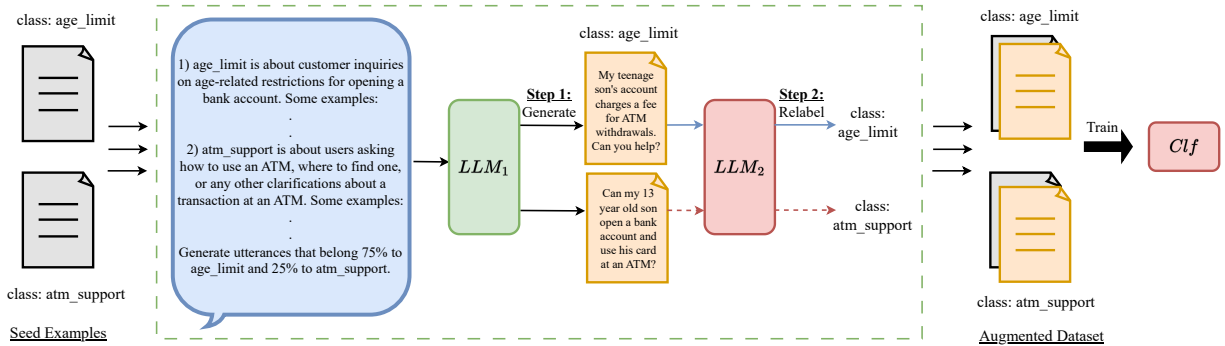


Figure 4.2: **PromptMix framework.** The dashed box shows the generation process for every class in the dataset. **Step 1:** we generate augmentations (yellow documents) by feeding the Mixup prompt proposed in Section 4.1.1 to an LLM. **Step 2:** we relabel *all* the augmentations using an LLM (as described in Section 4.1.2) to fix any incorrect labels from Step 1. **Note:** LLM_1 and LLM_2 can be identical. Refer to Figures C.2 and C.3 for detailed versions of our prompts.

t	Accuracy
2	73.3
4	76.8
6	69.1

Table 4.1: Validation accuracy for different values of t on TREC6. Refer to Section 4.3 for details on the experimental setup.

use 5-shot or more seed data; and **c)** we also benchmark open-source LLMs for the task like LLaMA-2 in addition to using closed-source LLMs like GPT3.5-turbo and GPT-4.

4.1 Methodology

We hypothesize that training a robust text classifier requires the training data to have a good mix of borderline examples (Swayamdipta et al., 2020). This section describes our method PromptMix, where we first instruct an LLM (GPT3.5-turbo, in our case) to generate difficult examples near class boundaries and then relabel those generations to improve the faithfulness of the generated data (see Figure 4.2).

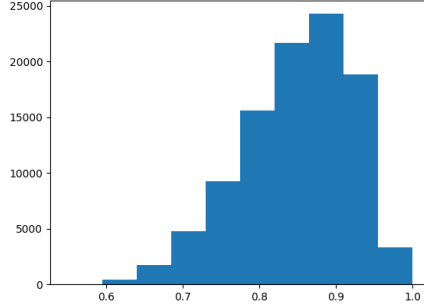


Figure 4.3: $\alpha = \text{round}(10(x + 1))/20 \forall x \sim \beta(5, 2)$.

4.1.1 Step 1: Generating examples

First, we manually write short descriptions for every class in the dataset. We use descriptions in our prompts to facilitate the usage of the approach in the extremely data-scarce zero-shot and two-shot settings. **Next**, we randomly select a group of $t(= 4)$ classes $c \subseteq C$ classes, where C denotes the set of all classes in the dataset. We conduct a small-scale experiment with different values of t (classes to include in the prompt) and choose $t = 4$ based on the validation performance of our method as shown in Table 4.1. For each class in c , we combine the description and k examples in the prompt, k being the k -shot setting (see part 1 in Figure C.2). **Lastly**, for each class $c_i \forall i \in [1, t]$ in the subset, we instruct GPT3.5-turbo¹ to generate n example utterances that are a mix of two classes: c_i and a randomly selected class $c_j \in c \setminus \{c_i\}$. In particular, we instruct the LLM to generate utterances that belong $\alpha\%$ to class c_i and $(1 - \alpha)\%$ to class c_j (see part 2 in Figure C.2). We use $\alpha = \text{round}(10(x + 1))/20 \forall x \sim \beta(5, 2)$ in our experiments. This is a modification of the standard β -distribution by restricting its range to the half interval of $(0.5, 1.0]$ with a peak near 1.0. We choose $\alpha > 0.5$ to incentivize the LLM to generate examples that are mixed up but still belong to c_i . We round off the α to the nearest 0.05 to avoid decimal values that would be arbitrary, given we are operating with natural language instructions. Figure 4.3 visualizes the distribution α is sampled from.

¹specifically, we use the `gpt-3.5-turbo-0613` engine

4.1.2 Step 2: Improving Fidelity

Since borderline examples are inherently difficult to put into a category, the LLM may generate examples for the minority class in the prompt (c_j). In other words, the LLM can generate false positives. To address this issue, we employ GPT3.5-turbo as a classifier and relabel the generated examples. When constructing the classification prompt, we choose the top-5 closest classes according to the similarity between the SBERT sentence embedding (Reimers and Gurevych, 2019) of the generated example and available examples in the few-shot training set. For the zero-shot setting, we use the class name’s embedding instead of the available examples. We then follow a similar process as in Step 1 to construct the prompt, but instead ask the LLM to classify the provided sentence into one of the classes in the prompt (see Figure C.3). To ensure a valid prediction, we retrieve the closest class in the dataset based on the cosine similarity of the SBERT embedding of the GPT-generated class and the ground-truth classes in the dataset ². We do not include all the classes in the prompt because **a)** some datasets can have hundreds of classes that would not fit in the context size of the LLM, and **b)** we found in our preliminary experiments that long contexts degraded GPT’s classification ability. Figure C.3 shows the prompt structure for relabeling a generated example ³.

After generating borderline examples and relabeling them, we train a text classifier on the combined PromptMix-generated data and the original seed data. Specifically, we fine-tune DistilBERT_{base} and BERT_{base} classification models. In the subsequent sections, we show that our method achieves text augmentation, pseudo-labeling, and knowledge distillation in one cohesive pipeline.

4.2 Does PromptMix Generate Borderline Examples?

Table 4.2 shows that using mixup generates sentences that contain information from the majority and the minority class, compared to sentences generated without mixup that only contain information about the majority class. This demonstrates that mixup leads to the generation of borderline examples.

Table 4.2 also shows some false positives, where GPT generates sentences like, “Do I need to be over a certain age to use an ATM?” for the majority class age_limit. The

²we use the `sentence-transformers/all-mpnet-base-v2` model from the sentence-transformers library

³We open-source our code at <https://github.com/ServiceNow/PromptMix-EMNLP-2023>

Input classes	age_limit (75%) atm_support (25%)
w/ Mixup	- Can someone under 18 open an account with unlimited ATM withdrawal limit? - Can I open an account for my teenage daughter ? - Do I need to be over a certain age to use an ATM? - Can I use my children's debit card to withdraw money from the ATM?
w/o Mixup	- Can I open an account for a minor? - What is the minimum age requirement to get a bank account? - Is there an age restriction for account holders? - Is there a minimum age requirement to open an account?

Table 4.2: **Effect of Mixup.** GPT3.5-turbo generations for the prompt shown in Figure C.2. We’ve highlighted parts about age_limit in yellow and parts about atm_support in cyan. We see clear evidence of mixup at work as sentences generated using mixup contain information about both classes.

class age_limit covers all age-related queries about *opening a bank account*, whereas the generated sentence is an age-related query about *using an ATM*, making it a better fit for the minority class atm_support. By relabeling such false positives, we aim to rectify the mismatch between the generated examples and their assigned class labels. Figure C.3 shows that GPT correctly predicts the new class of the sentence as atm_support, verifying the importance of the relabeling step in our approach.

Overall, we find the observations in this section to be strong evidence that PromptMix can generate high-quality datasets even in aggressive few-shot text classification setups. Next, we conduct an extensive suite of experiments to verify the effectiveness of PromptMix.

	B77	TREC6	SUBJ	TC
Classes	77	6	2	2
# Train	9002*	5452	8000	100*
# Valid	1001*	500	2000	-
# Test	3080	500	2000	3349*

Table 4.3: Statistics of the text classification datasets we use in our experiments. * indicates that we split the original data into training and validation/testing instead of using a split provided by the dataset authors.

4.3 Experimental Setup

4.3.1 Datasets

We use four text classification datasets with varying levels of granularity among the classes. Banking77 (**B77**) (Casanueva et al., 2020) is a single-domain dataset with 77 banking-related classes, where the difference between multiple classes is nuanced. The fine-grained nature combined with a high number of target classes makes Banking77 a good test bed for verifying the scalability and effectiveness of our approach. The following three datasets have coarse labels but cover a variety of domains, allowing us to test the adaptability of our method across different domains. **TREC6** (Voorhees et al., 1999) is a question classification dataset with six broad classes of questions in English. The subjectivity dataset (**SUBJ**) (Pang and Lee, 2004) contains movie reviews with objectivity labels. Lastly, the twitter complaints dataset (**TC**) (Preotiuc-Pietro et al., 2019) contains tweets annotated by whether they contain a complaint or not. We refer the reader to Table 4.3 for exact statistics of all the datasets.

4.3.2 Few-shot Setup

For our experiments, we consider 1) a 2-shot setup, where only $k = 2$ training examples are available for every class, and 2) a zero-shot setup, where we do not have access to any training examples.

Notations. We will use D_{part} to refer to the dataset parts, i.e., train, validation, and test. When augmenting the training data using any method, we generate N examples per class and refer to the resulting data as $\tilde{D}_{A,train}$ (obtained after Step 1). We refer to the relabeled version of the resulting data (obtained after Step 2) as $\tilde{D}_{A+R,train}$.

Method	Prompt Features				B77		TREC6		SUBJ		TC	
	Ex.	Desc.	41 class	Mixup	A1	A2	A1	A2	A1	A2	A1	A2
DistilBERT_{base}												
Baseline	-	-	-	-	16.0 (0.9)		31.7 (0.8)		64.4 (0.7)		38.8 (0.6)	
EDA	-	-	-	-	47.8 (0.7)		40.9 (0.6)		82.3 (0.4)		42.9 (0.7)	
GPT3Mix	✓		✓	✓	-		57.4 (2.8)		89.3 (1.5)		-	
Sahu et al. (2022)	✓				68.9 (1.4)	71.6 (0.6)	51.1 (1.3)	56.9 (0.7)	81.8 (1.3)	83.7 (0.4)	51.5 (0.6)	55.4 (0.5)
+ desc.	✓	✓			71.1 (1.2)	72.4 (0.7)	63.8 (1.1)	64.9 (0.6)	84.2 (1.1)	86.3 (0.5)	57.2 (0.7)	67.9 (0.2)
PromptMix w/o Mixup (zero-shot)		✓	✓		72.2 (1.3)	76.1 (0.8)	61.0 (1.3)	61.6 (0.6)	82.5 (1.2)	84.2 (0.5)	56.7 (1.3)	67.7 (0.5)
PromptMix (zero-shot)		✓	✓	✓	69.2 (2.3)	77.4 (1.2)	54.1 (1.7)	65.7 (0.7)	80.0 (1.5)	85.4 (0.6)	56.0 (1.3)	71.5 (0.9)
PromptMix w/o Mixup	✓	✓	✓		73.1 (1.3)	78.4 (0.5)	65.4 (1.2)	66.2 (0.5)	85.4 (1.2)	87.8 (0.3)	64.6 (0.7)	71.5 (0.6)
PromptMix	✓	✓	✓	✓	72.3 (1.1)	79.7 (0.7)	60.6 (1.4)	68.7 (0.6)	77.5 (1.7)	89.9 (0.8)	61.8 (1.4)	75.3 (1.2)
BERT_{base}												
Baseline	-	-	-	-	22.6 (1.2)		33.0 (0.6)		71.6 (0.8)		42.7 (0.6)	
EDA	-	-	-	-	49.2 (0.9)		51.1 (0.6)		84.5 (0.5)		47.8 (0.4)	
GPT3Mix	✓		✓	✓	-		60.5 (6.1)		90.6 (1.1)		-	
LINDA (5-shot)	✓		✓	✓	-		62.2 (3.1)		-		-	
Sahu et al. (2022)	✓				70.7 (1.4)	71.6 (1.1)	51.3 (1.2)	57.8 (0.8)	83.6 (1.3)	85.5 (1.0)	55.3 (1.1)	58.1 (0.3)
+ desc.	✓	✓			72.8 (1.1)	73.0 (0.7)	64.3 (1.3)	67.1 (0.2)	87.0 (1.7)	87.2 (1.1)	66.1 (1.4)	65.2 (1.1)
PromptMix w/o Mixup (zero-shot)		✓	✓		74.0 (1.9)	76.4 (1.1)	64.2 (1.6)	68.5 (0.9)	83.6 (1.4)	85.9 (0.8)	64.3 (1.2)	68.9 (0.3)
PromptMix (zero-shot)		✓	✓	✓	71.3 (2.4)	77.6 (1.3)	55.5 (1.7)	67.5 (0.4)	80.7 (1.4)	89.5 (0.7)	57.6 (2.4)	74.7 (1.5)
PromptMix w/o Mixup	✓	✓	✓		74.4 (1.4)	78.5 (0.7)	70.1 (1.5)	71.6 (0.2)	87.0 (1.3)	90.0 (0.1)	71.0 (0.9)	72.3 (0.3)
PromptMix	✓	✓	✓	✓	74.2 (1.6)	80.1 (0.9)	63.3 (2.5)	73.7 (1.1)	77.3 (2.2)	91.7 (1.1)	66.8 (0.8)	78.4 (0.8)
NN+GPT3.5	✓	✓	✓	-	79.9		74.4		90.4		88.6	

Table 4.4: Test classification accuracy (out of 100%) on four datasets, averaged across three random seeds (with standard deviation in brackets). A1 is the accuracy of the classifier on the generated dataset, and A2 is the accuracy on the generated+relabelled dataset. **Note:** we use GPT3Mix results from Yoo et al. (2021), and LINDA results from Kim et al. (2021). “-” for a method shows that the particular prompt feature is inapplicable to that method.

4.3.3 Training and Evaluation

Training. We fine-tune DistilBERT_{base} and BERT_{base} models for text classification by adding a linear layer on top of the [CLS] token (Wolf et al., 2020). In all the setups, we fine-tune the classifier for 5 epochs. We use a learning rate of 6×10^{-5} and weight decay of 1×10^{-3} for B77 and a learning rate of 4×10^{-5} and weight decay of 1×10^{-2} for all the other datasets. Finally, we generate $N = 50$ examples per class for B77 and TREC6 and $N = 100$ examples per class for SUBJ and TC. We used the validation sets of TREC6 and SUBJ to choose the specific N values as they provide a good cost-to-performance ratio. We chose TREC6 over B77 to minimize our costs for using GPT3.5-turbo.

We perform all hyperparameter tuning using DistilBERT_{base} on B77 and TREC6. We limit our tuning to the two datasets to obtain two sets of hyperparameters: one for large-scale datasets like B77 and another for small-scale datasets like TREC6, SUBJ, and TC. Additionally, we use the same set of hyperparameters for the BERT_{base} classification model. We use the full validation set for tuning instead of a few-shot one to avoid issues with unstable hyperparameters.

We run experiments for the following scenarios: **1) Baseline (2-shot)**. All the classes are reduced to 2 examples per class, and we fine-tune a $\text{DistilBERT}_{base}/\text{BERT}_{base}$ model on the reduced dataset. **2) NN+GPT3.5**. We use the nearest-neighbor approach to populate the prompt as described in Section 4.1.2 and then prompt GPT3.5-turbo to classify the test set examples (see Figure C.3 for reference). **3) Sahu et al. (2022)**. The prompting-based approach for data augmentation described in Chapter 3 that lists all the seed examples for a single class in the prompt and prompts the LLM to generate more examples based on it. It does not promote mixup in the generated examples. **4) Prompt-Mix**. Our proposed approach with multiple classes in the prompt, instructing the LLM to generate borderline examples. **5) PromptMix (zero-shot)**. We remove seed examples from PromptMix but still use the manually written class descriptions. **6) Easy Data Augmentation (EDA)**. An edit-based augmentation technique proposed by Wei and Zou (2019) that applies rule-based changes to existing training examples to generate additional examples. **7) GPT3Mix**. A mixup-based augmentation method using soft labels for pseudolabeling proposed by Yoo et al. (2021). Notably, Yoo et al. (2021) measure the degree of few-shot in terms of the percentage of training examples used for augmentation (sub-sample percentage) and experiment with different percentages. We report their best-performing model for a sub-sample percentage of 1%, where GPT3Mix uses 55 training examples in TREC6 and 80 training examples in SUBJ (combined for all the classes). **8) CPFT**. Zhang et al. (2021) use contrastive pre-training before fine-tuning a model for few-shot intent detection using RoBERTa_{base} . **9) USE**. A large multilingual model pre-trained on 16 languages (Yang et al., 2020). **10) CONVERT**. An intent detection model from Casanueva et al. (2020), which uses a dual encoder setup pretrained on 654 million Reddit sentences.

Several works in the past have explored 5-shot, 10-shot, and even 100-shot settings for data augmentation (Kumar et al., 2020; Zhang et al., 2021). But, to the best of our knowledge, we are the first to explore data augmentation in 2-shot and zero-shot settings. Through our experiments, we aim to measure the extent of data scarcity that an LLM like GPT3.5-turbo can handle, which is optimized to follow instructions.

Evaluation. We measure the performance of the classifiers in terms of test classification accuracy and report the full set of results in Table 4.4. To note, **A1** denotes the accuracy of the classifier on $D_{train} \cup D_{A,train}$ (augmented dataset after Step 1 in Section 4.1.1) and **A2** denotes the accuracy of the classifier on $D_{train} \cup D_{A+R,train}$ (augmented+re-labeled dataset after Step 2 in Section 4.1.2). We run each experiment for three random seeds and report the mean accuracy in Table 4.4.

Method	Accuracy
USE (5-shot)	76.3
CONVERT (5-shot)	75.3
USE+CONVERT (5-shot)	77.8
CPFT (5-shot)	80.9
PromptMix (2-shot)	<u>80.1</u>

Table 4.5: Comparing 2-shot PromptMix results with 5-shot baselines on B77. **Note:** CPFT, USE, CONVERT, and USE+CONVERT as reported in [Zhang et al. \(2021\)](#).

Method	B77	TREC6	SUBJ	TC
Sahu et al. (2022)	9.4	21.0	3.9	2.5
+ desc.	8.5	28.0	1.5	8.5
PromptMix (zero-shot)	<u>32.8</u>	<u>36.2</u>	<u>28.8</u>	<u>16.2</u>
w/o Mixup	22.5	18.9	2.6	7.8
PromptMix	33.9	33.8	42.0	23.4
w/o Mixup	22.2	20.3	6.1	14.0

Table 4.6: Percentage of generated examples relabeled by GPT3.5-turbo in Step 2 for different methods. **Note:** percentages are averaged across three runs.

4.4 Results

Referring to Table 4.4, we first note that A1 performs significantly better than baseline (2-shot) for DistilBERT_{base} and BERT_{base} across all four datasets. This confirms that data augmentation is helpful in data-scarce setups.

4.4.1 On the Effect of Relabeling Step

Table 4.6 shows the percentage of generated examples relabeled by GPT3.5-turbo for different augmentation methods. We note that the relabeling percentage is higher when we add mixup to the prompt (PromptMix v/s PromptMix w/o Mixup). High relabeling percentages suggest that mixup generates more borderline examples than any other augmentation method. This verifies the importance of the relabeling step in our method, which is highly effective in remedying the problem of false positive generations. This is further demonstrated by higher A2 values than A1 across the board. Specifically, for PromptMix, we

GPT3.5-turbo generated sentences	Before Relabeling → After Relabeling
Twitter Complaints	
This tweet also expresses an opinion about a product, but includes a minor complaint about the battery life.	no_complaint → complaint
I appreciate the quick response from @Uber_Support, but the driver’s behavior was unacceptable. #goodandbad	no_complaint → complaint
@Starbucks this latte is amazing! Can you tell me what kind of beans you use? #No_Complaint	complaint → no_complaint
I’m really impressed with @Nike’s customer service. They helped me solve my problem quickly and efficiently.	complaint → no_complaint
Banking77	
Why was my transfer declined after I tried to send money to my daughter’s account?	age_limit → declined_transfer
My transfers keep getting declined, but I’m old enough to have an account. Can you help me?	age_limit → declined_transfer
My card got declined when I tried to top up my account. What’s going on?	topping_up_by_card → declined_card_payment
Is there a limit to how much I can top up with my card? It’s not letting me add more money.	topping_up_by_card → top_up_limits
I just tried topping up my account with my card and it didn’t work.	topping_up_by_card → top_up_failed

Table 4.7: **Effect of Relabeling.** GPT3.5-turbo sentences that were relabeled in the Twitter Complaints and Banking77 datasets. Highlighted text denotes the difference with respect to the class of the generated sentence before relabeling.

observe an improvement of 7.4% on B77, 8.1% on TREC6, 12.4% on SUBJ, and 13.5% on TC, when we use DistilBERT_{base}; and 5.9% on B77, 10.4% on TREC6, 14.4% on SUBJ, and 11.6% on TC when we use BERT_{base}. In addition to these significant improvements, we note that the degree of improvement increases as the classification task gets easier in terms of the number of target classes. Table 4.7 shows some examples of leaked generations that GPT rectifies during the relabeling step.

4.4.2 Borderline Examples Aid Knowledge Distillation

In Table 4.4, we notice that PromptMix achieves almost similar performance as NN+GPT3.5 on three out of four datasets. Specifically, PromptMix outperforms NN+GPT3.5 on B77 (80.1 v/s 79.9) and SUBJ (91.7 v/s 90.4) and is competitive on TREC6 (73.7 v/s 74.4). The gap between PromptMix and NN+GPT3.5 is larger on TC compared to other datasets (78.4 v/s 88.6). This might be due to the nature of the dataset, which contains extensive use of social media language that GPT knows about, but two examples might be too few to cover the vast diversity of complaints in the wild. Overall, these results are promising as GPT3.5-turbo with 175B parameters is $\sim 2600\times$ larger than DistilBERT_{base} and $\sim 1600\times$ larger than BERT_{base}, which only have 67M and 110M parameters, respectively. In both cases, our final classifiers are $> 99.9\%$ smaller than GPT3.5-turbo. We do not see such strong performance for any other generation method, even after relabeling. This confirms that generating borderline examples in PromptMix makes for a high-quality dataset generation method that aids the transfer of knowledge of large-scale models such as GPT3.5-turbo into much smaller classifiers in data-scarce settings.

4.4.3 PromptMix v/s Other Augmentation Methods

In Table 4.4, we compare the 2-shot performance of PromptMix against several strong baselines on the four datasets. First, we note that PromptMix is significantly better than the EDA baseline on all datasets. Next, we compare the results of PromptMix (which uses GPT3.5-turbo for generation) with GPT3Mix (which uses GPT3’s Davinci model for generation) and LINDA as they are the closest methods to ours in terms of motivation. PromptMix outperforms both LINDA and GPT3Mix by a huge margin even though LINDA is 5-shot and uses pretraining, and GPT3Mix utilizes 1% of training samples for augmentation, translating to 55 total seed examples compared to our 12 seed examples on TREC6, and 80 seed examples compared to our 4 seed examples on the SUBJ dataset.

Table 4.5 shows that 2-shot PromptMix outperforms USE, CONVERT, and USE+CONVERT on the Banking77 dataset. It also achieves an accuracy of 80.1% compared to 80.9% by CPFT, which is highly competitive. To emphasize, all the considered baselines are 5-shot, with some models using additional data for pretraining. Moreover, it is possible to further improve the performance of PromptMix with more dataset- and model-specific hyperparameter-tuning; however, our core message is to show that generating borderline examples combined with relabeling is a highly promising method for data augmentation in 2-shot and zero-shot text classification setups.

4.4.4 Descriptions are Impactful

Our experiments that were used to evaluate the effectiveness of using human-written descriptions show promising results. First, we note that simply adding a class description to the prompt leads to decent performance gains on all the datasets (Sahu et al. (2022) w/ and w/o desc). Comparing PromptMix (zero-shot) with and without Mixup against Sahu et al. (2022) + desc. shows that using only descriptions in the prompt (no seed examples) leads to a significant boost in model performance. We also observe that using multiple class descriptions in a prompt is either better or equivalent to using a single class with its description and examples. This suggests that providing more context about other classes helps LLM generate better quality augmentations for a given class. Therefore, it follows intuitively that PromptMix, which combines the usage of multiple classes with descriptions as well as seed examples, leads to the best performance on all the datasets.

4.4.5 Open-source v/s Closed-source models

Table 4.4 and 4.5 showcase strong capabilities of GPT3.5-turbo for our task. However, GPT3.5-turbo is a closed-source language model provided by OpenAI, and it can quickly get costly as we increase the number of classes in the dataset. Therefore, we conduct a small-scale experiment on the TREC6 dataset with open-source LLMs. In particular, we prompt open-source LLMs instead of GPT3.5-turbo to synthesize new examples and relabel them. Next, we finetune a BERT_{base} classifier on the augmented+reabeled dataset.

GPT-Neo (1.3B), GPT-J (6B), and the more recent instruction-tuned Vicuna and Stable-vicuna (13B) (Chiang et al., 2023) models achieve very poor performance compared to GPT3.5-turbo. We find that even for medium-sized LLMs like Vicuna and Stable-vicuna, inference time is a major bottleneck⁴. However, our experiments with Llama-2 show promising results. We used different-sized Llama-2 models (7b, 13b, 70b) on the TREC6 dataset and observed a strong correlation between model size and test accuracy. In particular, we note that Llama-2-70b-chat-hf might be a decent alternative to the closed-source GPT3.5-turbo model. We also test GPT-4 (larger than the GPT3.5 model) and observe a significant boost in classification accuracy. Table 4.8 shows the results⁵.

⁴We tried on a 32G V100 with 8-bit quantization

⁵We use [Anyscale endpoints](#) to access Llama-2 models.

Method	Accuracy
Baseline	33.0
PromptMix (Llama-2-7b-chat-hf)	55.6
PromptMix (Llama-2-13b-chat-hf)	66.6
PromptMix (Llama-2-70b-chat-hf)	70.8
PromptMix (GPT3.5-turbo)	73.7
PromptMix (GPT-4)	86.2
NN+GPT3.5	74.4
NN+Llama-2-70b-chat-hf	76.2
NN+GPT4	<u>88.2</u>

Table 4.8: Comparison of different open-source LLMs v/s GPT on TREC6. **Note:** Baseline refers to the 2-shot accuracy of the BERT_{base} classifier. We replace GPT3.5-turbo with GPT4 and LLama-2-70b-chat-hf to get the NN+GPT4 and NN+LLama-2-70b-chat-hf baselines, respectively.

4.5 Limitations

Similar to our method in Chapter 3, we employ an LLM to generate text data, so they can potentially generate harmful or biased content. We recommend monitoring the generated data for harmful content and bias if the model is deployed in real-world applications.

While our results indicate a promising potential to use LLMs for generating data in highly aggressive few-shot setups, we also identify a few limitations in this section for our approach. First, we rely completely on GPT’s pseudolabels to tackle false positive generations during the augmentation step. Secondly, since we use embeddings to ensure that pseudolabel is a valid class in the dataset, we ignore potential out-of-scope/out-of-domain (OOS/OOD) generations. We did observe a few instances where GPT pseudolabels were of the form, “This sentence does not belong to any of the provided classes.” This presents a good avenue to introduce human interventions that can judge GPT pseudolabels and can identify OOS/OOD examples, which can be helpful in many real-life tasks. Here, we would like to emphasize that while human interventions can greatly help, they also present the challenge of minimizing human labor. So, we could possibly introduce a self-verification check in the pipeline where the LLM automatically identifies such instances.

Second, while our experiments with open-source models suggest that the larger open-source models like LLama-2-70b might be a good alternative to closed-source models like

GPT3.5, thereby greatly cutting API costs, they still demand significant computational resources. Therefore, efforts towards distilling the knowledge of bigger LLMs like Llama-2-70b into smaller models would greatly aid in making these models more accessible and feasible for use.

4.6 Conclusion

To conclude, this chapter describes PromptMix, a novel two-step prompting-based method for generating borderline augmented examples in extreme few-shot text classification setups. Our method combines the process of text augmentation, pseudolabeling, and knowledge distillation in a cohesive pipeline. We show that by generating borderline training examples and relabeling them using a large teacher model like GPT, we can transfer the knowledge of such massive LLMs into much smaller models like DistilBERT_{base} and BERT_{base} that are much cheaper to deploy in real-life settings. Furthermore, we also demonstrate that our 2-shot PromptMix beats multiple 5-shot or higher data augmentation baselines, making it a highly effective data augmentation approach.

4.7 Key Takeaways

In summary, we list the key takeaways from this chapter:

- The 2-shot PromptMix approach is highly effective for aggressive few-shot text classification setups, beating multiple 5-shot or higher data augmentation baselines.
- Generating borderline training examples and relabeling them using a large teacher model like GPT effectively induces a transfer of knowledge from massive LLMs with billions of parameters into much smaller models like DistilBERT_{base} and BERT_{base}.
- Providing information about multiple classes is key to generating high-quality augmented examples that can handle use cases with closely related classes.
- Larger open-source models like Llama-2-70b might be a good alternative to closed-source models like GPT3.5, thereby greatly cutting API costs.

In the next two chapters, we will delve into the problem of text summarization in data-scarce settings and explore how we can leverage the power of LLMs for training summarization models that generate high-quality summaries with minimal supervision.

Chapter 5

Semi-Supervised Extractive Text Summarization with LLMs

Text summarization is a crucial task in today’s data-driven era. Major enterprises store customer service chatlogs and wish to create an automated text summarization system to enhance the user experience (Feigenblat et al., 2021). With the exponential growth of digital content, the need for automated summarization techniques that swiftly process and distill information has become paramount (Koupaei and Wang, 2018; Cohan et al., 2018). Broadly speaking, text summarization setups are categorized into two types: abstractive, where the summaries are concise *paraphrases* of the original text (Barzilay and McKeown, 2005; Nallapati et al., 2016), and extractive, where the summaries combine existing sentences *verbatim* from the original text (Radev et al., 2004; Li et al., 2006; Wong et al., 2008). While both systems have their advantages, in this chapter, we primarily focus on extractive summarization, as it is a simpler task, and we want to focus our efforts in this chapter on in-depth analysis of LLM-based methods for a semi-supervised summarization task. Moreover, in today’s fast-paced lifestyle, extractive summarization systems have become increasingly relevant as they are fast and interpretable while preserving the source’s integrity and reducing potential misinterpretation.

Obtaining a large annotated extractive summarization dataset to train a fully supervised model is financially expensive and it also incurs a high cost of human labor (Wang et al., 2022). Real-life scenarios often have a small labeled set alongside a large pool of unlabeled data. Semi-supervised learning (SSL) has been successfully used in such scenarios for images and text classification (Guillaumin et al., 2010; Gong et al., 2016; Liu et al., 2020; Miyato et al., 2016; Xu et al., 2017), but it is heavily underexplored for extractive text summarization.

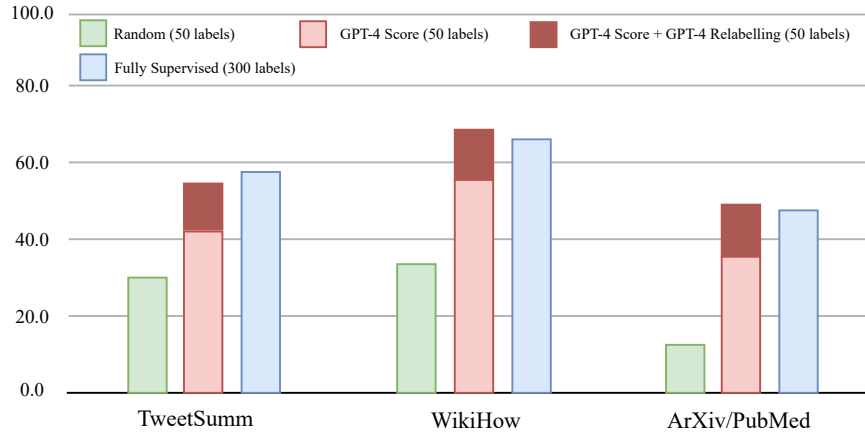


Figure 5.1: **L-Eval scores of semi-supervised v/s fully supervised models in a low-data setting.** The proposed semi-supervised method (middle column for each dataset) combines PreSumm’s confidence with GPT-4’s knowledge to generate better pseudo-labels. The proposed method performs competitively on the three summarization datasets while using $6\times$ fewer labels than a fully supervised method (rightmost column for each dataset)

Standard deep learning-based methods reformulate extractive summarization as a binary text classification problem: given a sentence in the input document, predict its likelihood of appearing in the output summary (Amini and Gallinari, 2001; Wong et al., 2008). SSL techniques, in particular, follow a teacher-student paradigm to train a classifier on the limited labeled data first to generate pseudo-labels for the unlabeled set and then use the same classifier’s confidence values to select the best pseudo-labels for the next training cycle. Most recently, (Liu and Lapata, 2019) and (Zhuang et al., 2023) employed this strategy using large language models (LLMs) like BERT (Devlin et al., 2019a) as their backbones. While these works have shown promising results, the underlying classification model is not suitable for determining the accuracy of pseudo-labels because it lacks specific fine-tuning for measuring pseudo-label accuracy. On the other hand, recent instruction-tuned LLMs like GPT-4 (OpenAI, 2023b) and LLaMA-2 and LLaMa-3 (Touvron et al., 2023) have been successfully used for generating pseudo-labels and evaluation for a wide range of tasks (Sahu et al., 2022, 2023a; Dziri et al., 2023; Mishra et al., 2023; Liu et al., 2023b; Zhang et al., 2023a).

In this chapter, we discuss our work, *PPSL*, a prompt-based pseudolabeling strategy for semi-supervised text summarization, which uses generative LLMs to improve the process of pseudo-labeling during the training phase of an SSL algorithm (Sahu et al., 2023b).

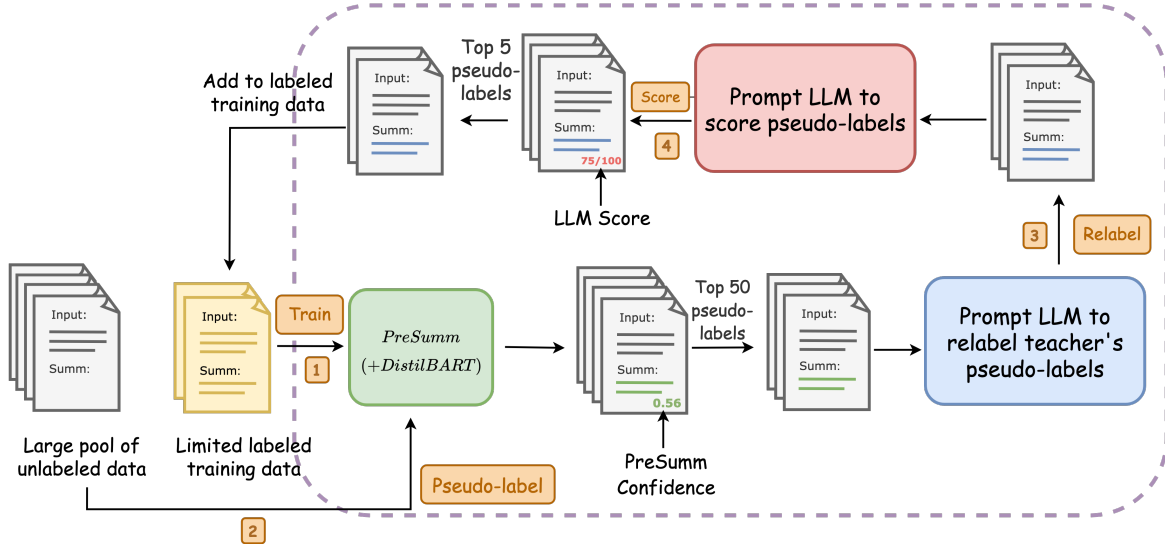


Figure 5.2: **PPSL pipeline**. Our approach has four main stages. **Step 1**: train a teacher model M on the limited labeled dataset. **Step 2**: generate pseudo-labels for the unlabeled set with M and shortlist 50 based on teacher confidence (see Equation 5.1). **Step 3**: prompt an LLM to summarize the shortlisted documents. **Step 4**: score the pseudo-labels in Stage 3 by prompting an LLM and select the top 5. These summaries are then added to the training data for the next cycle.

Scoring all the pseudo-labels with LLMs can quickly get expensive as the number of pseudo-labels increases, so we first use the classification model’s confidence values to shortlist a fixed number of pseudo-labels. Then, we prompt an LLM with specific instructions to score the pseudo-labels and select top- n pseudo-labels to include in the subsequent training cycle. Our method also includes a prompt-based relabeling mechanism that improves pseudo-labels’ accuracy. We evaluate our approach on three text summarization datasets: TweetSumm (Feigenblat et al., 2021), WikiHow (Koupaei and Wang, 2018), and ArXiv/PubMed (Cohan et al., 2018). We show that our method enhances the performance of a standard teacher-student pipeline to achieve competitive performance to that of a fully supervised model while using $6\times$ less labeled data in a data-scarce setting and outperforms fully supervised models in a data-abundant setting. We discuss the results in detail in Section 5.3, but figure 5.1 provides a quick demonstration of the effectiveness of our method compared to prior methods.

In summary, we 1) propose a prompt-based pseudo-label selection strategy for semi-supervised extractive text summarization. 2) We also propose a relabeling mechanism

using an LLM (such as GPT-4) to generate high-quality summaries for unlabeled examples. **3)** Finally, we show that our method is more sample-efficient and outperforms prior semi-supervised methods for extractive summarization on ROUGE-1, ROUGE-2, ROUGE-L, and L-Eval (same as G-Eval but using LLaMA-3 as the LLM) scores across three benchmark datasets.

5.1 Methodology

This section describes our *PPSL* approach for semi-supervised extractive text summarization. As shown in Figure 5.2, we employ a teacher-student training framework and divide our pipeline into four stages. **First**, we train a teacher model on the available labeled examples. **Second**, we use the trained model to generate pseudo-labels for the unlabeled set and shortlist 50 pseudo-labels based on the model’s confidence. **Third**, we prompt a generative LLM like GPT-4 to relabel the shortlisted pseudo-labels. **Lastly**, we prompt the LLM to score the new pseudo-labels and select the top 5 to include in the next training cycle. The following sections describe individual stages in more detail.

5.1.1 Stage 1: Training the Teacher Model

First, we train a fully-supervised model M (teacher) on the set of available labeled examples D_l . We use PreSumm (Liu and Lapata, 2019) as our summarization model M as it has been shown to perform well for extractive summarization. Notably, PreSumm reformulates the task of generating extractive summaries to binary classification, where, for each sentence in the input document, the model predicts if it will be present in the output summary. Then, the model combines the top- n sentences with the highest probabilities in their order of appearance in the original text to construct the extractive summary.

5.1.2 Stage 2: Generating Pseudo-labels using the Teacher Model

We use the teacher model M to generate pseudo-labels for the unlabeled set D_u . Next, we shortlist a subset of 50 pseudo-labels with the highest teacher confidence¹. We compute the teacher confidence for a generated summary (a.k.a pseudo-label) as follows: let p_{ij} denote the probability with which the i -th sentence s_i in an unlabeled document u_j is present in

¹We experiment 5, 10, 20, 50, and 75 and find 50 to be optimal.

its summary S_j , and let $\mathbb{1}$ denote the indicator function: $\mathbb{1}(s_i) = \begin{cases} 1, & \text{if } s_i \in S \\ 0, & \text{otherwise} \end{cases}$. We then compute the teacher confidence for the pseudo-label S_j by averaging the probabilities of selected sentences. We define the teacher confidence (C_j) for an input text u_j as follows:

$$C_j = \frac{\sum_{i=1}^{|u_j|} \mathbb{1}(s_i) \cdot p_{ij}}{n}, \quad (5.1)$$

where $|u_j|$ denotes the number of sentences in the unlabeled document u_j and n is the number of sentences in the generated summary.

We will show in Section 5.3 that shortlisting a subset of pseudo-labels helps make our method more sample-efficient, as we avoid relabeling a large unlabeled pool. This ultimately minimizes our LLM usage cost in the subsequent stages.

5.1.3 Stage 3: LLM Relabeling of Teacher’s Pseudo-labels

After selecting the top 50 pseudo-labels using teacher confidence defined in Equation 5.1, we prompt an LLM (GPT-4/LLaMA, in our case) to generate a summary for each shortlisted unlabeled example. This effectively relabels the pseudo-label from Stage 2. Specifically, we follow the prompt template in Figure C.4a when generating summaries using GPT-4, which uses the same mechanism as PreSumm, i.e., we instruct the LLM to output probabilities for each sentence in the input document and then concatenate the top- n lines in their order of appearance in the input text.

5.1.4 Stage 4: LLM Scoring of Pseudo-labels

In the last stage of our pipeline, we prompt LLaMA-3 as shown in Figure C.4b to output a rating between 0-100 for the pseudo-labels from Stage 3. Specifically, we list the following criteria for evaluating the pseudo-labels:

- The summary should be concise, ideally with less than k sentences.
- The summary should capture the key points from the input text. Do not penalize if it doesn’t include any minor additional details.
- The summary should be extractive, i.e., directly taken from the conversation without any modifications or removal of symbols.

Algorithm 1: An algorithm for prompt-based semi-supervised text summarization

Input : Labeled Data D_l , Unlabeled Data D_u , Number of cycles N_{cycles}

Output: Trained summarization model M

```

1  $M \leftarrow null$  ;
2 for  $i \leftarrow 1$  to  $N_{cycles}$  do
3   // Train PreSumm on labeled set;
4    $M \leftarrow PreSumm(D_l)$ ;
5   // Generate Pseudo-labels;
6    $Pseudos \leftarrow Top50(M(D_u))$ ;
7    $Pseudos \leftarrow Top5(LLM(Pseudos))$ ;
8   // Add top-5 pseudos to labeled set and update unlabeled set;
9    $D_l \leftarrow D_l \cup Pseudos$ ;
10   $D_u \leftarrow D_u \setminus Pseudos$ ;
11 end

```

- The response should contain only the numerical rating.

Here, k is a hyperparameter chosen based on the average summary length in the dataset. Finally, we choose the top 5 pseudo-labeled examples with the highest LLaMA scores and append them to the existing labeled set D_l for the next training cycle.

We repeat Stages 1-4 N_{cycles} times to improve the initial summarization model M and use the model obtained after the last cycle for generating summaries for the unseen test set. Algorithm 1 also provides an overview of the proposed semi-supervised approach. To note, $M(D_l)$ denotes generating summaries for D_l using M , and $LLM(Pseudos)$ denotes the prompt-based relabeling of pseudo-labels from Stage 2. Furthermore, $TopK(\cdot)$ denotes scoring pseudo-labels (based on teacher confidence in Stage 2 and LLM score in Stage 4) and selecting the top K pseudo-labels.

5.2 Experimental Setup

This section describes our experimental setup, including the datasets, training, and evaluation details, as well as the baselines used.

5.2.1 Datasets

We use three datasets for extractive summarization.

1) **TweetSumm** (Feigenblat et al., 2021) is a real-world customer service chat dataset. The dataset has 1100 conversations between a customer and an agent, and each conversation has three human-annotated extractive summaries. The training set has 858 dialogs, out of which we randomly sample a subset of 50 examples as the initial training set and treat the remaining as the unlabeled set. We use the official validation and test sets for evaluation.

2) **WikiHow** (Koupaei and Wang, 2018) contains WikiHow articles with their headlines as abstractive summaries. The dataset has over 180k articles, with around 168k training articles and 6000 test and validation articles.

3) **ArXiv/PubMed** (Cohan et al., 2018) is a collection of scientific articles from PubMed and ArXiv with their abstracts as their summaries. The dataset has close to 325k articles, with nearly 300k training articles and 12.5k test and validation articles.

	TweetSumm	WikiHow	ArXiv/PubMed
# Train	858	168,000	300,000
# Valid	100	6,000	12,500
# Test	100	6,000	12,500

Table 5.1: Statistics of the text summarization datasets we use in our experiments.

Table 5.1 summarizes the dataset statistics. Notably, the WikiHow and ArXiv/PubMed datasets do not have extractive labels. Therefore, we follow the same steps as the original PreSumm paper (Liu and Lapata, 2019) to obtain the ground truth extractive labels. Additionally, due to the large training corpus size for WikiHow and ArXiv/PubMed, we limit the pool of unlabeled examples by randomly sampling a subset of 1,000 articles in each training cycle. We choose the above three datasets to cover a diverse set of scenarios: TweetSumm contains real-world conversations and human-annotated extractive summaries, WikiHow contains medium-sized documents, and ArXiv/PubMed contains long documents.

5.2.2 Training and Evaluation Details.

Training. We use the TransformerSum repository² to implement our training pipeline. We use PreSumm as our teacher model M and experiment with two backbones: `distilbert-base-uncased` and `bert-base-uncased`. We perform experiments in two settings: **1)** data-scarce setting, where we fix the size of the labeled set D_l to 50 for all the datasets, and **2)** data-abundant setting, where we set the size of D_l to 500. We set N_{cycles} to 50 for all experiments. We add 5 pseudo-labels to the training set in each cycle, thus resulting in a final training set size of 300 (50 labeled + 250 pseudo-labeled examples.) We set the summary size k to 4 for TweetSumm and 8 for WikiHow and ArXiv/PubMed. We base these summary sizes on the average summary size of the labeled training set. For training, we start with a learning rate of 2×10^{-5} on all the datasets and use a cyclic learning rate scheduler during training (Smith, 2015), which is the default setting in TransformerSum. Additionally, we use AdamW as our optimizer with $\epsilon = 1 \times 10^{-8}$, $\beta_1 = 0.9$, $\beta_2 = 0.99$. We train all our models on a single V100 GPU with 12G VRAM. We repeat each experiment for three different seeds and report the mean and standard deviation in our results unless otherwise stated.

Evaluation. We evaluate the summary quality of the models using the following metrics:

ROUGE Scores. We use ROUGE-1 (R-1), ROUGE-2 (R-2), and ROUGE-L (R-L) F1 scores (Lin, 2004) for evaluation, where R-1 and R-2 measure the unigram and bigram overlap between the predicted and the ground truth summaries, respectively, while R-L also considers the order of n -grams. We use the `pyrouge` Python package to compute ROUGE scores in our setup and report them in Table 5.2.

L-Eval Scores. In addition to ROUGE, we use an LLM-based evaluation metric for our task. Specifically, we use LLaMA-Eval (L-Eval), an open-source variant of the G-Eval metric (Liu et al., 2023b), where we prompt LLaMA-3-70b-Instruct instead of a GPT model. We use L-Eval as it better aligns with human preferences for text summarization, compared to ROUGE scores and other model-based evaluation metrics, such as BERTScore and BARTScore (Zhang et al., 2019; Yuan et al., 2021). It is also not biased towards LLM-generated content; however, since LLM-inference speed is low for long documents, we did not compute L-Eval scores during training and only computed them during final testing. When computing L-Eval scores, we provide the LLM with the input article and a (generated) summary and instruct it to score the summary on a scale of 1-10 (see Prompt C.2

²<https://transformersum.readthedocs.io/en/latest/>

Method	TweetSumm			WikiHow			ArXiv/Pubmed		
	R-1 (%)	R-2 (%)	R-L (%)	R-1 (%)	R-2 (%)	R-L (%)	R-1 (%)	R-2 (%)	R-L (%)
DistilBERT_{base} (50 labels)									
Random	36.7 (1.5)	25.4 (1.4)	36.7 (1.3)	19.7 (1.4)	1.5 (1.1)	7.2 (1.3)	19.5 (1.3)	2.9 (0.9)	7.8 (1.2)
Confidence	43.5 (1.4)	35.1 (1.2)	46.8 (1.1)	21.3 (0.4)	3.7 (0.8)	10.3 (1.0)	23.4 (1.1)	5.2 (0.7)	12.5 (1.1)
+ G-4 relabeling	55.4 (1.3)	46.7 (0.6)	56.1 (0.9)	22.1 (0.4)	5.7 (0.6)	13.5 (0.7)	23.8 (0.8)	7.3 (1.3)	15.3 (0.8)
Confidence + G-4 score	46.8 (1.3)	37.4 (0.4)	48.3 (1.2)	21.7 (0.9)	4.6 (0.4)	12.1 (1.1)	24.1 (0.9)	6.7 (0.3)	13.8 (1.4)
+ G-4 relabeling (Ours)	57.6 (1.2)	46.3 (1.7)	56.2 (1.3)	22.7 (0.3)	5.9 (0.4)	13.8 (0.5)	24.7 (0.9)	8.1 (1.3)	15.9 (0.8)
Confidence + L-3 score	45.7 (1.1)	36.9 (0.2)	47.8 (1.2)	21.6 (0.4)	4.1 (0.5)	11.1 (0.8)	23.9 (0.9)	6.1 (0.3)	12.9 (1.3)
+ L-3 relabeling (Ours)	<u>56.2 (1.1)</u>	45.1 (1.2)	55.9 (1.1)	<u>22.3 (0.1)</u>	<u>5.8 (0.2)</u>	<u>13.6 (0.3)</u>	<u>24.5 (0.6)</u>	<u>7.7 (1.4)</u>	<u>15.7 (0.3)</u>
BERT_{base} (50 labels)									
TSL (500:500)	49.0	37.7	48.2	-	-	-	-	-	-
Random	45.4 (1.4)	32.4 (1.9)	42.5 (1.8)	22.1 (1.7)	2.4 (1.5)	9.6 (1.5)	23.3 (1.4)	6.1 (1.2)	12.4 (1.3)
Confidence	49.7 (1.6)	39.5 (1.4)	49.4 (1.3)	24.5 (0.6)	4.8 (1.1)	12.8 (1.0)	27.6 (1.1)	7.7 (0.7)	14.2 (1.2)
+ G-4 relabeling	57.8 (1.2)	50.3 (0.5)	58.9 (1.2)	26.4 (0.3)	7.3 (0.5)	16.4 (0.8)	28.7 (0.9)	9.5 (1.1)	17.1 (0.8)
Confidence + G-4 score	52.3 (1.6)	42.8 (0.7)	51.0 (1.4)	25.2 (0.7)	5.6 (0.5)	13.1 (1.0)	27.7 (0.9)	7.9 (0.2)	15.5 (1.3)
+ G-4 relabeling (Ours)	58.9 (1.4)	50.4 (0.8)	59.4 (1.5)	26.1 (0.4)	7.2 (0.6)	15.9 (0.9)	29.1 (0.7)	9.7 (1.2)	17.7 (0.6)
Confidence + L-3 score	51.7 (1.2)	41.6 (1.2)	50.3 (1.2)	25.9 (0.3)	5.2 (0.2)	13.0 (0.8)	27.6 (0.4)	7.9 (0.5)	15.3 (1.1)
+ L-3 relabeling (Ours)	<u>58.4 (1.2)</u>	50.1 (0.3)	59.1 (1.2)	26.0 (0.2)	6.9 (0.3)	15.1 (0.2)	<u>29.0 (0.5)</u>	<u>9.4 (0.7)</u>	<u>17.4 (0.3)</u>
BERT_{base} (500 labels)									
TSL (500:500)	59.0	48.3	58.2	-	-	-	-	-	-
Random	55.1 (1.4)	42.7 (1.1)	50.3 (1.2)	25.6 (1.3)	4.5 (1.1)	15.2 (1.3)	25.4 (1.5)	9.5 (1.2)	24.1 (1.2)
Confidence	61.8 (0.7)	54.9 (0.8)	60.3 (0.9)	28.4 (0.6)	8.0 (1.1)	22.5 (1.0)	29.4 (0.5)	11.5 (0.6)	27.7 (0.8)
+ L-3 score	63.4 (0.5)	55.6 (0.8)	62.1 (0.7)	28.9 (0.4)	8.2 (0.5)	28.4 (0.4)	31.7 (0.3)	11.8 (0.2)	29.4 (0.4)
+ L-3 relabeling (Ours)	64.2 (0.2)	56.2 (0.4)	62.8 (0.6)	30.7 (0.4)	8.8 (0.3)	29.5 (0.3)	33.5 (0.3)	12.3 (0.2)	32.2 (0.3)

Table 5.2: Mean (Std.) ROUGE F-1 scores of different pseudo-labeling strategies. R-1, R-2, and R-L denote ROUGE-1, ROUGE-2, and ROUGE-L metrics, respectively. TSL results from (Zhuang et al., 2023). Refer to Section 5.2.3 for method details. **Bold** indicates the best-performing and underline denotes the second-best performing method, respectively.

in Appendix C for the full L-Eval prompt template). Formally, given a test article A and a summary s , we compute the L-Eval score as follows:

$$\text{L-Eval}(A, s) = \sum_{r=1}^{10} p_r \cdot r, \quad (5.2)$$

where p_r is the probability of generating the rating r . In practice, we can only look at the probabilities of top-5 tokens for LLaMA-3-70b-Instruct, so we assign a probability of 0 to the remaining ratings (that did not appear in the top-5).

We use a metric like G-Eval as it substantially surpasses ROUGE scores and previous model-based evaluation metrics, such as BERTScore (Zhang et al., 2019) and BARTScore (Yuan et al., 2021), for summarization. More importantly, it is shown to be better aligned with human preferences than the existing automated evaluation metrics for text summarization. Finally, G-Eval has been shown to not exhibit bias towards GPT-generated content.

5.2.3 Baselines

We compare our method with the following baselines: **1) PreSumm** (Liu and Lapata, 2019). The original PreSumm model that pretrains a BERT model for summarization. We train two PreSumm models – one on a limited training set with 50 labeled examples to match the starting point of our semi-supervised setting and another with 300 labeled examples, the same as the dataset size at the end of our training cycle. **2) Teacher-Student Learning (TSL)** (Zhuang et al., 2023). Current state-of-the-art semi-supervised method on TweetSumm. The teacher-student learning framework uses a similar formulation for computing model confidence to ours, as follows: $C_j = \sum_{i=1}^n (C_{ij})/n_j$. Here, $C_{ij} = p_{ij}q_{ij} + (1 - p_{ij})(1 - q_{ij})$, where p_{ij} is the probability of sentence i being selected for summary for dialog j estimated by the teacher model, and $q_{ij} = 1$ if p_{ij} in top 4, else 0. We report the performance of the TSL (50:500) and TSL (500:500) models from the paper, as they are the closest to our setup (50/500 labeled examples + 500 unlabeled examples). **3) Confidence + G-4 relabeling + G-4 score (Ours)**. Our proposed method following the methodology in Section 5.1. We first use the PreSumm teacher model to shortlist 50 pseudo-labels (Stage 1 and 2), relabel them using GPT-4 (Stage 3), and then select the top 5 using GPT-4 score (Stage 4). **4) Confidence + G-4 score**. We skip Stage 3 from **3)** to directly score the top 50 PreSumm pseudo-labels using GPT-4. We run this baseline to measure the effect of relabeling in our pipeline. **5) Confidence + G-4 relabeling**. We skip Stage 4 from **3)** and select the final 5 pseudo-labels based on PreSumm confidence. **6) Confidence + L-3 relabeling + L-3 score (Ours)**. Same as **3)** but using LLaMA-3. **7) Confidence + L-3 score (Ours)**. Same as **4)** but using LLaMA-3. **8) Confidence**. We skip Stage 3 and 4 from **3)** and select 5 PreSumm pseudo-labels based on PreSumm confidence. **9) Random**. Same as **6)** but instead of using teacher confidence defined in Equation 5.1, we randomly select five PreSumm pseudo-labels to include in each cycle.

5.3 Results

We now discuss results in Table 5.2 and Table 5.3. Table 5.2 reports the ROUGE scores for different pseudo-labeling strategies for the three datasets. All the methods operate in (50:250) or (500:250) setups, except TSL (50:500), which uses 500 unlabeled examples. Notably, Table 5.2 has two blocks of results, one for DistilBERT_{base} as the backbone of the PreSumm model and another for BERT_{base} as the PreSumm backbone. Table 5.3 compares L-Eval scores of our semi-supervised approach with fully-supervised setups.

Method	TweetSumm		WikiHow		ArXiv/Pubmed	
	R-2	L-Eval	R-2	L-Eval	R-2	L-Eval
PreSumm (50 labels)	37.1 (1.1)	31.2 (0.5)	3.2 (0.8)	34.2 (1.5)	7.3 (0.9)	13.5 (1.2)
PreSumm (300 labels)	51.1 (2.1)	60.5 (1.2)	7.6 (0.6)	68.1 (1.1)	10.8 (0.9)	49.5 (2.4)
PreSumm (500 labels)	54.4 (1.2)	67.1 (0.3)	7.9 (0.5)	74.4 (0.6)	11.3 (0.5)	58.2 (1.1)
PreSumm (750 labels)	<u>56.1 (0.7)</u>	<u>70.3 (0.5)</u>	<u>8.5 (0.4)</u>	<u>76.5 (0.4)</u>	<u>12.1 (0.7)</u>	<u>62.8 (0.7)</u>
50 labels						
Random	32.4 (1.9)	32.1 (1.1)	2.4 (1.5)	37.7 (1.6)	6.1 (0.2)	15.1 (2.3)
Confidence + G-4 score	42.8 (0.7)	46.2 (0.2)	5.6 (0.5)	59.4 (1.3)	7.9 (0.2)	40.1 (1.9)
+ G-4 relabeling (Ours)	50.4 (0.8)	58.4 (0.4)	7.2 (0.6)	70.3 (1.4)	9.7 (1.2)	52.5 (1.3)
Confidence + L-3 score	41.6 (1.2)	45.8 (0.7)	5.2 (0.2)	57.5 (1.4)	7.9 (0.5)	37.1 (1.8)
+ L-3 relabeling (Ours)	50.1 (0.3)	56.3 (0.9)	6.9 (0.3)	69.3 (2.1)	9.4 (0.7)	49.3 (1.4)
500 labels						
Random	42.7 (1.1)	52.3 (1.2)	4.5 (1.1)	52.7 (1.8)	9.5 (1.2)	44.1 (0.9)
Confidence + L-3 score	55.6 (0.8)	69.2 (0.7)	8.2 (0.5)	75.2 (1.4)	11.8 (0.2)	60.2 (0.5)
+ L-3 relabeling (Ours)	56.2 (0.8)	71.2 (0.9)	8.8 (0.3)	77.3 (1.3)	12.3 (0.2)	65.7 (0.3)

Table 5.3: **Fully-supervised methods (first two rows) semi-supervised approaches (remaining rows).** All models use BERT_{base} as PreSumm’s backbone. The number of labeled examples for fully supervised models is shown in brackets. The semi-supervised methods use 50/500 labeled and 250 unlabeled examples.

5.3.1 Comparison of Pseudo-label Selection Strategies

Referring to Table 5.2, we note that all pseudo-label selection strategies outperform the random baseline. The “Random” baseline performs worse than the fully supervised counterpart on all datasets (R-2 in Table 5.2 v/s R-2 in Table 5.3), meaning that the *majority* of the shortlisted PreSumm pseudo-labels are low-quality. Using teacher confidence leads to slight performance gains on all the datasets, and adding GPT-4 score further improves the results (“Confidence” v/s “Confidence + G-4 score” in Table 5.2). These improvements indicate that the shortlisted PreSumm pseudo-labels include some good-quality pseudo-labels, too, and using GPT-4 to rate those pseudo-labels is crucial to picking them. We see similar trends when using LLaMA-3.

To further confirm our findings, we conduct a qualitative study in the data-scarce setup, where we compute the ROUGE scores of the final 5 pseudo-labels for each method against the respective ground truth summaries, and Figure 5.3 shows the mean ROUGE-2 of the five selected pseudo-labels. To clarify, we obtained the “Oracle” results by directly

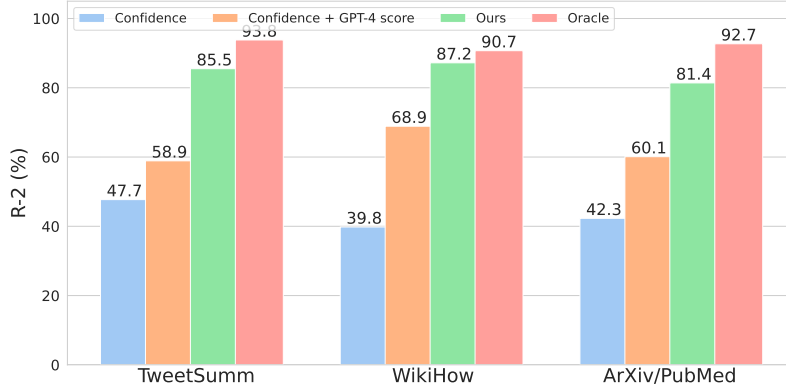


Figure 5.3: **Quality of pseudo-labels by different strategies (data-scarce setup).** The y-axis denotes the ROUGE-2 scores of the top 5 pseudo-labels computed against the respective ground truths. All results are for BERT_{base} as the backbone for PreSumm and three random seeds. Refer to Section 5.3.1 for complete details.

selecting the final 5 pseudo-labels using ROUGE-2 scores computed against the ground truth. We note a stark difference between “Confidence” and “Oracle,” which shows that relying solely on teacher confidence consistently leads to a selection of low-quality pseudo-labels. Combining GPT-4 score with teacher confidence is effective (“Confidence + G-4 score”), and adding the GPT-4 relabeling greatly boosts the quality of selected pseudo-labels (“Ours”).

5.3.2 Effect of Relabeling

Referring to Tables 5.2 and 5.3, we observe that relabeling with LLMs leads to a significant boost in the summarization performance in terms of both ROUGE scores and L-Eval. When using BERT_{base} as the backbone, we note that the ROUGE-1 improves from 52.3 to 58.9 on the TweetSumm dataset, 25.2 to 26.1 on the WikiHow dataset, and 27.7 to 29.1 on the ArXiv/PubMed dataset. GPT-4 relabeling is also effective when using teacher confidence without GPT-4 score (“Confidence” v/s “Confidence + G-4 relabeling”). Our previous qualitative study also supports these results, showing that relabeling improves the quality of pseudo-labels. We observe similar trends when using DistilBERT_{base} as PreSumm’s backbone and LLaMA-3 instead of GPT-4. When using 500 labels, we note boosts in performance but the relative scale is smaller compared to when using 50 labels.

We conduct additional testing to analyze the performance of our best- and second-best-

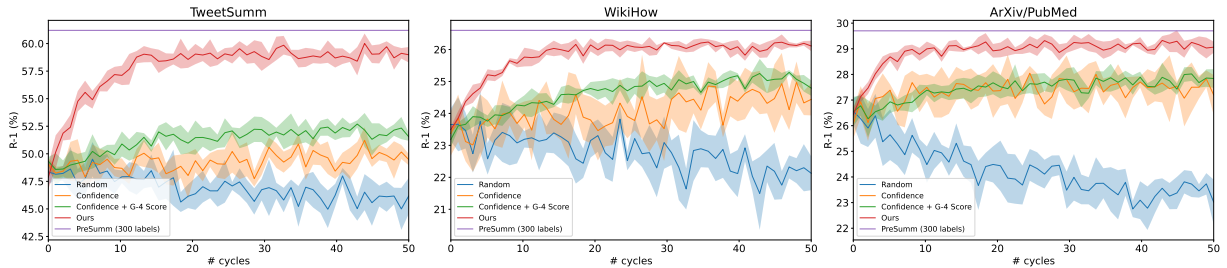


Figure 5.4: **ROUGE-1 curves v/s # cycles for data-scarce setting.** Each cycle denotes an addition of 5 new pseudo-labels to the training set. All results use BERT_{base} as the backbone for PreSumm. The curves are averaged for three seeds (the width denotes the std). Note that we report the GPT-4 version of our method here.

performing models, both of which involve relabeling. We find that the p-value < 0.016 for Welch’s test for an R-1 of 58.9 (1.4) for “Confidence + G-4 score + G-4 relabeling” v/s 57.8 (1.2) for “Confidence + G-4 relabeling” on the TweetSumm dataset, denoting the differences are significant.

5.3.3 On Sample Efficiency of Different Approaches

Referring to Table 5.3, we now analyze the sample efficiency of different learning algorithms. For fully supervised methods, we note that including more labeled examples improves L-Eval and ROUGE scores across the board (“PreSumm (50 labels)” v/s “PreSumm (300 labels)”). Our semi-supervised approach, using 50 labels with GPT-4 relabeling and GPT-4 score, achieves competitive performance compared to the fully supervised PreSumm model trained on 300 labels. Notably, we get better L-Eval scores than “PreSumm (300 labels)” on WikiHow and ArXiv/PubMed datasets and are competitive on TweetSumm. This is encouraging, as “PreSumm (300 labels)” approximates the best-case scenario of 100% high-quality labels in the training set. In the data-abundant setting, our proposed method with LLaMA outperforms the respective fully supervised model in terms of both ROUGE and L-Eval.

We plot the R-1 scores against the number of training cycles, as shown in Figure 5.4. Overall, “Random” setting is highly unstable, “Confidence + G-4 score” slightly improves over “Confidence” on TweetSumm and WikiHow, but more importantly, it is consistently more stable. Finally, our method with GPT-4 scoring and relabeling not only significantly boosts the R-1 scores (a visible gap between “Ours” and the rest) but also does so at a much faster rate. For all the datasets, our method peaks and stabilizes under 20 cycles

(100 pseudo-labels), further endorsing the sample efficiency of our method compared to other approaches.

5.4 Limitations

Our results indicate a promising potential to use LLMs for semi-supervised extractive text summarization; however, this work has a few limitations, as detailed in this section.

First, we rely completely on LLMs pseudo-labels for scoring and relabeling, which can be erroneous and we do not currently have a way to automatically judge them. We need a human in the loop to judge the quality of such pseudo-labels.

Second, we can also potentially improve the sample efficiency of the model. Currently, we relabel all the top 50 pseudo-labels selected by PreSumm in Stage 3. This is cheaper than pseudo-labeling all the pseudo-labels, but there’s a chance that LLM relabels an already good-quality summary, as we note from our analysis that the initial shortlist contains some good pseudo-labels. Therefore, we can develop an additional strategy to decide whether to relabel a given pseudo-label. Such a method would further reduce costs for using closed-source models like GPT-4. We hope this work furthers research toward incorporating LLMs to address such problems for low-resource semi-supervised tasks.

Lastly, we want to highlight again that we use GPT and LLaMA-3 to generate new examples, and even though they are instruction-tuned, some generations might depict undesirable biases with respect to the current objective. For this reason, we recommend using our proposed models with human monitoring when employing the method for real-world settings.

5.5 Conclusion

To conclude, we propose a novel prompt-based pseudo-labeling strategy for semi-supervised extractive text summarization. Our method effectively combines teacher confidence and knowledge of LLMs like GPT-4 to improve pseudo-labeling in low-resource summarization setups. Adding an LLM-based critic to relabel and rank pseudo-labels leads to highly performant summarization models that are competitive with their fully supervised counterparts while utilizing $6\times$ fewer labeled examples.

5.6 Key Takeaways

- Prompt-based pseudo-labeling combined with an LLM-based critic is effective for semi-supervised extractive text summarization, leading to highly performant summarization models that are competitive with their fully supervised counterparts while utilizing $6\times$ fewer labeled examples.
- Relabeling with LLMs significantly boosts the summarization performance in terms of both ROUGE scores and L-Eval and significantly improves the sample efficiency of the approach.

In the next chapter, we move to a more challenging setup of few-shot summarization, where we do not have access to the unlabeled examples, and also consider abstractive summarization in addition to extractive summarization.

Chapter 6

LLM-based Data Augmentation for Low-resource Text Summarization

In Chapters 3 and 4, we demonstrated that data augmentation (DA) can be successfully used to address data scarcity, mitigate data annotation costs, and enhance model robustness for text classification. In the previous chapter, we explored the data-scarce setup of semi-supervised text summarization and showed that pseudo-labeling can be effectively used to improve the summarization model’s performance; however, real-world scenarios might not always have a large pool of unlabeled data. Some existing approaches study LLM-based data augmentation techniques (Dai et al., 2023; Ding et al., 2024) that can generate contextually rich textual augmentations to enhance model performance in various generative NLP tasks such as dialog modeling and grammatical error correction (Chintagunta et al., 2021; Wan et al., 2022; Feng et al., 2021; Wang et al., 2022) that use state-of-the-art LLMs like GPT-4 (Achiam et al., 2023), LLaMA-3 (Touvron et al., 2023), and Claude-3 (Anthropic, 2024); however, data augmentation for text summarization has remained an underexplored area of research. In this chapter, we explore the low-resource setup of text summarization and study the effectiveness of LLM-based data augmentation for the task.

To address the previous limitations, we propose *MixSumm*, an LLM-based data augmentation technique for few-shot text summarization (Sahu and Laradji, 2024). Specifically, *MixSumm* is a two-stage prompt-based data augmentation approach that **first** instructs an LLM to synthesize diverse documents by mixing topical information derived from a small set of seed documents, and **then** generates summaries for the synthesized documents.

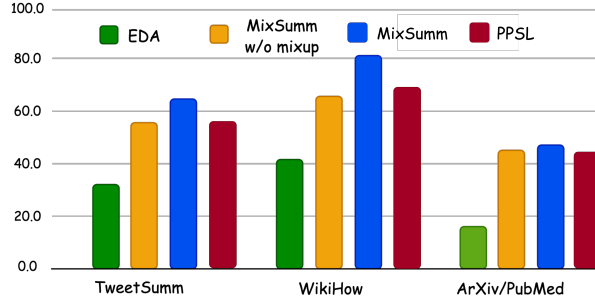


Figure 6.1: **L-Eval scores of different methods on low-resource extractive text summarization.** The proposed *MixSumm* approach generates new documents by combining topics from multiple examples and outperforms other methods, including a strong LLM-based DA method (*MixSumm* w/o mixup) and a prompt-based semi-supervised approach (PPSL).

To evaluate the effectiveness of our method, we conduct extensive experiments on the TweetSumm (Feigenblat et al., 2021), the WikiHow (Koupaei and Wang, 2018), and the ArXiv/PubMed (Cohan et al., 2018) text summarization datasets. We also use the open-source LLaMA-3-70b-Instruct LLM for all our experiments instead of a closed-source LLM like the GPT family of LLMs to enhance the reproducibility of our findings while avoiding expensive API costs for accessing closed-source models. For evaluation, we use the standard ROUGE scores (Lin, 2004) as well as L-Eval, an open-source version of the promising LLM-based evaluator for text summarization, G-Eval (Liu et al., 2023b). Our experiments demonstrate that *MixSumm* outperforms strong data augmentation and semi-supervised baselines for low-resource summarization setups *and* we show a knowledge distillation effect, where the knowledge of a LLaMA-3-70b model is distilled into a much smaller summarization model using BERT_{base} and DistilBART backend (with 110M and 306M parameters, respectively).

To summarize the contributions of our work: **1)** we propose *MixSumm*, a novel prompt-based data augmentation framework for the challenging low-resource setup of 50-shot text summarization, **2)** we conduct comprehensive experiments to demonstrate the effectiveness of our approach over other state-of-the-art methods like *PPSL*, **3)** we show the effectiveness of LLaMA-3-70b-Instruct, an open-source LLM, instead of using expensive closed-source LLMs like GPT-4, and **4)** we demonstrate effective knowledge distillation from LLaMA-3-70B (70B parameters) to BERT and DistilBART-based summarization models with 110M and 306M parameters.

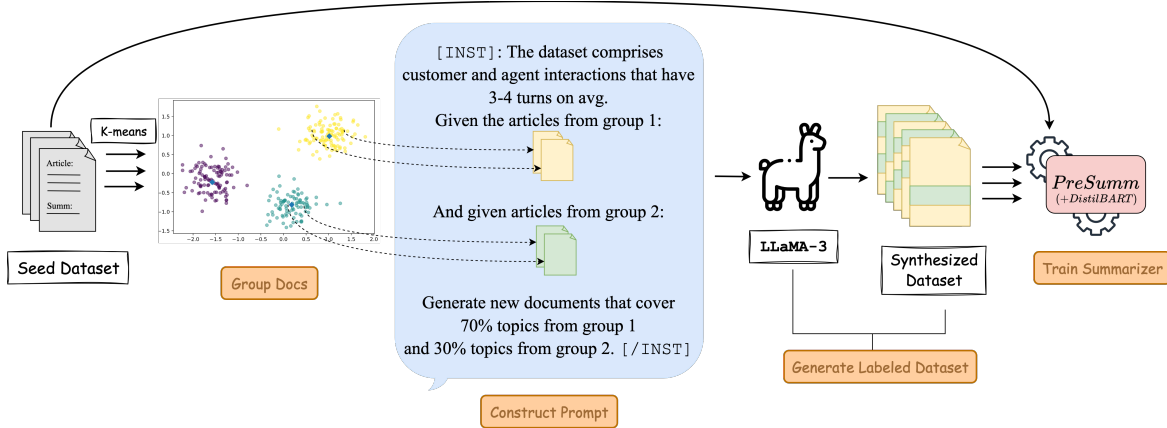


Figure 6.2: **MixSum** pipeline. We first group the documents into T groups using the k -means algorithm. Then, we construct the prompt for LLaMA-3-70b-Instruct by including documents from different groups and instructing the LLM to mix information from multiple topics when generating the new documents. Finally, we train a PreSumm extractive summarizer (Liu and Lapata, 2019) on the combined seed and the synthesized dataset. For abstractive summarization, we add a DistilBART model after PreSumm.

6.1 Notations

We denote an annotated, many-shot summarization dataset as $\mathcal{D} = (d_i, s_i) \forall i \in \{1, \dots, N\}$ where (d_i, s_i) denotes the i -th datapoint with input text document d_i and its ground truth summary s_i . We refer to the training, validation, and testing parts of the dataset as $\mathcal{D}_{train}$, $\mathcal{D}_{val}$, and $\mathcal{D}_{test}$, respectively. Given the many-shot training set $\mathcal{D}_{train}$, we construct a few-shot version of the dataset with k examples $\mathcal{D}_{F,train}$ as follows:

Step 1. Given $\mathcal{D}_{F,train}$, we group the training articles by topics. We do not define the topics explicitly and identify T groups by applying the k -means algorithm on the document embeddings (where $k = T$)¹. We use the SBERT encodings (Reimers and Gurevych, 2019) of the input documents as document embeddings². If an input document exceeds SBERT’s context window length of 512 tokens (roughly 300-400 English words), we chunk the document into smaller pieces and then average the chunk embeddings to obtain the final document embedding.

¹to clarify, k in k -means is different from k in k -shot

²sentence-transformers/all-mpnet-base-v2 model was used from the sentence-transformers library

Step 2. We construct our k -shot dataset $\mathcal{D}_{F,train}$ by randomly sampling an equal number of data points from each of the T clusters so that $\mathcal{D}_{F,train}$ has k examples in total. In Section 6.4, we empirically show that our principled approach for constructing few-shot datasets is better than randomly sampling k examples from $\mathcal{D}_{train}$ as it provides better topical coverage.

Problem Formulation. Given a text summarization dataset $\mathcal{D}$: **1)** perform data augmentation on $\mathcal{D}_{F,train}$ to synthesize a labeled dataset $\mathcal{D}_{A,train}$, and **2)** train a text summarization model on the combined dataset $\mathcal{D}_{F+A,train}$.

6.2 Methodology

We now describe *MixSumm*, a two-step approach for synthesizing labeled summarization documents. **First**, we instruct an LLM to generate documents that cover multiple topics derived from a small set. **Next**, we instruct an LLM to generate summaries for those documents. The following sections describe our two-step procedure in detail.

Step 1: Synthesizing New Documents. *First*, for every dataset, we manually write a short description detailing the type and approximate size of articles in the dataset. These descriptions enable our approach to be used in even zero-shot settings. *Next*, we construct T pairs of clusters $(c_i, c_j) \forall i, j \in 1, \dots, T, i \neq j$, such that c_j is the most distant cluster from c_i . We use the centroids of the clusters obtained during k -means clustering in Section 6.1 for our computation. We also ensure that all cluster pairs are *unique* as $(c_i, c_j) \equiv (c_j, c_i)$. *Finally*, we combine the dataset description with k examples from each cluster and instruct the LLM to generate new examples that cover topics from both clusters. Specifically, we instruct the LLM to generate examples that contain $\alpha\%$ topics from the first cluster c_i and $(1 - \alpha)\%$ topics from the second cluster c_j , where α is sampled from a uniform distribution between 1-100. This is similar to applying the mixup algorithm (Zhang et al., 2018) in a natural language space and has proven highly effective for data augmentation in low-resource text classification setups (Yoo et al., 2021; Sahu et al., 2023a). Prompt C.1 in Appendix C shows the complete template for this step.

Step 2: Generating Summaries for the Synthesized Documents. Next, we instruct the LLM to generate extractive and abstractive summaries for the synthesized documents. **For extractive summaries**, we provide a generated document to the LLM and

then instruct it to output a probability score for each sentence indicating whether that sentence should be included in the summary or not. We then rank the lines by the scores and choose the top- p lines, where p is the summary size and depends on the dataset. We truncate the input document if it exceeds the LLM’s context window length. This approach ensures the extractiveness of the generated summary labels as it mimics PreSumm (Liu and Lapata, 2019), a strong baseline for extractive text summarization. **For abstractive summaries**, instead of passing the entire source document and prompting the LLM to generate a summary, we follow an *extractive-then-abstractive* approach and add a DistilBART model to our pipeline that summarizes the previously generated extractive summaries. This approach is faster than passing the source document and summarizing, as our input context is significantly smaller. More importantly, it enhances the factual correctness of the summaries.

We separate the summarization step from the generation step to **1)** simplify the task of the LLM in one prompt, and **2)** avoid exceeding the context window length of the generative LLM used ³. After generating the new documents, we train a summarization model on the *MixSumm*-generated data and the original seed data. Specifically, we fine-tune a PreSumm model for the extractive task and a DistilBART model for the abstractive task.

6.3 Experimental Setup

6.3.1 Datasets

We use the same benchmark of text summarization datasets as the previous chapter to evaluate our method: TweetSumm (Feigenblat et al., 2021), WikiHow (Koupaei and Wang, 2018), and ArXiv/PubMed (Cohan et al., 2018) datasets. We refer the reader to Section 5.2.1 for a detailed description of the datasets.

We follow the same approach as the previous chapter to obtain the extractive labels for WikiHow and Arxiv/Pubmed datasets and report the training implementation details in Section 6.3.3.

³8192 tokens or roughly 5000-6700 English words for LLaMA-3-70b-Instruct used in our experiments

Method	TweetSumm				WikiHow				ArXiv/Pubmed			
	R-1 (%)	R-2 (%)	R-L (%)	L-Eval (%)	R-1 (%)	R-2 (%)	R-L (%)	L-Eval (%)	R-1 (%)	R-2 (%)	R-L (%)	L-Eval (%)
<i>Extractive Summarization</i>												
<i>Oracle</i>	65.7 $\pm$ 0.3	56.6 $\pm$ 0.4	64.9 $\pm$ 0.4	86.2 $\pm$ 0.3	30.5 $\pm$ 0.5	8.7 $\pm$ 0.3	19.2 $\pm$ 0.6	87.3 $\pm$ 0.1	34.6 $\pm$ 0.4	12.4 $\pm$ 0.2	19.6 $\pm$ 0.4	78.1 $\pm$ 0.5
TSL (50:500)	49.0	37.7	48.2	-	-	-	-	-	-	-	-	-
TSL (500:500)	59.0	48.3	58.2	-	-	-	-	-	-	-	-	-
EDA	51.1 $\pm$ 0.7	39.2 $\pm$ 0.9	53.0 $\pm$ 0.2	34.3 $\pm$ 1.2	23.4 $\pm$ 0.5	4.1 $\pm$ 0.3	13.0 $\pm$ 0.5	42.1 $\pm$ 0.8	26.2 $\pm$ 1.1	7.9 $\pm$ 1.0	13.1 $\pm$ 0.6	17.2 $\pm$ 0.5
PPSL (50:250)	58.4 $\pm$ 1.2	50.1 $\pm$ 0.3	59.1 $\pm$ 1.2	56.3 $\pm$ 0.9	26.0 $\pm$ 0.2	6.9 $\pm$ 0.3	15.1 $\pm$ 0.2	69.3 $\pm$ 2.1	29.0 $\pm$ 0.5	9.4 $\pm$ 0.7	17.4 $\pm$ 0.3	49.3 $\pm$ 1.4
<i>MixSumm</i> (rand.)	58.6 $\pm$ 3.2	50.6 $\pm$ 2.1	59.7 $\pm$ 2.3	60.3 $\pm$ 0.9	26.4 $\pm$ 1.0	7.5 $\pm$ 1.2	15.8 $\pm$ 0.2	72.5 $\pm$ 1.2	30.7 $\pm$ 1.7	10.6 $\pm$ 1.5	18.5 $\pm$ 1.1	48.4 $\pm$ 1.1
w/o Aug.	49.4 $\pm$ 0.7	36.9 $\pm$ 1.0	49.0 $\pm$ 0.2	31.5 $\pm$ 0.5	21.3 $\pm$ 0.4	3.2 $\pm$ 0.4	11.4 $\pm$ 0.5	34.2 $\pm$ 1.5	23.4 $\pm$ 1.1	7.5 $\pm$ 1.4	12.3 $\pm$ 0.8	13.5 $\pm$ 1.2
<i>MixSumm</i> (ours)	59.1 $\pm$ 1.7	52.7 $\pm$ 1.6	60.5 $\pm$ 1.3	65.3 $\pm$ 1.2	27.3 $\pm$ 2.1	7.8 $\pm$ 1.3	16.6 $\pm$ 1.8	81.1 $\pm$ 1.7	31.2 $\pm$ 1.2	10.7 $\pm$ 1.1	18.3 $\pm$ 1.1	53.1 $\pm$ 0.5
w/o Mixup	56.1 $\pm$ 1.1	47.3 $\pm$ 1.2	55.3 $\pm$ 1.1	57.3 $\pm$ 0.5	25.7 $\pm$ 1.4	6.2 $\pm$ 1.2	14.7 $\pm$ 0.7	67.3 $\pm$ 2.1	28.4 $\pm$ 1.9	8.3 $\pm$ 1.3	16.8 $\pm$ 1.6	52.3 $\pm$ 1.2
w/o Aug.	50.1 $\pm$ 0.6	38.1 $\pm$ 1.0	49.9 $\pm$ 0.6	32.3 $\pm$ 3.1	21.9 $\pm$ 0.3	3.5 $\pm$ 0.2	12.1 $\pm$ 0.9	33.3 $\pm$ 1.7	24.1 $\pm$ 0.9	7.9 $\pm$ 1.0	12.7 $\pm$ 0.5	19.0 $\pm$ 2.5
LLaMA-3 (0-shot)	50.3 $\pm$ 0.5	47.7 $\pm$ 0.4	49.9 $\pm$ 0.3	52.3 $\pm$ 1.2	12.2 $\pm$ 0.2	2.7 $\pm$ 0.5	8.1 $\pm$ 0.4	32.3 $\pm$ 0.3	23.6 $\pm$ 0.2	4.6 $\pm$ 0.7	15.4 $\pm$ 0.3	38.4 $\pm$ 0.5
LLaMA-3 (1-shot)	51.7 $\pm$ 0.2	49.2 $\pm$ 0.3	51.9 $\pm$ 0.3	58.7 $\pm$ 1.1	14.3 $\pm$ 0.2	4.1 $\pm$ 0.5	10.6 $\pm$ 0.2	39.4 $\pm$ 0.5	32.6 $\pm$ 0.4	6.5 $\pm$ 0.7	17.2 $\pm$ 0.3	38.3 $\pm$ 1.8
LLaMA-3 (5-shot)	62.4 $\pm$ 0.5	54.3 $\pm$ 0.7	60.3 $\pm$ 1.1	67.5 $\pm$ 0.6	28.7 $\pm$ 0.3	7.5 $\pm$ 0.9	17.1 $\pm$ 0.3	71.3 $\pm$ 0.4	-	-	-	-
<i>Abstractive Summarization</i>												
<i>Oracle</i>	44.7 $\pm$ 0.2	20.1 $\pm$ 0.4	36.8 $\pm$ 0.2	72.3 $\pm$ 0.6	28.7 $\pm$ 0.3	6.2 $\pm$ 0.7	13.6 $\pm$ 0.4	78.4 $\pm$ 0.8	28.4 $\pm$ 0.2	10.2 $\pm$ 0.4	15.8 $\pm$ 0.8	64.3 $\pm$ 0.5
EDA	41.5 $\pm$ 1.2	15.0 $\pm$ 0.8	32.2 $\pm$ 1.1	44.2 $\pm$ 1.6	14.7 $\pm$ 1.8	3.2 $\pm$ 1.0	6.8 $\pm$ 1.5	40.5 $\pm$ 1.4	16.3 $\pm$ 1.5	5.9 $\pm$ 0.8	8.1 $\pm$ 1.7	36.8 $\pm$ 1.3
PPSL (50:250)	42.7 $\pm$ 1.5	18.1 $\pm$ 1.1	33.8 $\pm$ 1.3	58.1 $\pm$ 1.3	26.9 $\pm$ 1.8	5.7 $\pm$ 1.0	12.1 $\pm$ 1.5	62.1 $\pm$ 1.4	26.7 $\pm$ 1.5	9.5 $\pm$ 0.8	13.8 $\pm$ 1.7	61.3 $\pm$ 1.3
<i>MixSumm</i> (ours)	43.1 $\pm$ 1.1	18.4 $\pm$ 1.5	34.7 $\pm$ 1.0	62.3 $\pm$ 1.4	26.7 $\pm$ 1.7	5.3 $\pm$ 0.9	11.3 $\pm$ 1.4	67.5 $\pm$ 1.3	27.1 $\pm$ 1.4	9.8 $\pm$ 0.7	13.5 $\pm$ 1.6	61.4 $\pm$ 1.2
w/o Mixup	37.5 $\pm$ 1.0	16.0 $\pm$ 1.3	31.2 $\pm$ 0.9	58.2 $\pm$ 1.2	23.2 $\pm$ 1.5	4.6 $\pm$ 0.8	9.8 $\pm$ 1.2	58.7 $\pm$ 1.1	23.6 $\pm$ 1.2	8.5 $\pm$ 0.6	11.7 $\pm$ 1.4	55.8 $\pm$ 1.0
w/o Aug.	23.7 $\pm$ 1.2	10.1 $\pm$ 1.7	18.3 $\pm$ 1.1	34.9 $\pm$ 1.5	14.0 $\pm$ 1.9	2.9 $\pm$ 1.0	6.2 $\pm$ 1.6	36.4 $\pm$ 1.4	14.5 $\pm$ 1.3	5.4 $\pm$ 0.8	7.4 $\pm$ 1.8	19.2 $\pm$ 1.3
LLaMA-3 (0-shot)	37.5 $\pm$ 1.1	13.4 $\pm$ 0.7	21.3 $\pm$ 0.3	42.0 $\pm$ 1.2	11.3 $\pm$ 0.4	2.5 $\pm$ 0.2	7.6 $\pm$ 1.1	34.7 $\pm$ 0.2	20.4 $\pm$ 1.2	2.3 $\pm$ 0.7	9.6 $\pm$ 1.3	26.7 $\pm$ 1.5
LLaMA-3 (1-shot)	37.8 $\pm$ 1.0	13.5 $\pm$ 0.8	21.5 $\pm$ 0.4	41.7 $\pm$ 1.3	11.5 $\pm$ 0.5	2.4 $\pm$ 0.2	7.8 $\pm$ 1.0	34.9 $\pm$ 0.3	20.2 $\pm$ 1.1	2.4 $\pm$ 0.6	9.5 $\pm$ 1.2	26.9 $\pm$ 1.4
LLaMA-3 (5-shot)	44.2 $\pm$ 0.9	19.8 $\pm$ 1.1	36.1 $\pm$ 1.2	64.4 $\pm$ 0.7	26.2 $\pm$ 0.6	5.6 $\pm$ 0.4	12.1 $\pm$ 0.6	69.3 $\pm$ 1.2	-	-	-	-

Table 6.1: **Summarization Results.** Comparison of different text summarization models on TweetSumm, WikiHow, and ArXiv/PubMed datasets. We report ROUGE-1 (R-1), ROUGE-2 (R-2), ROUGE-L (R-L) F_1 scores, and L-Eval scores. We report the mean $\pm$ std. performance across 5 different seeds. Refer to Appendix 6.3.3 and Section 6.3.4 for metric and implementation details. **Note.** TSL results are reported from (Zhuang et al., 2023). For EDA and *MixSumm*, we use a 50-shot $\mathcal{D}_{F,train}$ and generate 1000 examples as $\mathcal{D}_{A,train}$. **Bold** denotes the best-performing model in a given block and **highlight** denotes the *overall* best-performing model. For the ArXiv/PubMed dataset, we could fit only 2 documents into LLaMA-3’s context (1 from $\mathcal{D}_{F,train}$ + 1 generated), so we do not report LLaMA-3 (5-shot).

6.3.2 Evaluation.

We evaluate the summary quality of the models using ROUGE Scores and L-Eval as described in Section 5.2.2 of Chapter 5. Notably, computing *test* L-Eval scores for all the summarization models included in Table 6.1 took $\sim$ 5.6 hrs for TweetSumm, $\sim$ 2.1 days for WikiHow, and $\sim$ 6 days for the ArXiv/PubMed dataset.

T	TweetSumm		WikiHow	
	ROUGE-2	L-Eval	ROUGE-2	L-Eval
5	52.1	67.7	6.1	65.3
10	54.3	69.2	7.2	70.2
15	54.2	69.6	7.6	70.5
20	54.4	69.6	7.7	71.1

Table 6.2: Validation ROUGE-2 and L-Eval scores for different values of T on the TweetSumm datasets.

6.3.3 Implementation Details

Data Augmentation. We set the number of groups $T = 10$ for all datasets⁴, where we find that $T = 10$ provides the best trade-off between the number of clusters and model performance as increasing T further leads to minimal gains or sometimes no gain at all. To get a 50-shot $\mathcal{D}_{F,train}$, we randomly sample 5 examples from each group. Then, we obtain $\mathcal{D}_{A,train}$ by generating 1000 examples using the procedure described in Section 6.2. In the data generation prompt, we include five examples for each group for TweetSumm and WikiHow, but for ArXiv/PubMed, we could only fit two documents at a time in LLaMA-3’s context window after applying the following truncation heuristic to the text. We include l lines before and after each sentence in the ground truth summary so that we are able to fit two examples in the prompt. The average value of l was 5.21 (so approximately ~ 90 for an average summary size of 8 sentences for the ArXiv/PubMed dataset sentences were selected for example⁵). Here, we set the summary size p to 4 sentences for TweetSumm and WikiHow datasets and 8 sentences for the Arxiv/PubMed dataset. We determine these summary sizes based on the average summary size in the few-shot training data $\mathcal{D}_{F,train}$. We host LLaMA-3-70b-Instruct on 4×A100 GPUs with 80G VRAM each and use it as the backbone LLM for all our experiments. Generating $\mathcal{D}_{A,train}$ took ~ 4.2 hrs for TweetSumm, ~ 11.3 hrs for WikiHow, and ~ 1.4 days for ArXiv/PubMed dataset.

Training. For extractive summarization, we train a PreSumm model on the combined *MixSumm*-generated and seed few-shot dataset $\mathcal{D}_{F+A,train}$. We use the TransformerSum repository⁶ to implement our training pipeline. To handle long documents that cannot be fed to the PreSumm at once, we introduce a subroutine that iteratively chunks and

⁴based on validation R-2 and L-Eval scores reported in Table 6.2

⁵ $(5.21 \times 2 \times 8) + 8 = 91.1$

⁶<https://transformersum.readthedocs.io/>

summarizes the document until we obtain a summary of size p . The iterative subroutine is crucial to train PreSumm models on the WikiHow and ArXiv/PubMed datasets with long input documents. For abstractive summarization, we follow an *extractive-then-abstractive* approach, where for a given input document, we first obtain its extractive summary using the full-trained PreSumm model from the previous step. Then, we finetune a DistilBART model that summarizes the PreSumm summaries to generate abstractive summaries.

We initialize the training process with a learning rate of 2×10^{-5} and use a cyclic learning rate scheduler (Smith, 2015). We train all our models for 100 epochs with an early stopping criterion, where we stop the training process if the validation ROUGE-2 score does not improve for more than 10 epochs. We use the AdamW optimizer (Loshchilov and Hutter, 2017) with $\epsilon = 1 \times 10^{-8}$, $\beta_1 = 0.9$, $\beta_2 = 0.99$ and train all our models on one V100 GPU with 12G VRAM. We use `distilbart-12-6-cnn` backbone for abstractive summarization and experiment with two backbones for the PreSumm model: `DistilBERTbase` and `BERTbase` and overall, find `BERTbase` to be better. Training a model on *MixSumm*-generated data took ~ 2.5 hrs for TweetSumm, ~ 13.4 hrs for WikiHow, and ~ 2.7 days for ArXiv/PubMed. Crucially, we repeat each experiment (data augmentation+model training) for 5 random seeds and report the mean and standard deviations for all models unless otherwise stated.

6.3.4 Baselines

We run the following baselines: **1) *MixSumm* (Ours).** We augment $\mathcal{D}_{F,train}$ using the proposed *MixSumm* approach then train a summarization model on $\mathcal{D}_{F+A,train}$. We also run two variants of this baseline to determine the effect of applying data augmentation and mixup, denoted by ***MixSumm* w/o Aug.** and ***MixSumm* w/o Mixup** respectively. **2) Easy Data Augmentation (EDA).** We use an edit-based data augmentation technique (Wei and Zou, 2019) to construct $\mathcal{D}_{A,train}$ instead of using *MixSumm*. Specifically, we apply the EDA technique to each sentence in an article to construct a new example. **3) *MixSumm* (rand.).** Same as **1)** but $\mathcal{D}_{F,train}$ is constructed by randomly selecting k examples from the full training set instead of selecting examples from the T clusters. We also run ***MixSumm* (rand.) w/o Aug.** where we do not perform any data augmentation. **4) Teacher Student Learning (TSL).** A semi-supervised setup proposed by (Zhuang et al., 2023) that employs a teacher-student learning framework similar to ours except they do not use LLM-based pseudo-labeling or relabeling. We report the performance of the TSL (50:500) and TSL (500:500) models ⁷. **5) *PPSL*.** Proposed semi-supervised approach from Chapter 5 using teacher confidence and prompt-based pseudolabel scor-

⁷TSL (m:n) denotes access to m labeled examples and n unlabeled examples

ing for text summarization. We report results for the PPSL (50:250) setting that uses LLaMA-3-70b-Instruct. **6) LLaMA-3-70b (k -shot.)** An in-context learning-based approach where we prompt LLaMA-3-70b-Instruct with k examples randomly selected from $\mathcal{D}_{F,train}$ and then instruct it to summarize a test article. We use the same prompt as the one we use for summarizing articles (Prompt C.3 in Appendix C), except we remove the group information and directly populate it with k examples. **7) Oracle.** A fully supervised model trained on the complete training set $\mathcal{D}_{train}$ to gauge the upper-bound performance for this task.

6.4 Results

6.4.1 *MixSumm* Generates Diverse Documents.

Table 6.3 shows qualitative examples generated by EDA, *MixSumm* w/o mixup and *MixSumm* in Table 6.3. In the table, for each method, the first row shows the generated document, and the second row shows its LLaMA-3-generated summary. For these examples, group 1 contained customer conversations with phone companies, such as Verizon, and group 2 contained customer interactions with airline representatives, such as AirAsia and Delta. **Note.** For brevity, we do not include the input examples used in the prompt, and for EDA, we perform augmentations one sentence in the document at a time. We note that w/o mixup, *MixSumm* generated a decent quality document, but it only covers phone/electronic device-related sentences. *MixSumm*, on the other hand, generated an example that contains mention of flight as well as a device-related issue. Additionally, we note that no matter the quality of the document, LLaMA-3-70b generates a high-quality summary in all cases.

In the context of Table 6.3, we note that w/o mixup, *MixSumm* generates decent quality documents, but it only covers a single topic (phone/electronic device-related sentences.) *MixSumm*, on the other hand, generated an example that contains mention of terms from two topics (flight as well as a device-related issue.) EDA generates the lowest-quality documents with grammatical errors and other artifacts. However, we note that regardless of the quality of the original document, LLaMA-3-70b generates a high-quality summary in all cases.

Comparison w/ Other DA methods. From Table 6.1, we note that *MixSumm* achieves significantly higher L-Eval and ROUGE scores for both extractive and abstractive summarization tasks. This demonstrates the superior generation ability of LLMs compared to

EDA

Generated Document: Customer:@VerizonSupport I penury need help. I do not have your service (6+ iodine hours to setup and cancel an install!) but the premature homeowner did. There is an ONT in the basement that every out a loud, very annoying beep sends 15 minutes? This is ridiculous. How do I iodine stop it? Agent:@633638 Press and hold the alarm silence button/ONT reboot button on the bring up unit for 30 seconds. That bequeath will silence it. Customer:@VerizonSupport done. Will reply back respond later if it beeps again. @VerizonSupport comprise I expected to do this every 24 hours? That's how long the Alarm Silence lasted. Bedden back to beeping every 15 moment since this morning. Agent:@633638 assay unplugging the power and removing the battery. indium After 3mins, put the battery back in and power on. This should assistant with the beeping. Let the states know. DDG Customer:@VerizonSupport I no service with you at all. Can't I just unplug it and remove assault and battery the battery forever? Agent:@633638 Yes, you can do that since you do not inspection and repair have service with us. personify When the battery is dead the beeping will stop. HSB Customer:@VerizonSupport you! Thank Agent:@633638 personify You are welcome!

LLaMA-3 Summary: Customer:@VerizonSupport I penury need help. There is an ONT in the basement that every out a loud, very annoying beep sends 15 minutes? This is ridiculous. How do I iodine stop it? Agent:@633638 Press and hold the alarm silence button/ONT reboot button on the bring up unit for 30 seconds. That bequeath will silence it. Customer:@VerizonSupport done. Will reply back respond later if it beeps again. @VerizonSupport comprise I expected to do this every 24 hours? Agent:@633638 assay unplugging the power and removing the battery.

MixSumm w/o mixup

Generated Document: Customer: My iPhone's battery is draining much faster than usual, and it's overheating. Any suggestions? Here's a link to a screenshot of my battery usage: <https://t.co/xyzABC123> @AppleSupport can you help with this issue? Agent: @UserHelp123 We're here to assist you. First, could you let us know which iOS version your iPhone is currently using? Customer: @AppleSupport It's updated to iOS 14.4. Agent: @UserHelp123 Thanks for the information. Have you installed any new apps or updates around the time you started noticing these issues? Customer: @AppleSupport Yes, I installed a few new games last week. Agent: @UserHelp123 It's possible that those games are causing the battery drain and overheating. Could you try removing them to see if the situation improves? Customer: @AppleSupport I've uninstalled the games, but my phone is still overheating. Agent: @UserHelp123 Understood. Let's perform a soft reset on your iPhone. This won't erase your data but can help with such issues. If the problem persists, we should continue this conversation in DM to get more details and possibly set up a service appointment. Please reach out in DM with the exact time you started experiencing these issues and if you've noticed any specific patterns. <https://t.co/NmDpIWxYfO>

LLaMA-3 Summary: Customer: My iPhone's battery is draining much faster than usual, and it's overheating. Any suggestions? Agent: Have you installed any new apps or updates around the time you started noticing these issues? Customer: @AppleSupport I've uninstalled the games, but my phone is still overheating. Agent: Let's perform a soft reset on your iPhone. This won't erase your data but can help with such issues.

MixSumm

Generated Document: Customer:@VerizonSupport I recently moved into a house that has your equipment but I don't use your service. The device in the basement beeps every 20 minutes. How do I stop it? Agent:@VerizonHelper Please press and hold the alarm silence button on the ONT for about 30 seconds. That should silence it temporarily. Customer:@VerizonSupport I did that, but it started beeping again after a few hours. I need a permanent fix. I have a flight next week, and I can't deal with this noise. Agent:@VerizonHelper If the alarm silence is temporary, try unplugging the device and removing its battery. After a few minutes, reconnect the battery and power it on. Customer:@VerizonSupport I need this sorted out before I leave. Is there no faster way? Agent:@VerizonHelper For a quick solution, you can permanently disconnect the power and remove the battery. This will stop the beeping entirely. Customer:@VerizonSupport Thank you for the advice! I'll handle it immediately.

LLaMA-3 Summary: Customer:@VerizonSupport I recently moved into a house that has your equipment but I don't use your service. The a device in the basement beeps every 20 minutes. How do I stop it? Agent:@VerizonHelper Please press and hold the alarm silence button on the ONT for about 30 seconds. That should silence it temporarily. Customer:@VerizonSupport I did that, but it started beeping again after a few hours. I need a permanent fix. I have a flight next week, and I can't deal with this noise. Agent:@VerizonHelper For a quick solution, you can permanently disconnect the power and remove the battery. This will stop the beeping entirely.

Table 6.3: Full qualitative comparison of EDA, *MixSumm w/o mixup*, and *MixSumm* examples. Refer to Section 6.4.1 for a detailed discussion.

a simple edit-based DA technique like EDA. Next, we compare *MixSumm* with *MixSumm* w/o Mixup, a strong LLM-based data augmentation baseline, and note that removing the mixup component from *MixSumm* significantly lowers ROUGE and L-Eval scores across the board (as verified by a t-test).

6.4.2 Effect of Clustering Documents

We perform a student’s t-test comparing results from *MixSumm* and *MixSumm* (rand.) and note that while ROUGE scores for *MixSumm* are generally higher than *MixSumm* (rand.), the differences are *not* significant. For instance, *MixSumm* and *MixSumm* (rand.) methods achieve L-Eval scores of 65.3 v/s 60.3 on TweetSumm, 81.1 v/s 72.5 on WikiHow and 53.1 v/s 48.4 on ArXiv/PubMed. The only exception was R-2 scores on TweetSumm, where *MixSumm* outperforms *MixSumm* (rand.) by 2.1 points (R-2 of 52.7 v/s 50.6). On the other hand, the difference in L-Eval scores for the two methods was found to be significant by the t-test for all the datasets. This further suggests that ROUGE scores might not be able to capture the semantic correctness of the generated summaries and highlights the importance of an LLM-based evaluator that can discern between nuanced semantics in natural language text. We observe a similar trend after removing the augmentation component from both methods (*MixSumm* w/o Aug. v/s *MixSumm* (rand.) w/o Aug.). Overall, we conclude that *MixSumm* is better than *MixSumm* (rand.), and we should include diverse examples, if possible, in the prompt as it leads to direct improvements in generation quality.

6.4.3 DA v/s SSL Methods

First, removing data augmentation significantly drops ROUGE and L-Eval scores for all datasets (“w/o Aug.” rows v/s EDA v/s *MixSumm*). This suggests that data augmentation is a crucial component in the pipeline. Comparing *MixSumm* with *PPSL* and *TSL* in Table 6.1, we note that our 50-shot *MixSumm* and *MixSumm* (rand.) methods outperform *TSL* (50:500), which uses 50 labeled examples and 500 unlabeled examples. Next, our two methods outperform *TSL* (500:500) on all the metrics except the R-1 score (where the difference was not found to be significant). *MixSumm*, *MixSumm* (rand.), and *TSL* (500:500) achieve a mean R-1 score of 59.1, 58.6, and 59.0, respectively; however, we should also note that the difference in R-1 scores for the two *MixSumm* models on TweetSumm is not significant, as determined through the t-test earlier. Overall, *MixSumm* is better than *TSL* for extractive summarization in extreme data-scarce settings. Next, we note that

MixSumm achieves slightly higher ROUGE scores and significantly higher L-Eval scores than PPSL (50:250) for extractive summarization; however, for abstractive summarization, *MixSumm* and *PPSL* achieve very similar performance for the three datasets. Specifically, *MixSumm* and PPSL (50:250) achieve 59.1 v/s 58.4 R-1 scores, 52.7 v/s 50.1 R-2 scores, 60.5 v/s 59.1 R-L scores, and 65.3 v/s 56.3 L-Eval scores on the TweetSumm dataset (when using BERT). Other datasets follow a similar trend. Overall, we conclude that prompt-based data augmentation might be better than using a semi-supervised method for extractive summarization in data-scarce setups, but both methods are equally performant for abstractive summarization.

6.4.4 Knowledge Distillation from LLaMA-3

We include the 0-shot, 1-shot, and 5-shot LLaMA-3 results in Table 6.1. First, we note that increasing the number of examples for the LLaMA-3 method leads to expected improvements in performance except L-Eval scores on the ArXiv/PubMed dataset, where 0-shot and 1-shot LLaMA-3 models achieve similar L-Eval scores. This may suggest that LLaMA-3 struggles with understanding very long documents. Next, we note that 0-shot LLaMA-3 outperforms 50-shot *MixSumm* w/o Aug baseline on the TweetSumm dataset in terms of ROUGE scores and L-Eval scores, and it achieves competitive results on ArXiv/PubMed. Lastly, we note that *MixSumm* achieves competitive performance against LLaMA-3 as a summarizer for both extractive and abstractive tasks, whereas *PPSL* is competitive with LLaMA-3 on only the abstractive task. Additionally, we note that our methods achieve comparable ROUGE scores to the Oracle model despite using just 50 labels compared to 1000 examples used by the oracle (95% less). We perform a T-Test to determine the significance of the performance difference and find that the difference in the R-1, R-2, and L-Eval scores is significant for TweetSumm; differences in the R-1, R-L, and L-Eval scores are significant for WikiHow; and differences in the R-2, R-L, and L-Eval scores are significant for ArXiv/PubMed. Overall, we conclude that both *MixSumm* and *PPSL* are highly performant models compared to LLaMA-3-70b model, demonstrating effective distillation effect from LLaMA-3-70b to BERT- and DistilBART-based models.

6.5 Limitations

We use LLaMA-3-70b-Instruct for our experiments, which has a context window size of 8192 tokens, so it is not possible to fit many long documents in the model’s context (like articles in the ArXiv/PubMed dataset). We can explore the use of position interpolation

(PI) to increase the context window length of LLaMA (Chen et al., 2023a) or switch to more recent LLaMA-3.1 family of models.

Currently, we only consider text summarization for the English language. Moving forward, we can expand our method to multiple languages. More research on efficiently handling long documents during the training process is also needed, as we currently rely on a chunk-and-summarize subroutine to train our models for long documents, which results in significant delays in document processing. We can consider using alternative transformer architectures such as LongFormer (Beltagy et al., 2020) as PreSumm’s backend. Our experiments on the ArXiv/PubMed dataset also suggest that even with LLMs that can process really long documents, we might struggle with truly understanding the context to generate high-quality summaries.

We generate large textual datasets using LLMs, and even though we use an instruction-tuned model, we need to be careful about any bias it might exhibit or any potentially harmful content that it might generate. Language model debiasing is a common potential solution to address this issue (Meade et al., 2021; Guo et al., 2022). Additionally, we suggest involving a human moderator if these systems are to be made public-facing.

6.6 Conclusion

In this chapter, we focus on low-resource text summarization and show that *MixSumm*, a two-step data augmentation method, is better than existing approaches for low-resource summarization. It further demonstrates that generating novel documents by mixing topical information from different documents leads to a diverse augmented set, and such data generation schema induces knowledge transfer from a large teacher model **LLaMA-3-70b-Instruct** into much smaller BERT- and DistilBART-based models. LLM-based approaches are underexplored for low-resource text summarization, and through this work, we hope to spark an interest in the research community to address various challenges of this task.

6.7 Key Takeaways

- LLM-based data augmentation leads to highly performant models for low-resource text summarization, achieving better results than traditional data augmentation strategies and semi-supervised methods.

- Between *MixSumm* and *PPSL*, data augmentation has a slight advantage over the pseudolabeling strategy for extractive summarization, while for abstractive summarization, both models achieve competitive results with LLaMA-3-70b.
- The Mixup style data augmentation is also effective for text summarization, and removing it significantly reduces the performance of the model.

The learnings from this chapter with the insights from Chapter 5 can be combined to establish that LLMs can be effective data generators *and* data labelers for a generative NLP task like text summarization. In the next chapter, we study how to utilize LLMs for a subjective task where the objective is not as clear-cut as summarization or classification.

Chapter 7

Computational Modeling of Artistic Inspiration

The creative process, along with the inspiration that drives the creation of art—whether visual, poetic, or musical—remains one of the least understood aspects of human experience. Every artist employs different methods to summon inspiration or, as the XIX century American artist Robert Henri noted, to “be in that wonderful state which makes art inevitable.”

[Rubin \(2023\)](#) describes three distinct stages to a creative process: the Seed phase, where one is entirely open to any and all forms of inspiration; the Experimentation phase, where one tries multiple approaches to allow the nascent ideas or “seeds” to grow and take shape; and the Crafting phase, where the artist refines these ideas into a cohesive work. This work focuses on the Seed phase, exploring how AI-generated poetic lines can inspire new creative pathways for artists. Much like organically encountered stimuli, these AI-generated lines serve not as direct inputs into the final artwork but as triggers that guide the artist into a creative mindset.

Assessing creativity in AI-generated content is challenging due to its subjective and individual nature. Traditional evaluation methods, such as the Consensual Assessment Technique (CAT) ([Amabile, 1982](#)) and the Torrance Test of Creative Thinking (TTCT) ([Torrance, 1966](#)), provide structured evaluation frameworks, but they may not be well-suited for evaluating AI-generated outputs intended to inspire artists rather than serve as final artifacts. For instance, CAT uses the collective judgment of experts to assess creativity, but it does not assess the inspirational potential or the ability of the outputs to spark further creative development in individual artists. Similarly, while TTCT measures

divergent thinking, it has faced criticism for not reflecting the diverse forms that creativity can take (Baer, 2011). Overall, in the context of creating art, it is crucial to measure how well AI systems support individual creative processes that encompass a wide spectrum of artistic preferences and skill levels.

In this chapter, we will study a novel evaluation framework that combines the usage of LLMs and statistical methods to identify and formalize the key linguistic and poetic properties of AI-generated lines that best explain the subjective preferences of artists during the *Seed* phase of the creative process. Our framework does not assume a pre-requisite skill level of an artist and can be applied to study the artistic preferences of a wide spectrum of creative individuals for any task that is inherently subjective. Further, we focus on analyzing the inspirational quality of lyric lines in the Seed phase rather than evaluating them as the final artifact, thereby addressing the key limitation of previous creativity evaluation frameworks that were too constrained in their definition of what they considered creative or inspiring.

We test our framework on a collection of 3025 poetic lines generated by LyricJam (Vechtomova et al., 2021), a publicly available research system that creates poetic lines in response to instrumental music, with the goal of assisting creative individuals in finding novel lyrical and poetic ideas for use in the Seed phase of the creative process¹. We refer to the dataset of poetic lines as *EvocativeLines* and conduct comprehensive experiments to establish the reliability of our framework in accurately forecasting the artistic preferences of an individual. Lastly, we provide insights about the suitability of recent large language models (LLMs) like LLaMA-3-70b (Dubey et al., 2024) for our task of modeling diverse artistic preferences, which involves a high degree of subjectivity.

7.1 Methodology

The relationship between aesthetic pleasure and complexity, known as the inverted U-shape or “Wundt” effect, suggests that stimuli of intermediate complexity are most pleasing. This U-shape effect, first explored by Berlyne (Berlyne, 1975), has been observed in various domains such as music (Gold et al., 2019) and product design (Althuizen, 2021). Gold et al. (2019) offered a possible explanation for the inverted U-shaped effect grounded in learning. Learning engages the dopaminergic reward system in the brain, and when the events (in our case, the sequence of words in a sentence) are completely predictable, little or no learning is taking place. On the other end of the complexity scale, when the events

¹Link to the app: <https://lyricjam.ai/>

are fully random, no patterns can be discerned hence no learning can take place either. However, events with intermediate degrees of predictability enhance learning and make us engage with the stimulus more meaningfully. Complexity in works of art, such as music and poetry, is often also associated with unpredictability and surprise. Many artists and poets since the early 20th century have recognized that established language forms can become worn out and lose their evocative power. William Burroughs argued that language acts as a "lock" that restricts creativity and confines us to predictable patterns (Burroughs, 2012). Burroughs popularized the cut-up technique, which involves creating poetic texts by randomly re-arranging words. This method was later adopted by music artists like David Bowie (Jones, 2017) and Kurt Cobain (Cross, 2001), who used it to find inspiration and generate new creative ideas for their lyrics. Poets and writers have also sought other ways to make their writing less predictable. Tom Spanbauer, for instance, coined the term "burnt tongue" to describe a literary device where language is deliberately misused to slow readers down and force them to engage more deeply with the text (Palahniuk, 2020). This intuition is supported by eye-tracking studies, which have shown that predictable words are often skipped or elicit shorter fixation times (Lowder et al., 2018). While there is evidence that the U-shape effect exists across different aesthetic domains, there is also strong evidence that our preferences for aesthetic stimuli are highly individual (Gold et al., 2019; Althuizen, 2021). Inspired by the different theories and evidence, we propose a two-step framework that **a)** measures multiple poetic and linguistic characteristics of a lyric line; and **b)** uses those measurements to learn the artistic preferences of an individual.

7.2 Step 1: Identifying The Key Poetic and Stylistic Characteristics

Poetic/Stylistic Characteristics. There are over 100 stylistic devices in the English language that a poet or lyricist can use to achieve a certain emotional effect. The most prominent stylistic devices in poetry include: **sound devices** such as alliteration and assonance focusing on the auditory properties of the poem; **figurative and descriptive language** like metaphor, and poetic imagery that encompasses the usage of language in non-literal ways to convey a deeper meaning or create vivid imagery; and **structural devices** like anaphora, and epistrophe that relate to the arrangement of words and lines. While many of these devices act together to create the desired effect, a poem is foremost intended to evoke emotion in the reader and heighten perceptual awareness (Dunnigan, 2014). Figurative and descriptive language plays an important role in achieving this purpose. For this reason, we design the following four features to capture and reflect these

critical aspects of poetry.

7.2.1 Poetic Imagery

Poetic Imagery is a stylistic device that involves references to physical objects, appealing to the reader’s senses (e.g., visual, auditory, tactile) to evoke specific emotions or feelings or make abstract ideas more tangible and relatable (Pound, 1913; Brooks, 1976; Kao and Jurafsky, 2012).

To compute poetic imagery for a lyric line, we prompt a LLaMA-3-70b model to output a rating between 1-5. Since poetic imagery is a well-established stylistic device in English, we perform prompting in a zero-shot fashion. This further avoids the induction of preference bias in the LLM outputs. A higher rating denotes a higher degree of poetic imagery. Formally, given the set of ratings $\{r_1, r_2, r_3, r_4, r_5\}$, we compute the LLM score for a sentence S as follows:

$$score_{img}(S) = \sum_{i=1}^5 p(r_i) \times r_i, \quad (7.1)$$

where $p(r_i)$ is the probability of the LLM assigning an imagery rating of r_i to the sentence S (see Figure C.10 in Appendix C for the full prompt). We use a weighted summation instead of directly using the output tokens as they provide more fine-grained, continuous scores that better capture the nuances of texts (Liu et al., 2023b). To construct the final imagery feature vector, we combine the scores of all the subsequences in a given lyric line to also capture the temporal fluctuations and development in the sentence:

$$\mathbf{S}_{\text{imagery}} = \langle score_{img}(\mathcal{S}_t) \rangle_{t=1}^T, \quad (7.2)$$

where $\mathcal{S}_t = (w_1, w_2, \dots, w_t)$ denotes a subsequence with the first t words in S . We find this method to appropriately capture the poetic imagery of a lyric line. For instance, “there’s a little red to the sea” creates a more evocative image in the reader’s mind and has a higher imagery score of 4.01 compared to “this is all,” which only achieves a score of 1.27.

7.2.2 Word Energy

Word Energy is a composite feature that encompasses symbolism and diction, both of which contribute to the emotional and conceptual impact of a poem. Symbolism uses

references to objects that evoke in the reader’s mind certain abstract concepts, associations, and emotions (Cassirer, 1946). For example, the word “chains” is used to represent a feeling of entrapment and lack of control. Diction refers to the choice of words made by the poet to achieve a desired effect (Brooks, 1976). For example, by choosing the word “beseeching” over a more neutral word “searching,” the poet emphasizes the sense of longing and desperation. Similar to imagery, we prompt a LLaMA-3-70b model to output a rating between 1-5 to measure the notion of word energy in a given lyric line. Note that the prompting is zero-shot, and since word energy is a composite feature, we include three example terms (not complete phrases) that reflect the notion of word energy. A higher rating denotes higher word energy. The final feature vector also assumes a similar form to Equation 7.2 for a lyric line S :

$$\mathbf{S}_{\text{energy}} = \langle \text{score}_{\text{eng}}(\mathcal{S}_t) \rangle_{t=1}^T, \quad (7.3)$$

where $\mathcal{S}_t = (w_1, w_2, \dots, w_t)$ denotes a subsequence with the first t words in S , and $\text{score}_{\text{eng}}(\mathcal{S}_t)$ is computed similar to $\text{score}_{\text{img}}(\mathcal{S}_t)$ in Equation 7.1 except the ratings reflect the notion of word energy instead of poetic imagery. For example, “all the words will drown” achieves a score of 4.11 compared to “where i know about you,” which gets a score of 1.92.

7.2.3 Level of Abstraction

We measure the level of abstraction or ambiguity in a line, as a higher level of abstraction indicates that the line is more likely to be unresolved and allows for multiple interpretations; hence, it can be potentially more engaging. The *Linguistic Category Model (LCM)* is a widely employed theoretical framework that considers the social-cognitive functions of four linguistic categories and can measure the degree of abstraction in a sentence (Semin and Fiedler, 1988; Johnson-Grey et al., 2020). As a feature for quantifying the degree of abstraction of the language used in a poetic line, this measure captures key stylistic devices by assessing the specific types of words chosen: Descriptive Action Verbs (DAVs), Interpretative Action Verbs (IAVs), State Verbs (SVs), and Adjectives (ADJs). LCM is intended to represent the balance between abstract and concrete language, which is important in defining the poem’s tone, mood, and the vividness of its imagery. Furthermore, the LCM indirectly indicates the presence of figurative language, such as metaphors and similes, by highlighting the use of more abstract expressions. We compute the feature vector representing the level of abstraction of a sentence S as follows:

$$\mathbf{S}_{\text{abs}} = \langle \text{score}_{\text{abs}}(\mathcal{S}_t) \rangle_{t=1}^T \quad (7.4)$$

where $\mathcal{S}_t = (w_1, w_2, \dots, w_t)$ denotes a subsequence with the first t words in S , and $score_{abs}(\mathcal{S}_t)$ is computed as follows:

$$score_{abs} = \frac{D + (2 \times I) + (3 \times S) + (4 \times A)}{D + I + S + A} \quad (7.5)$$

In Equation 7.5, D , I , S and A denote the number of times DAVs like “eating” and “walking,” IAVs like “helping” and “playing,” SVs like “love” and “admire,” and ADJs like “ethical” and “uneven” occur in the text. Weights assigned to the four counts are based on their theorized abstraction level determined through a linguistic study of 40,000 English words (Brysbaert et al., 2014). Here, definitive, concrete verbs (DAVs) receive the lowest weight of 1.0, whereas abstract adjectives get the highest weight of 4.0.

We prompt a LLaMA-3-70b model to obtain the counts of the four linguistic categories in a lyric line and then we compute the final feature vector.

We find our scores highly reliable for this task as they are able to distinguish between the phrase “rearranging this stage,” which is very definitive and has a lower level of abstraction compared to “the greatest began to deny,” which remains somewhat unresolved. Our experiments (discussed in Section 7.5) also indicate that the level of abstraction in a line is one of the most prominent features in identifying potentially evocative lines for an artist.

7.2.4 Valence

Valence reflects the emotions expressed in a lyric line and is crucial for assessing an artist’s preferred emotional tone that may guide their own creative process (Scherer, 1984; Charland, 2005). For instance, an artist might favor lines that creatively blend positive and negative connotations, such as “the greatest began to deny,” or they may prefer lines that convey a purely positive sentiment, like “tranquility in the lovers.”

To measure the valence of a line, we build a computation model of emotion that assigns a valence score based on the detected emotional tone of a line. **First**, we define a list of 29 fine-grained emotion categories—including admiration, nostalgia, gratitude, grief, and remorse. The list of emotions is derived from the GoEmotions dataset (Demszky et al., 2020), with the addition of one extra category, “nostalgia,” to capture the unique emotional nuances often present in lyrical content. **Next**, we employ a prompting-based classifier built on the LLaMA-3-70b language model to obtain the probabilities for the top 5 emotional categories for a given lyric line. Let S represent the input sentence, and E_k denote the set of the top- k emotions predicted by the classifier (with $k = 5$ in our case). The probability vector for S is then constructed as follows:

$$\mathbf{p}(S) = \mathcal{T}(f(S)) = \langle \tilde{p}_1, \tilde{p}_2, \dots, \tilde{p}_m \rangle, \quad (7.6)$$

where m is the total number of emotion categories ($m = 29$ in our case), $f(\cdot)$ denotes the probabilities of the top 5 emotion categories predicted by the LLaMA-3.1 classifier, and $\mathcal{T}$ represents the transformation defined as:

$$\tilde{p}_e = \begin{cases} p_e, & \text{if } e \in E_k \\ p_{rem}, & \text{if } e \notin E_k \end{cases} \quad (7.7)$$

where p_e is the probability of emotion e predicted by the LLaMA classifier and $p_{rem} = \frac{1 - \sum_{e \in E_k} p_e}{m - k}$ represents the remaining probability mass, equally distributed among the categories not included in the top 5. In essence, this process preserves the probabilities of the top 5 emotions as predicted by LLaMA while redistributing the remaining probability mass evenly across the other emotion categories. **Finally**, we construct the complete valence vector for a lyric line $S = (w_1, w_2, \dots, w_T)$ as follows:

$$\mathbf{S}_{\mathbf{vfull}} = \langle \mathbf{p}(\mathcal{S}_t) \rangle_{t=1}^T, \quad (7.8)$$

where $\mathcal{S}_t = (w_1, w_2, \dots, w_t)$ denotes a subsequence with the first t words in S . We use subsequences to capture the temporal development of valence.

We also construct a uni-dimensional variant of $\mathbf{S}_{\mathbf{vfull}}$, where we first group the 29 fine-grained emotions into three sets E_+, E_0, E_- denoting sets of positive, neutral, and negative emotions, respectively. Then, we collapse the full emotion probability vector in Equation 7.6 to an overall score $\tilde{p} \in [-1, 1]$ denoting the probability of a sentence conveying a positive emotion:

$$\mathbf{p}_{\mathbf{bin}}(S) = \tilde{p}_+ = \sum \mathbb{I}_e \cdot \tilde{p}_e, \quad (7.9)$$

where $\mathbb{I}_e$ is an indicator function defined as:

$$\mathbb{I}_e = \begin{cases} +1, & \text{if } e \in E_+ \\ 0, & \text{if } e \in E_0 \\ -1, & \text{if } e \in E_- \end{cases} \quad (7.10)$$

so that the final feature vector becomes:

$$\mathbf{S}_{\mathbf{vbin}} = \langle \mathbf{p}_{\mathbf{bin}}(\mathcal{S}_t) \rangle_{t=1}^T, \quad (7.11)$$

Notably, we also tried a RoBERTa classifier (Liu, 2019) fine-tuned on the GoEmotions dataset as the classifier $f(\cdot)$, but our preliminary results suggested that features from the LLaMA-based prompting-based classifier have higher predictive power.

Linguistic and Statistical Characteristics. In addition to quantifying the stylistic characteristics and emotional aspects, we measure the inherent complexity of a poetic line. Complexity is typically expressed in features such as predictability and surprise, and we propose the following features to measure them:

7.2.5 Surprisal

Surprisal (Shannon, 1948) of a word indicates the predictability of that word in a sentence. It is measured as the negative log probability of the word given its context:

$$s(w_t) = -\log_2 P(w_t|\mathbf{w}_{<t}), \quad (7.12)$$

where, $s(w_t)$ denotes the surprisal value for the t -th word of the sentence. To compute the surprisal of the complete sentence, we compose a vector of word-level surprisals, appended with the average surprisal value of all the words in the sentence as follows:

$$\mathbf{S}_{\text{surprisal}} = \langle s(w_t) \rangle_{t=1}^T \oplus \langle \bar{s} \rangle, \quad (7.13)$$

where T denotes the number of words in the sentence, $\bar{s} = \sum s(w_t)/T$ denotes the mean of surprisal values for all the words in the sentence, and $\oplus$ denotes the concatenation operation. Since we do not have access to the true probability distribution P , we use GPT-2 (Radford et al., 2019)², a powerful autoregressive language model pre-trained on a large text corpus, to obtain a good approximation of the desired log probabilities $P(w_t|\mathbf{w}_{<t})$. The described surprisal feature can distinguish between a sentence like “a knowledge of the world,” which has a lower surprisal ($s_{avg} = 10.12$), indicating higher predictability, and “a just dream with the fire,” which has a higher surprisal ($s_{avg} = 11.21$), indicating a higher potential of novelty in the sentence (note that these values are on a log scale, so the difference is exponential.)

7.2.6 Contextual Entropy

Contextual Entropy (Shannon, 1948) of a word is defined as the expected value of its *Surprisal*:

$$h(w_t) = \sum_{w_i \in V} p(w_i|\mathbf{w}_{<t}) \log_2 P(w_i|\mathbf{w}_{<t}), \quad (7.14)$$

²specifically, we use the 774M parameter gpt2-large model from Huggingface: <https://huggingface.co/openai-community/gpt2-large>

where $h(w_t)$ denotes the contextual entropy value for the t -th word of the sentence, and V denotes the vocabulary (all possible words that can appear next). Similar to surprisal, we use GPT-2 to compute the probabilities and construct the final sentence-level entropy feature vector by combining the word-level entropy values and their mean as follows:

$$\mathbf{S}_{\text{entropy}} = \langle h(w_t) \rangle_{t=1}^T \oplus \langle \bar{h} \rangle, \quad (7.15)$$

where T denotes the number of words in the sentence, and $\bar{h} = \sum h(w_t)/T$ denotes the mean of entropy values for all the words in the sentence. Like surprisal, contextual entropy can also help us gauge the unpredictability/novelty of a sentence. For instance, “i remember you” has a lower entropy ($h_{\text{avg}} = 6.21$) compared to “pure crime of the lost” ($h_{\text{avg}} = 9.53$). Additionally, contextual entropy also exhibits a strong positive correlation with the reading times of a word (Lowder et al., 2018), indicating its potential to capture the complexity of a sentence as well (higher entropy $\rightarrow$ higher complexity).

7.2.7 Normalized Pointwise Mutual Information (NPMI)

NPMI measures the association of two words by comparing the probability of their co-occurrence with the probabilities of their individual occurrences in a corpus (Bouma, 2009). Formally, the NPMI of a word x occurring next to a word y is defined as follows:

$$N(x, y) = \left(\ln \frac{p(x, y)}{p(x)p(y)} \right) / -\ln p(x, y) \quad (7.16)$$

In Equation 7.16, $p(x) = \text{count}(x)/N$, where $\text{count}(x)$ denotes the number of occurrences of the word x in the corpus and N is the total number of word occurrences in the corpus. Similar definitions hold for $p(y)$ and $p(x, y)$. NPMI value is bounded between $[-1, 1]$, where a high NPMI value denotes high predictability and vice versa. As an example, the pair (“see”, “you”) has a high NPMI value in our corpus (0.47) compared to (“never”, “the”), which has a low NPMI value (-0.11) as “see” and “you” often occur together, whereas “never” is more frequently followed by verbs like “hurt,” “end,” and “leave” compared to a determiner like “the.”

In our work, we compute two variants of NPMI values: unidirectional and bidirectional. When computing unidirectional $N(x, y)$, we only consider the co-occurrence of (x, y) if y appears *immediately after* x , but for bidirectional $N(x, y)$, we also consider the co-occurrence if y appears *immediately before* x . We denote the unidirectional and bidirectional NPMI values as $N_1(x, y)$ and $N_2(x, y)$, respectively. We construct the final uni-

and bi-directional NPMI vectors in a similar fashion to surprisal and contextual entropy:

$$\mathbf{S}_{\text{npmi,uni}} = \langle N_{\text{uni}}(w_t, w_{t+1}) \rangle_{t=1}^{T-1} \oplus \langle \overline{N}_{\text{uni}} \rangle \quad (7.17)$$

$$\mathbf{S}_{\text{npmi,bi}} = \langle N_{\text{bi}}(w_t, w_{t+1}) \rangle_{t=1}^{T-1} \oplus \langle \overline{N}_{\text{bi}} \rangle \quad (7.18)$$

where T denotes the number of words in the sentence, and $\overline{N}_{\text{uni}} = \sum N_{\text{uni}}(w_t, w_{t+1})/T$ denotes the mean of unidirectional NPMI values for all the word pairs in the sentence and $\overline{N}_{\text{bi}} = \sum N_{\text{bi}}(w_t, w_{t-1})/T$ denotes the mean of bidirectional NPMI values.

7.2.8 Banality

Banality refers to the use of common or clichéd phrases, themes, or expressions that lack originality or depth. It can also reflect the cultural and temporal context of the lyrics. For instance, an artist may use clichéd expressions to resonate with a broad audience or fit a certain era. Therefore, banality is a crucial feature to measure when studying the creative preferences of different artists. Similar to poetic imagery and word energy, we prompt a LLaMA-3-70b model to provide a rating from 1-5 that indicates how banal or clichéd a poetic expression is. Again, we prompt in a zero-shot fashion, and a high rating indicates a banal expression, whereas a low rating indicates a unique or original expression. We construct the final feature vector denoting banality as follows:

$$\mathbf{S}_{\text{banality}} = \langle \text{score}_{\text{bnl}}(\mathcal{S}_t) \rangle_{t=1}^T, \quad (7.19)$$

where $\mathcal{S}_t = (w_1, w_2, \dots, w_t)$ denotes a subsequence with the first t words in S , and $\text{score}_{\text{bnl}}(\mathcal{S}_t)$ is computed similar to $\text{score}_{\text{img}}(\mathcal{S}_t)$ except the ratings now reflect the level of banality instead of poetic imagery or word energy. For instance, the expression “take the test” is a banal expression and achieves a score of 3.18 compared to “breathe back the world,” which achieves a score of 0.98. Table 7.1 summarizes the different features proposed in this section along with their notations.

7.3 Step 2: Training the Calibration Network

Artistic preferences can vary from one individual to another – a lyric line with the same stylistic and linguistic properties might be inspiring to one person but not another. Therefore, we must calibrate the measurements from the first step to an individual’s artistic preference before we can predict the artistic preference of an individual for a given lyric line.

Feature Name	Notation
Poetic Imagery	$\mathbf{S}_{\text{imagery}} \in \mathbb{R}^{T \times T}$
Word Energy	$\mathbf{S}_{\text{energy}} \in \mathbb{R}^{T \times T}$
Level of Abstraction	$\mathbf{S}_{\text{abs}} \in \mathbb{R}^{T \times T}$
Valence (full)	$\mathbf{S}_{\text{vfull}} \in \mathbb{R}^{T \times m}$
Valence (binary)	$\mathbf{S}_{\text{vbin}} \in \mathbb{R}^T$
Surprisal	$\mathbf{S}_{\text{surprisal}} \in \mathbb{R}^{T \times T}$
Contextual Entropy	$\mathbf{S}_{\text{entropy}} \in \mathbb{R}^{T \times T}$
NPMI (uni-dir.)	$\mathbf{S}_{\text{npmi,uni}} \in \mathbb{R}^{T \times T}$
NPMI (bi-dir.)	$\mathbf{S}_{\text{npmi,bi}} \in \mathbb{R}^{T \times T}$
Banality	$\mathbf{S}_{\text{banality}} \in \mathbb{R}^{T \times T}$

Table 7.1: Summary of features and their dimensions. Here, T is the number of words (or time steps) in the input sequence, and m is the total number of emotions (29, in our case).

We train the calibration network on a classification objective: given the linguistic and poetic features of a lyric line, predict if the line is inspiring to the end user. Specifically, consider a dataset of artistic preferences of a creative individual, $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, where $x_i \in \mathcal{X}$ denotes features of the i -th lyric line in the dataset and $y_i \in \mathcal{Y}$ denotes the corresponding label of whether or not the artist likes the lyric line. Then, the goal of the calibration network is to learn a function $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ that maps each feature vector x_i to a probability distribution over classes $\hat{y} = f_\theta(x_i) = P(y|x_i; \theta)$. For our binary classification setup, $\mathcal{Y} = \{\text{inspiring}, \text{not inspiring}\}$. Notably, we do not directly map the embeddings of lyric lines to the predictions during calibration, as sentence embeddings do not necessarily capture the creative attributes and focus more on the general semantic and syntactic properties of the text.

7.4 Experimental Setup

7.4.1 Dataset

We collect 3,025 lyric lines from the LyricJam platform, a system that autonomously generates novel lyric lines using bimodal neural networks (Vechtomova et al., 2021; Vechtomova and Sahu, 2023)³. We do not perform any pre-processing on the lyric lines, but notably, all

³<https://lyricjam.ai>



Figure 7.1: Inter-annotator agreement on the **EvocativeLines (small)** dataset.

Annotator	#pos	#neg
EvocativeLines		
$\mathcal{A}_1$	1466	1559
$\mathcal{A}_2$	1422	1603
EvocativeLines (small)		
$\mathcal{A}_1$	100	100
$\mathcal{A}_2$	59	141
$\mathcal{A}_3$	117	83
$\mathcal{A}_4$	121	79
$\mathcal{A}_5$	116	84
$\mathcal{A}_6$	71	129
$\mathcal{A}_7$	112	88
$\mathcal{A}_8$	77	123
$\mathcal{A}_9$	140	60
$\mathcal{A}_{10}$	117	83

Table 7.2: Dataset statistics, broken down by annotator.

of them are lowercase. Next, two participants labeled each line as either “inspiring” or “not inspiring.” By “inspiring,” we refer to whether these lines created an emotional impact and stimulated created ideas. Both participants actively engage in the process of creating art—including writing poems and short stories—which makes them suitable for the labeling task. We refer to our final dataset of 3,025 annotated lyric lines as the ‘Evocative Lyric Lines Dataset’ (or *EvocativeLines* for short.) The labeling process took approximately 12 hours, spread over multiple weeks.

To test the robustness of our framework on more preference profiles, we reached out to eight human labelers, each actively engaging in at least one creative activity like writing poetry or playing a musical instrument. Specifically, we asked each annotator to rate a lyric line on a scale of 1-10 based on how inspiring they found the line. Figure C.7 shows the complete guidelines presented to the annotators, where we also clarify what we mean by ‘inspiration.’ We perform mix-max scaling to normalize the annotator ratings and obtain “inspiring” and “not inspiring” ratings. Notably, we show the eight human annotators a subset of 200 lyric lines from the original dataset to keep the labeling process fast and efficient. We obtain the 200 lines by randomly selecting 100 “inspiring” and 100 “not inspiring” lines of varying lengths (short, mid, and long) from the original dataset. Choosing a smaller subset ensured that the estimated labeling time remained manageable, ranging from 30 to 45 minutes, thus also minimizing decision fatigue among the annotators. This strategy allowed us to obtain highly reliable annotations while covering a broad spectrum of creative preferences to test the framework.

We denote our set of annotators as $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_{10}\}$, where $\mathcal{A}_1$ and $\mathcal{A}_2$ are the two participants who labeled the complete set of lines and $\mathcal{A}_{3-10}$ are the remaining human annotators. Furthermore, we refer to the underlying preferences captured by an annotator’s labels as a *preference profile*. Figure 7.1 shows the inter-annotator agreement of the annotators on the 200 lines (henceforth, referred to as *EvocativeLines (small)* dataset). The agreement score between two users denotes the percentage of lines they both found inspiring/not inspiring. We note that the agreement values show high fluctuations, ranging between 47.5% – 72.5%, with a mean (std) agreement of $65.3 \pm 3.1\%$. In other words, the annotators disagree on $\sim 70/200$ lines on average. Such disagreement is expected in the perception of aesthetic stimuli, as shown by Gold et al. (2019). This showcases that our annotations capture diverse preference profiles of what lines different individuals consider inspiring. Table 7.2 summarizes dataset statistics.

7.4.2 Experiments

Step 1: Computing Features. To test our framework on the *EvocativeLines* dataset, we first compute all the features described in Section 7.2. Specifically, we combine all the features *along the time dimension* to obtain the following overall feature vector for a given lyric line:

$$\mathbf{S}_{\text{all}} = [\mathbf{S}_{\text{imagery}} | \dots | \mathbf{S}_{\text{banality}}] \quad (7.20)$$

where $|$ denotes the concatenation operation along the time dimension, T is the number of time steps (= number of words in the sentence), and $\mathbf{S}_{\text{all}} \in \mathbb{R}^{T \times n}$ with $n = (8T + m)$ when using $\mathbf{S}_{\text{vfull}}$ and $n = (8T + 1)$ when using $\mathbf{S}_{\text{vbin}}$.

Step 2: Training the Calibration Network We create an 80-20 split of training-testing in the dataset and use the features in the training set to train the calibration network. We use stratified sampling when creating the splits to ensure that the distribution of labels is preserved across the two dataset parts. We perform 5-fold cross-validation for tuning the hyperparameters of the calibration network. When training the calibration network on the *EvocativeLines (small)* dataset, we create a 75-25 split of training-testing and then do a 5-fold cross-validation.

Choice of calibration network. We explore multiple deep-learning (DL) and machine learning (ML) based architectures for the calibration network. Namely, we try the following networks:

1. **LSTM+Attn ($\mathbf{S}_{\text{all}}$)**: a bi-directional classification network with Long Short Term Memory (LSTM) cells (Hochreiter and Schmidhuber, 1997) coupled with an attention mechanism (Bahdanau et al., 2014). We use an LSTM network as all of our features are temporal, and LSTMs are adept at modeling long-range temporal dependencies. We also perform ablation studies where we train the classifier on a subset of $\mathbf{S}_{\text{all}}$ to test the effect of the individual features;
2. **XGBoost (Chen and Guestrin, 2016)**: We train a standard XGBoost model on our features; however, we *flatten* $\mathbf{S}_{\text{all}}$ as XGBoost does not inherently handle sequential data with a timestep dimension. Specifically, we train XGBoost (and all the other ML-based classifiers) on the following flattened input feature:

$$\mathbf{S}_{\text{flat}} = [\dot{\mathbf{S}}_{\text{imagery}} \oplus \cdots \oplus \dot{\mathbf{S}}_{\text{banality}}], \quad (7.21)$$

where $\dot{\mathbf{S}}_{(\cdot)}$ is the flattened version of $\mathbf{S}_{(\cdot)}$ and $\oplus$ is the concatenation operation. $\mathbf{S}_{\text{flat}} \in \mathbb{R}^{Tn}$, where $n = 8T + m$;

3. **Other ML-based Classifiers.** In addition to XGBoost, we train multiple other ML-based classifiers on $\mathbf{S}_{\text{flat}}$. Specifically, we train Logistic Regression (**LR**), Support Vector Machine (**SVM**), Decision Trees (**DT**), and Random Forest (**RF**) models (Nelder and Wedderburn, 1972; Cortes and Vapnik, 1995; Quinlan, 1986; Breiman, 2001).

Other Frameworks. We compare our framework with multiple strong baselines. Specifically, we compare the following:

1. **LSTM+Attn (SBERT)**: we train an LSTM+Attn classifier that maps the Sentence-BERT (Reimers and Gurevych, 2019) embeddings of lyric lines to the final rating. SBERT is a popular text embedding model that modifies the BERT architecture with siamese (Bromley et al., 1993) and triplet networks (Schroff et al., 2015) to obtain semantically meaningful sentence representations. Note that in this scenario, there will be only one “time step;”
2. **BERT (Devlin et al., 2019a)**: a state-of-the-art transformer-based language model pre-trained on large textual corpora. It is widely adopted for various natural language processing tasks, such as classification, question answering, and named entity recognition. We fine-tune a Bert-base model, which has 110M parameters, on the *EvocativeLines* dataset, which serves as a strong baseline comparing the performance of mapping the sentence embeddings of lyric lines directly with the final prediction;

3. **LLaMA-3:** a prompting-based k -shot LLaMA classifier, where, for a given test line, we prompt a LLaMA-3-70b model with k “inspiring” and k “not inspiring” examples from the dataset and ask it to classify a test sentence. We use cosine similarity to find the k examples that are (semantically) closest to the test sentence in the SBERT embedding space. We find that this approach of selecting examples performs better than random sampling.

Evaluation of Different Frameworks. We adapt each framework on the *EvocativeLines* dataset and use test accuracy and AUC scores as primary indicators of the predictive power of different models. We report these results in Table 7.4. Further, we test our proposed framework on the *EvocativeLines (small)* dataset with 10 diverse sets of preference profiles, and report results in Table 7.5.

Parameter	Search Space	Optimal Value
input feature	$\{\mathbf{S}_{\text{all}}, \mathbf{S}_{\text{flat}}\}$	$\mathbf{S}_{\text{all}}$
bi-directional	$\{\text{True}, \text{False}\}$	True
attention type	$\{\text{global}, \text{general}, \text{self}\}$	global
layers	$\{1, 2, 3\}$	1
hidden dimensions	$\{8, 16, 32, 64, n/2\}$	n
learning rate	$10^{-i} \forall i \in \{1, \dots, 5\}$	10^{-4}

Table 7.3: Search space for grid search of LSTM. **Note:** n is from Equation 7.20

Implementation Details We implement the LSTM classifiers in PyTorch (Paszke et al., 2019) and train them for 500 epochs with early stopping (we stop a training run if the validation performance does not improve for 10 consecutive epochs). We perform grid search to find optimal values over the parameters listed in Table 7.3, and find a *one*-layer *bi*-directional LSTM classifier trained with a learning rate of 0.0001 on $\mathbf{S}_{\text{all}}$ input features performs best.

We use the `sentence-transformers`⁴ python package to obtain SBERT embeddings of sentences and use Huggingface’s `transformers`⁵ library to implement the fine-tuning of BERT classifier. Specifically, we finetune the BERT classifier for 500 steps with a weight

⁴<https://www.sbert.net/>

⁵<https://huggingface.co/docs/transformers/index>

decay of 0.01. We use the `sklearn` python package (Pedregosa et al., 2011) to implement all the ML-based classifiers. We use the default hyperparameters for all the models. Finally, we use a batch size of 32 for all the models.

For the LLaMA-3 classifier, we use the categorization of “inspiring” and “not inspiring” from the annotations from $\mathcal{A}_1$ and $\mathcal{A}_2$ (we randomly select which annotator to choose for every test sentence).

Finally, we perform 5-fold cross-validation (except for the LLaMA-3 classifier), repeat all our experiments for 5 random seeds, and report the mean and std values in our result tables.

Method	$\mathcal{A}_1$		$\mathcal{A}_2$	
	Accuracy	AUC	Accuracy	AUC
<i>Majority baseline</i>	<i>51.5</i>	<i>50.0</i>	<i>53.0</i>	<i>50.0</i>
LSTM+Attn ($\mathbf{S}_{\text{all}}$)	79.5 ^(0.4)	79.7 ^(0.1)	78.3 ^(0.3)	80.1 ^(0.3)
LSTM+Attn (SBERT)	71.6 ^(0.3)	71.7 ^(0.1)	69.9 ^(0.2)	70.1 ^(0.3)
BERT	77.6 ^(2.1)	78.8 ^(1.7)	75.2 ^(2.3)	76.1 ^(1.1)
RF	80.5 ^(1.7)	80.6 ^(1.7)	72.4 ^(2.0)	73.0 ^(1.2)
SVM	70.5 ^(1.2)	70.5 ^(1.0)	67.3 ^(2.3)	63.1 ^(1.1)
LR	71.1 ^(1.9)	71.1 ^(1.8)	64.3 ^(1.9)	62.0 ^(1.4)
DT	84.2 ^(2.5)	84.3 ^(2.3)	84.1 ^(1.0)	80.0 ^(1.7)
XGBoost	92.2 ^(1.7)	92.2 ^(1.6)	87.32 ^(1.4)	83.9 ^(1.2)
LLaMA (200-shot)	73.3 ^(1.7)	73.3 ^(1.6)	62.3 ^(3.6)	65.1 ^(1.6)
LLaMA (300-shot)	73.9 ^(2.1)	74.2 ^(1.8)	64.1 ^(1.2)	65.8 ^(1.4)
LLaMA (450-shot)	74.8 ^(3.1)	74.8 ^(3.0)	69.5 ^(3.5)	70.5 ^(1.6)

Table 7.4: Comparison of different frameworks on *EvocativeLines*. The first section shows the performance of DL-based methods, the second section shows the performance of ML-based methods, and the last section shows the performance of LLM-based methods.

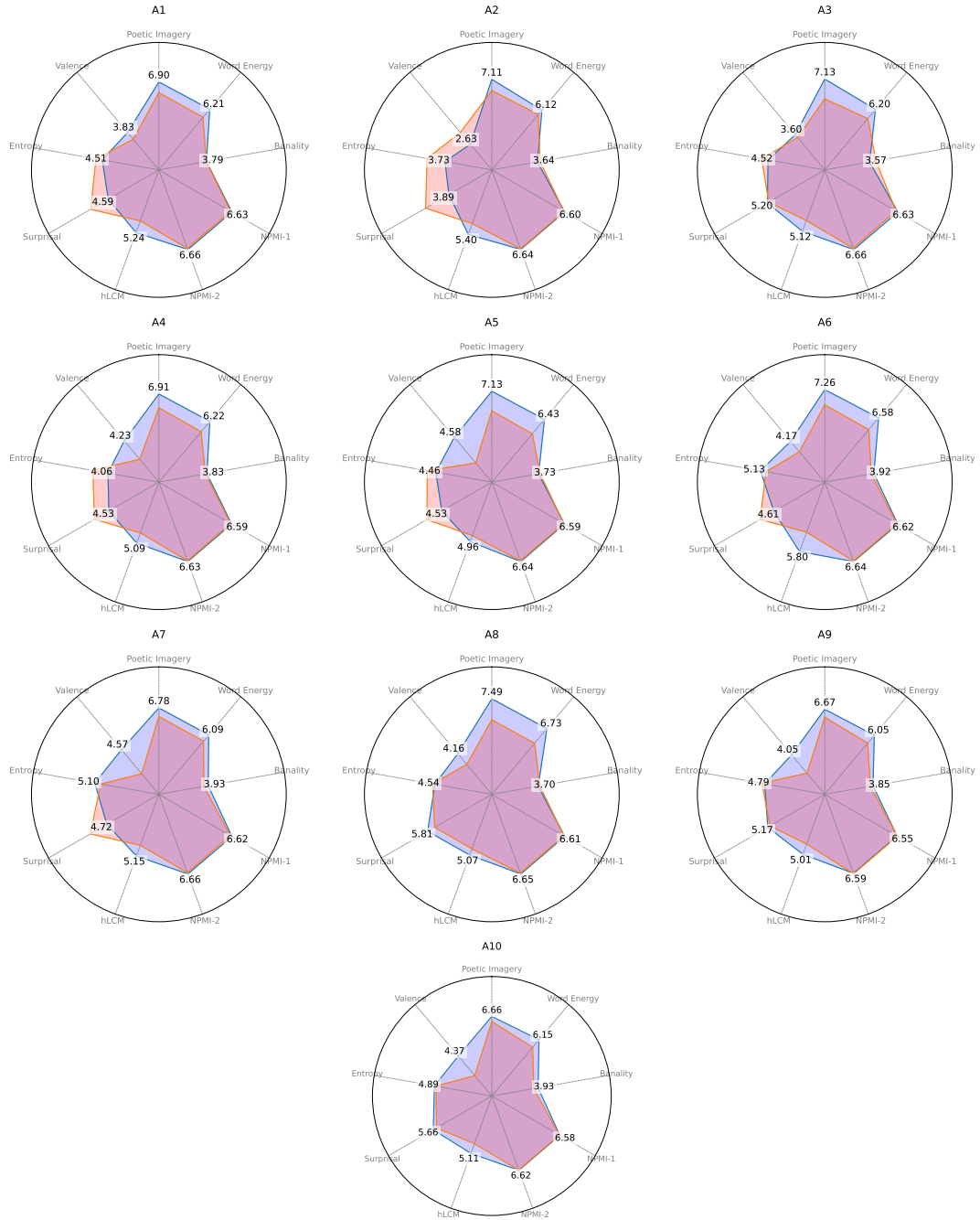


Figure 7.2: Radar charts showing the 10 different preference profiles of annotators in the *EvocativeLines* dataset. Blue webs denote the web for positive lines, and red webs denote the negative lines.

User	Majority		LSTM+Attn		XGBoost	
	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
$\mathcal{A}_1$	50.0	50.0	75.6 ^(1.2)	64.9 ^(4.0)	70.7 ^(5.3)	70.2 ^(5.3)
$\mathcal{A}_2$	70.5	50.0	80.5 ^(1.2)	62.3 ^(6.1)	69.3 ^(9.5)	57.1 ^(8.4)
$\mathcal{A}_3$	58.5	50.0	94.1^(1.2)	92.3^(2.1)	63.4 ^(11.1)	62.6 ^(11.7)
$\mathcal{A}_4$	60.5	50.0	68.7 ^(3.1)	61.4 ^(4.5)	59.5 ^(4.5)	56.4 ^(6.6)
$\mathcal{A}_5$	58.0	50.0	80.4 ^(2.5)	84.8 ^(5.1)	71.7 ^(6.7)	70.5^(6.6)
$\mathcal{A}_6$	64.5	50.0	75.6 ^(3.1)	75.4 ^(2.8)	68.8 ^(3.2)	62.8 ^(3.0)
$\mathcal{A}_7$	58.0	50.0	75.6 ^(4.3)	82.3 ^(4.2)	61.0 ^(4.4)	60.3 ^(4.2)
$\mathcal{A}_8$	62.5	50.0	80.5 ^(5.4)	78.2 ^(5.1)	72.2^(5.0)	68.5 ^(7.1)
$\mathcal{A}_9$	70.0	50.0	76.5 ^(4.7)	72.4 ^(3.6)	70.7 ^(4.9)	57.3 ^(5.1)
$\mathcal{A}_{10}$	58.5	50.0	82.3 ^(2.3)	84.2 ^(3.5)	62.4 ^(5.9)	60.7 ^(5.0)
Avg.	61.1	50.0	79.0^(6.3)	75.8^(9.9)	67.0 ^(4.6)	62.6 ^(5.1)

Table 7.5: Performance of the proposed framework with LSTM+Attn and XGBoost calibration networks on the *EvocativeLines (small)* dataset.

7.5 Results

7.5.1 Quantitative Results

Proposed Linguistic and Poetic Features Have a High Predictive Power

From Table 7.4, we note that our proposed framework with ML-based and DL-based calibration networks outperforms a 450-shot LLaMA-3-70b classifier. Specifically, our XGBoost model with just 163 parameters achieved a test accuracy of 92.2 for $\mathcal{A}_1$ and 87.3 for $\mathcal{A}_2$, compared to 450-shot LLaMA-3 classifier with 70B parameters that achieves an accuracy of 74.8 and 69.8 for $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively. Note that 450 was the maximum number of positive and negative examples we were able to add to the prompt of the LLaMA-3-70b classifier. Among the DL-based models, the LSTM+Attn model (with roughly 110k parameters) outperforms a BERT model (with 110M params) fine-tuned on the *EvocativeLines* dataset. It also outperforms the LSTM+Attn (SBERT) baseline, showcasing the higher predictive power of the proposed features.

These higher predictive power of our features computed using an LLM compared to using the LLM out-of-the-box shows that while we can leverage the knowledge of LLMs to compute well-established linguistic and stylistic characteristics of a lyric line, the LLM itself

has difficulty in capturing the nuanced differences between what is perceived as “inspiring” and “not inspiring” by an individual. Overall, this highlights that our proposed framework that identifies the key linguistic and poetic features is more suitable for modeling artistic preferences of lyrical lines compared to using significantly larger models out-of-the-box.

Feature	$\mathcal{A}_1$		$\mathcal{A}_2$		$\mathcal{A}_3$		$\mathcal{A}_6$		$\mathcal{A}_7$	
	coef	p -value	coef	p -value	coef	p -value	coef	p -value	coef	p -value
$\bar{S}_{\text{surprisal}}$	-0.4149	0.107	0.288	0.404	0.4351	0.233	0.416	0.242	-0.2176	0.518
$\bar{S}_{\text{entropy}}$	-0.8377	0.034	0.0534	0.889	0.02	0.958	-0.0632	0.873	-0.0817	0.822
$\bar{S}_{\text{npmi,uni}}$	7.2427	0.000	9.7483	0.068	59.1835	0.003	75.9894	0.018	21.0617	0.152
$\bar{S}_{\text{npmi,bi}}$	-6.2588	0.002	-9.8577	0.552	-55.0718	0.007	-73.9488	0.024	-19.2738	0.224
$\bar{S}_{\text{abs}}$	0.5402	0.298	0.6791	0.047	0.7423	0.045	0.9499	0.012	0.7226	0.03
$\bar{S}_{\text{vbin}}$	0.3281	0.001	-103.4697	0.556	-197.0881	0.028	-264.4955	0.158	9.1933	0.957
$\bar{S}_{\text{imagery}}$	14.8346	0.000	0.6861	0.048	2.065	0.057	0.4584	0.643	1.1321	0.241
$\bar{S}_{\text{energy}}$	-0.0543	0.985	0.9405	0.031	1.3163	0.423	2.4855	0.086	1.8852	0.191
$\bar{S}_{\text{banality}}$	-2.8474	0.128	0.5269	0.636	-0.2816	0.804	0.1945	0.866	3.2007	0.003
$\bar{S}_{\text{I1}}$	0.0549	0.084	-0.1213	0.609	-0.0655	0.745	-0.2414	0.255	0.3546	0.116
$\bar{S}_{\text{I2}}$	-0.0188	0.499	13.8702	0.027	14.5632	0.142	6.5436	0.438	3.1841	0.72
$\bar{S}_{\text{all}}$	0.0760	0.025	-1.0131	0.022	-0.3216	0.31	-0.2537	0.306	1.3457	0.145

Table 7.6: Results of interaction testing. I1 denotes the interaction between linguistic features (surprisal, entropy, banality, npmi), and I2 denotes the interaction between stylistic features (imagery, word energy, abstraction, valence).

The Proposed Framework is Robust to Diverse User Preferences

Table 7.5 shows the results for our experiments on *EvocativeLines (small)* that contains 10 preference profiles captured by the 10 annotators. Specifically, we note that our framework with LSTM+Attn and XGBoost calibration network can adapt to the diverse range of artistic preferences captured in the *EvocativeLines (small)* dataset, with the LSTM+Attn variant achieving a test accuracy of $79.0 \pm 6.3\%$ average across the 10 annotators and the XGBoost variant achieving $67 \pm 4.6\%$ average test accuracy. We notice a significant drop in the performance of XGBoost relative to the performance on the full dataset, which can be explained by the fact that while XGBoost is a flexible model, it might suffer from overfitting as we shift to data-scarce setups. Overall, these results highlight the robustness of our framework against the high degree of variations across different preference profiles, and we conclude that the proposed framework can be successfully employed to forecast the artistic preferences of different individuals, even in low-data setups, by choosing an appropriate calibration network.

Significance of Individual Features

To understand the feature interactions at a more fine-grained level and how they control the final outcome, we conduct statistical testing to determine the correlations amongst the individual features and their influence on the output variable (whether or not an individual found a lyric line inspiring). Specifically, we fit a logistic regression model with interaction terms to evaluate the significance of each independent variable (feature) on the binary outcome variable ⁶. We simplify our features for this test by taking a mean across the timesteps dimension so that each feature is converted into a score. While this simplified representation is not truly representative of our actual features, it serves as a good proxy. We show the results of significant testing in Table 7.6 for $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$, $\mathcal{A}_6$, and $\mathcal{A}_7$. For brevity, we only show results for five annotators. We chose $\mathcal{A}_1$ and $\mathcal{A}_2$ as their annotations allow us to study interactions in a data-rich setting, and we chose $\mathcal{A}_3$, $\mathcal{A}_6$, and $\mathcal{A}_7$ as this triplet has one of the lowest inter-annotator agreement in the smaller version of our dataset (refer to Figure 7.1). We also study three composite interactions: **a**) linguistic interaction between surprisal, entropy, banality, and npmi (uni and bi) (denoted by $\bar{\mathbf{S}}_{\mathbf{I1}}$); **b**) poetic and artistic interaction between imagery, word energy, level of abstraction, and valence (denoted by $\bar{\mathbf{S}}_{\mathbf{I2}}$); and **c**) interaction among all the features (denoted by $\bar{\mathbf{S}}_{\mathbf{all}}$).

We report the results of our interaction study in Table 7.6. Starting with $\mathcal{A}_1$, we note that $\bar{\mathbf{S}}_{\mathbf{entropy}}$ has a low p -value of 0.034 (< 0.05) and a negative coefficient of -0.8377. Therefore, we conclude that contextual entropy has a *significantly negative* correlation with the outcome variable. In other words, as the entropy of a lyric line increases, $\mathcal{A}_1$ is likely to find that lyric line evocative. We follow a similar analysis pattern to conclude that $\mathbf{S}_{\mathbf{npmi,uni}}$ is *significantly positively* correlated and $\mathbf{S}_{\mathbf{npmi,bi}}$ is *significantly negatively* correlated with the outcome. Simultaneously, poetic imagery has a *significantly positive* correlation with a high coefficient of 14.8346. This contradicting preference in word associations based on directionality coupled with a high preference for imagery indicates a strong inclination for causality in lyrics, where the line has a clear linear progression but loses its creative mark upon reversing the chain of events. Similarly, we conclude that $\mathbf{S}_{\mathbf{vbin}}$ is *significantly positively* correlated with the outcome variable, indicating a preference for positive lines by $\mathcal{A}_1$. Finally, we find $\mathbf{S}_{\mathbf{all}}$ to be *positively* correlated with the outcome variable. We conduct a similar analysis for the other annotators and highlight the significant features for each of them in Table 7.6. These results show that each annotator’s artistic preferences were influenced by a different set of features, and our framework can be used to explain aesthetic preferences in an interpretable manner.

⁶we use `statsmodels` python package to implement logistic regression with interaction terms.

7.5.2 Qualitative Examples

We include a radar chart of different preference profiles in Figure 7.2 that shows how the artistic preferences of different individuals vary from one another. We construct a preference profile by computing the average value for the positive and negative samples across the different feature dimensions we have. We denote the profile of positive samples in blue and negative samples in red. We notice that all the preference profiles show a visibly different distinction between the positive and negative examples. Most prominently, all profiles have a higher imagery and energy value for the positive examples compared to the negative examples. Additionally, we also show radar charts in Figure 7.3 that visualize some positive and negative examples based on their feature values along the different feature dimensions. These charts show how different concepts in the lyrical content influence the different dimensions of the features. For instance, “all the world is the fire” (row 1 column 2) has a 0 valence value but high imagery and energy values.

7.6 Limitations

Although our proposed framework is highly interpretable and is useful for modeling artistic preferences in lyrical lines, it has some limitations. For instance, we compute the NPMI metric based on *EvocativeLines* corpus, which might not be representative of the entire English language. We can address this limitation by using a larger corpus to compute the NPMI metric.

Additionally, we use an LLM to measure some stylistic features, such as imagery. While our experiments show that these features are crucial in modeling artistic preferences, we did not conduct extensive prompt optimization, and further optimization might lead to better results.

7.7 Conclusion

In this chapter, we propose an interpretable and flexible framework for computational modeling of artistic inspiration in the context of AI-generated lyrical lines that can be used to assist artists in the Seed phase of creation by identifying potentially “inspiring” ideas (or seeds) amongst a larger pool of candidates. We identified the key linguistic and stylistic features that can accurately forecast aesthetic preferences among a diverse set of individuals. Our experiments demonstrate that the predictive power of our framework

surpasses that of several orders of magnitude larger state-of-the-art language models like LLaMA-3-70b.

Crucially, our approach is not limited to the analysis of lyrics and poetry and can be adopted (with a different or expanded set of features) to predict other aesthetic, literary, or artistic preferences that are inherently subjective (for instance, comparing sonnets from multiple different artists). Overall, in the context of creating art, it is crucial to measure how well AI systems support individual creative processes that can encompass a wide spectrum of artistic preferences and skill levels, and through this work, we hope to provide a foundation for future research in this area.

7.8 Key Takeaways

To summarize, the key takeaways from this chapter are as follows:

1. We can use linguistic and stylistic features that can help in accurately modeling artistic preferences in lyrical lines.
2. The framework is highly interpretable and is applicable across a wide range of creative individuals, regardless of their skill level or aesthetic preferences.
3. LLMs struggle with capturing nuanced differences in artistic preferences, even with a large number of parameters and examples, but they are useful for the more fundamental task of extracting linguistic and stylistic features that can be used to model artistic preferences.

In the next chapter, we will progress to a more complex and emerging task of automated data analysis, where we will explore the effectiveness of LLMs in navigating the different stages of a data analysis pipeline.



Figure 7.3: Examples of lyric lines rated as inspiring (blue) and not inspiring (red) by annotator $\mathcal{A}_1$.

Chapter 8

LLMs for Advanced Data Analytics

So far, in this thesis, we have discussed LLMs on single-step tasks, where the LLM only needs to take one specific action to solve a task. For instance, for data augmentation in text classification, the LLM needs to perform the single task of generating new samples; when using LLMs as a classifier (like for relabeling in PromptMix) or a critic (like scoring source sentences in extractive summarization or measuring imagery of a lyric line), the LLM needs to perform the single task of outputting a score. However, real-world scenarios often involve multi-step tasks, where the agent needs to perform a sequence of actions, where each action is dependent on the previous one. Therefore, in this chapter, we explore how LLMs can be utilized for more complex tasks requiring LLMs to operate as part of a multi-step system.

First Steps Towards Agentic AI Systems Agentic AI Systems are autonomous, goal-driven models that actively pursue specific objectives in a dynamic, often unpredictable environment. Agentic AI can be described as a four-step process for problem-solving ([NVIDIA, 2023](#)):

1. **Perceive:** AI agents gather and process data from various sources, such as sensors, databases and digital interfaces. This involves extracting meaningful features, recognizing objects, or identifying relevant entities in the environment.
2. **Reason:** An LLM acts as the reasoning engine that understands tasks, generates solutions, and coordinates specialized models for specific functions like content creation, vision processing, or recommendation systems. This step uses techniques like RAG to access proprietary data sources and deliver accurate, relevant outputs.

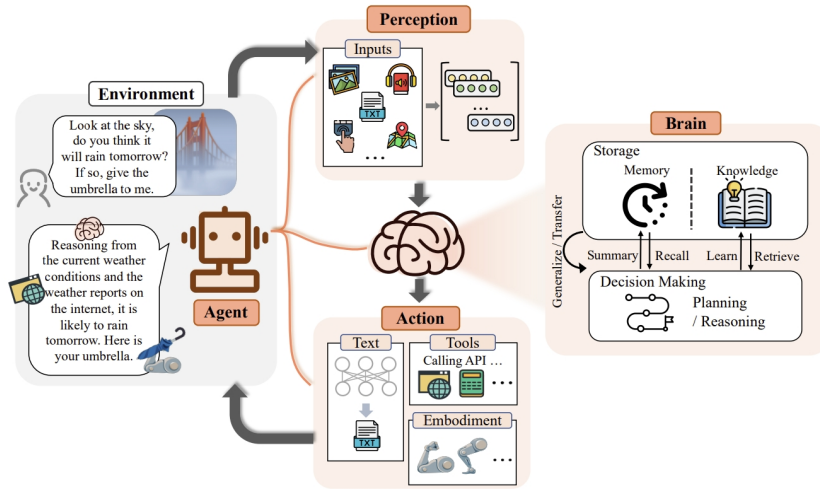


Figure 8.1: Illustration of an agentic AI system. Image Source: <https://github.com/Jenqyang/Awesome-AI-Agents>

3. **Act:** By integrating with external tools and software via application programming interfaces (APIs), agentic AI can quickly execute tasks based on the plans it has formulated. Guardrails can be built into AI agents to help ensure they execute tasks correctly. For example, a customer service AI agent may be able to process claims up to a certain amount, while claims above the amount would have to be approved by a human.
4. **Learn:** Agentic AI continuously improves through a feedback loop, where the data generated from its interactions is fed into the system to enhance models. This ability to adapt and become more effective over time offers businesses a powerful tool for driving better decision-making and operational efficiency.

The applications of agentic AI span from tasks like improving customer service chatbots by enhancing self-service capabilities and automating routine queries, boosting developer productivity by automating repetitive coding tasks, and distilling critical information from big chunks of medical and patient data to help doctors make more informed decisions.

In this thesis, we consider the utility of AI agents for autonomous data analytics, a crucial task in today's data-driven world. We choose data analytics as a testbed for LLMs in agentic systems as it demands diverse skill sets to accomplish a goal. For instance, the task of end-to-end data analysis starts with stating a high-level goal that a data analyst

uses to identify key questions to explore and then extracts valuable insights from their answers, working towards the goal. This iterative process continues until they have a comprehensive summary of insights and recommended actions. Automating such complex workflows demands the system to perform multiple tasks together. For instance, the agent first needs to generate relevant questions to explore. Then, it needs to generate code that performs the analysis required to answer the question. Then, it needs to interpret the output of the code to obtain meaningful insights and, finally, repeat this process until it has a comprehensive summary of insights and recommended actions. There is a rising demand for such agents as they could significantly reduce the manual effort required to interpret large datasets, thereby enabling faster, data-informed decision-making across domains (McAfee and Brynjolfsson, 2012; Colson, 2019; Bean, 2022). Our focus is on understanding how such agents can autonomously perform end-to-end analytics—from data ingestion and preprocessing to analysis and visualization—while maintaining high accuracy and reliability. To that end, we introduce *AgentPoirot*, an agentic system for end-to-end data analysis on a given dataset. *AgentPoirot* generates high-level questions, answers them, and then summarizes the insights obtained.

Recent work on agents based on LLMs has emerged as promising assistants to users in performing data science tasks (OpenAI, 2022, 2023a). But these agents (such as Pandas Agent (LangChain, 2024) and Code Interpreter (OpenAI, 2024)) only focus on solving narrow, single-step code-completion (or data analytics) tasks. In the context of the data analytics process, that is equivalent to performing a single action like answering a specific question (E.g., “What is the R^2 value for a linear regression model on this dataset?”) rather than following the entire data analytics process. One of the reasons most existing agents have limited data analysis capabilities is because current benchmarks focus on evaluating agents on single-step code-completion tasks (Lai et al., 2022; Chen et al., 2021b; Hu et al., 2024). Another reason is that it is challenging to evaluate LLM-based agents accurately without including substantial information about the problem details, expected code structure, and expected outputs (such as identifying variable correlation) in the prompt. Recent works such as InsightPilot (Ma et al., 2023) and InsightLens (Weng et al., 2024) propose agents that can perform end-to-end data analytics, but they were evaluated using human assessments, which require significant effort and often lack consistency due to variability in individual biases and interpretations of evaluation criteria.

We evaluate *AgentPoirot* on *InsightBench* (Sahu et al., 2024), which consists of 100 datasets focused on business operations and workflows. Each dataset is accompanied by a set of ground truth goals, questions, and a summary of insights emulating findings that a human data analyst would typically extract from the data. Specifically, we use L-Eval (similar to how we used it in Chapters 5 and 6) to evaluate the quality of the insights

generated by *AgentPoirot* with respect to the ground truth insights in the benchmark. We also benchmark the current state-of-the-art LLM-based agent for data analytics, Pandas Agent [LangChain \(2024\)](#), that can perform end-to-end data analysis by recommending questions and extracting insights from their answers.

To summarize, we make the following contributions in this chapter:

- We present a comprehensive set of experiments contrasting open-source and closed-source models and different prompting strategies for evaluation on *InsightBench*.
- Our results show that *AgentPoirot* outperforms existing analytics methods such as Pandas Agent for the task.
- We also show that LLaMA-3-Eval is a feasible alternative to using the closed-source G-Eval for evaluating agentic pipeline for data analytics.

In the following section, we present more details about *InsightBench* and *AgentPoirot*.

8.1 Experimental Setup

8.1.1 Datasets

We use *InsightBench*, an automated data analysis benchmark that can rigorously assess the performance of LLM-based data analysis agents in real-world scenarios. *InsightBench* covers a distinct range of business analytics themes and varying difficulty levels, ensuring a comprehensive evaluation of the agents. The benchmark includes 100 datasets with a total of 475 insights spread across five key topics. These topics were chosen to reflect typical scenarios encountered in enterprise settings, each requiring clear and well-defined insights. The distribution across the topic is shown in [Figure 8.2](#).

Each dataset in the benchmark consists of 500 synthetically generated entries stored as a CSV file. Moreover, each dataset has systematic anomalies and trends embedded in it. For instance, one dataset exhibits a trend where the time-to-resolution (TTR) of an incident increases over time, denoting a potential issue like backlogs or understaffing in the incident management system. Specifically, every dataset comes with an expert-annotated Jupyter notebook containing ground truth insights, questions, and a summary of insights.

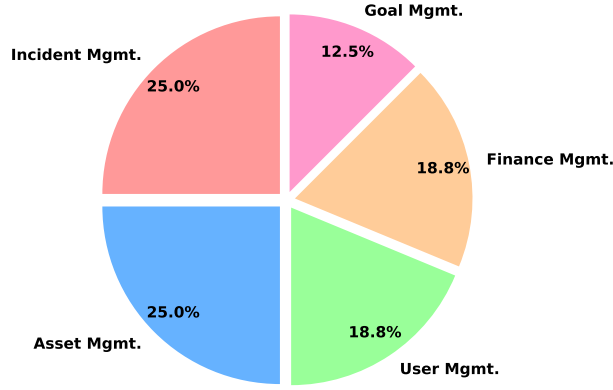


Figure 8.2: This diagram illustrates the scope of *InsightBench*, categorizing 100 distinct datasets into five key thematic areas.

8.1.2 Evaluation

To evaluate agents on *InsightBench*, we compare the ground-truth annotations (*GT*) in Jupyter notebooks against the list of insights (*A*) provided by an agent. Formally speaking, we compare two natural language texts. Therefore, we adopt an LLM-based evaluator, LLaMA-3-Eval, to measure the similarity between the ground truth and agent-provided insights. We use LLaMA-3-Eval as the primary metric to measure the correctness of agent-provided insights. We chose LLaMA-3-Eval over G-Eval because LLaMA-3-70b-Instruct is open-sourced, allowing us to a) avoid API costs for using GPT models from OpenAI and b) fix the model weights to obtain a stable and reliable evaluation metric, unlike G-Eval, whose output scores can change due to periodic updates to GPT endpoints.

Since the ground truth notebooks contain both individual insights and a final summary, we perform a two-way evaluation:

1. *Summary-level.* To obtain the summary-level score, we simply compute the LLaMA-3-Eval score between the agent-provided summary of insights and the ground-truth summary.

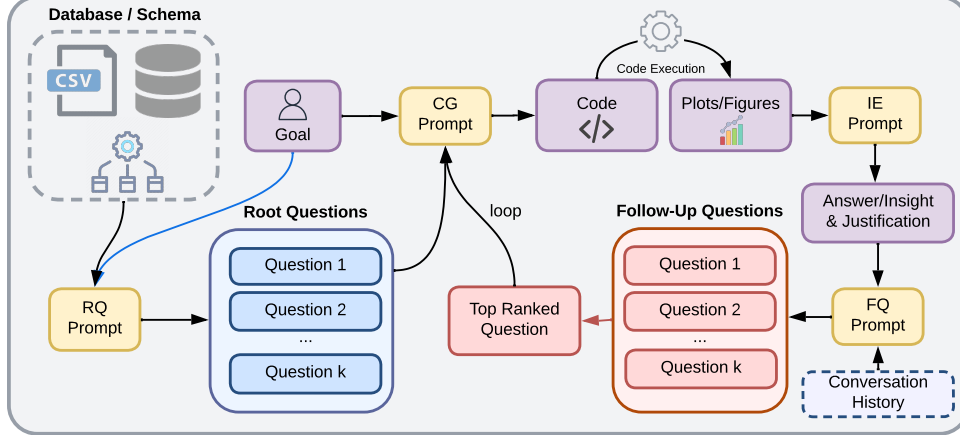


Figure 8.3: *AgentPoirot* uses the “Question Generation (or **QG**) Prompt” (Prompt C.4 in Appendix C) that takes as input the dataset schema and the goal to generate k high-level questions. Then, it uses the “Code Generation (or **CG**) Prompt” (Prompt C.5 in Appendix C) to generate plots answering the high-level questions. Using the “Insight Extraction (or **IE**) Prompt” (Prompt C.8 in Appendix C), and the outputs from the previous step, it generates an *insight* and the *justification* of that insight. Finally, *AgentPoirot* uses the “Follow-Up Question (**FQ**) Prompt” (Prompt C.6 in Appendix C) to generate diverse follow-up questions and selects the top one using the “Question Selection (or **QS**) Prompt” (Prompt C.9 in Appendix C). This question is injected back into the **CG** prompt to start the cycle again.

2. *Insight-level*. First, we select the most appropriate insight $a \in A$ to evaluate against a given ground-truth insight ($gt \in GT$) and then average the scores for all the ground-truth insights. The insight-level score can be expressed as in Equation 8.1, where $|GT|$ is the number of ground-truth insights in a dataset, and $\mathcal{M}$ represents LLaMA-3-Eval evaluator:

$$score = \frac{\sum_{gt \in GT} \arg \max_{a \in A} \mathcal{M}(gt, a)}{|GT|} \quad (8.1)$$

We average the summary-level and insight-level scores for all 100 datasets in *Insight-Bench* to obtain a measure of the agent’s performance. Additionally, we compute scores using ROUGE-1 for comparison.

8.2 Methodology: AgentPoirot

We propose *AgentPoirot* that, given a dataset ($\mathcal{D}$) in *InsightBench* and a goal ($\mathcal{G}$), performs systematic data exploration aimed at achieving $\mathcal{G}$. Figure 8.3 shows the working of *AgentPoirot*.

1. **Extract Dataset Schema ($\mathcal{S}$):** The dataset schema contains information about every column in the dataset. *AgentPoirot* extracts the name, type, total unique values, and total NA values for every column. Additionally, if a column contains numerical data, it extracts the minimum, maximum, mean, and standard deviation; if a column contains dates, it extracts the minimum and maximum dates; otherwise, it extracts the top 5 unique values from the column.
2. **Generate Insights:** Given the triplet $(\mathcal{D}, \mathcal{G}, \mathcal{S})$, *AgentPoirot* uses the Question Generation Prompt to generate $k(= 3)$ high-level questions. It answers each high-level question by first generating Python code and then interpreting the output of the code. Then, it generates $n(= 4)$ follow-up questions for each high-level question and then answers them as well to obtain a total of generates $(n + 1) \times k$ insights(see Figure 8.3 to see a complete overview of *AgentPoirot*).
3. **Generate Summary:** *AgentPoirot* summarizes the insights from the previous step (using Prompt C.10.)

8.2.1 Baselines

We run the following data analytics agents on InsightBench:

a) **Pandas Agent (PA)** (LangChain, 2024). A data science agent by LangChain optimized for question-answering¹. Given a data frame and a question related to it, Pandas Agent first generates Python code and then executes it to produce the answer. In our experiments, we input the goal to the agent and let it iteratively generate questions and their answers. We then pass the list of answers to the agent again and ask it to generate a summary of those answers.

b) ***AgentPoirot* (Ours).** *AgentPoirot* as described in the previous section. Figure 8.3 shows the working of *AgentPoirot*.

¹LangChain is an MIT Licensed Python library for building context-aware reasoning applications

c) **AgentPoirot (Ours) w/ generic goal.** We use the generic goal, “I want to find interesting insights in this dataset,” instead of the carefully designed goal and run *AgentPoirot* on *InsightBench* as in b).

d) **AgentPoirot (Ours) non-diverse follow-ups.** We use Prompt C.7 instead of Prompt C.6 to generate only one type of follow-up questions and run *AgentPoirot* on *InsightBench* as in (b). Refer to Figure 8.3 in the Appendix for a visualization of our proposed method.

In addition to PandasAgent, we also considered the following baselines: Insightpilot (Ma et al., 2023), but neither the code nor the prompts are available; Data-Copilot (Zhang et al., 2023d), but it is limited to Chinese financial data, which isn’t suitable for our task; OpenAgents (Xie et al., 2023), but the public version only allows ten requests, which isn’t enough for meaningful benchmarking; InfiAgent-Dabench (Hu et al., 2024), but we excluded it due to its extremely poor performance in this benchmark. It handles only simple tasks like “Generate Python code to compute the mean of a list,” but struggles with more complex queries like “What is the distribution of incidents by category in this dataset?” Lastly, we considered using PowerBI², but it is a closed-source tool that is not suitable for large-scale benchmarking. We conducted a small-scale evaluation of PowerBI and found that it obtained low-quality insights and generated inconsistent outputs for the same prompt. Our benchmark emphasizes transparency and reproducibility, therefore we focussed on methods that the scientific community can easily inspect and scrutinize, which is not possible with the aforementioned baselines.

8.2.2 Implementation Details

We use Python’s `openai` package to access the family of GPT models for our experiments and `vllm` to host LLaMA-3-70b³ model on 4 A100 GPUs. We use different LLMs as backbones in our experiments to benchmark, including `gpt-4o`, `gpt-4-turbo`, `gpt-3.5-turbo`, and `llama-3-70b`⁴. All results are reported for the sampling temperature of 0.0, unless otherwise stated. We use the `evaluate` Python package to compute ROUGE-1 scores and Prompt C.12 in Appendix C for computing LLaMA-3-Eval scores. We also fix the temperature of LLaMA-3-70b-Instruct to 0 during evaluation. We repeat all our experiments for 5 seeds and report the mean and standard deviation in our results.

²<https://app.powerbi.com/home>

³<https://github.com/meta-llama/llama3>

⁴We use `gpt-4-turbo-2024-04-09` and `gpt-3.5-turbo-0125`, specifically.

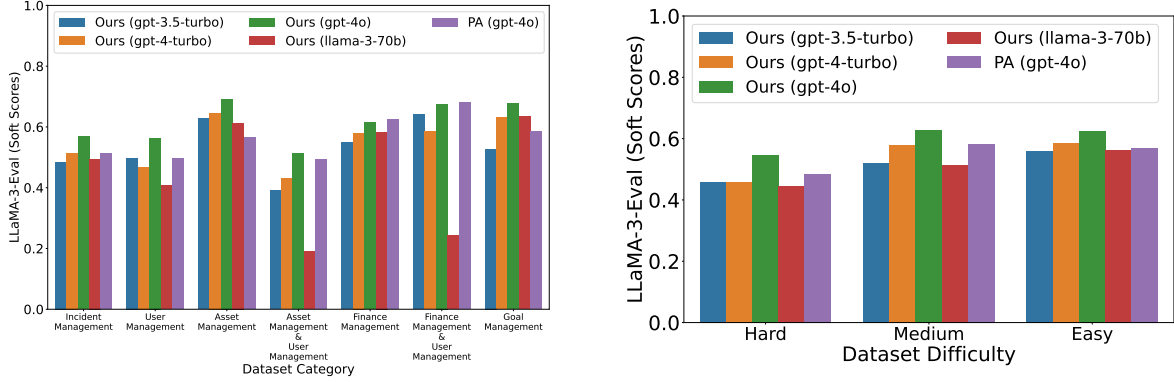
Table 8.1: Performance of different agents on *InsightBench*. All results are for 5 different seeds.

Agent	Insight-level Scores		Summary-level Scores	
	ROUGE-1	LLaMA-3-Eval	ROUGE-1	LLaMA-3-Eval
PA (gpt-4o)	0.35 ± 0.03	0.54 ± 0.01	0.35 ± 0.01	0.40 ± 0.04
Ours (gpt-4o)	0.32 ± 0.02	0.60 ± 0.03	0.37 ± 0.09	0.44 ± 0.03
- w/ generic goal	0.30 ± 0.03	0.40 ± 0.03	0.30 ± 0.08	0.33 ± 0.12
Ours (gpt-4-turbo)	0.30 ± 0.02	0.56 ± 0.02	0.35 ± 0.08	0.35 ± 0.04
- w/ generic goal	0.28 ± 0.01	0.38 ± 0.03	0.29 ± 0.02	0.27 ± 0.11
Ours (gpt-3.5-turbo)	0.34 ± 0.01	0.50 ± 0.02	0.27 ± 0.14	0.31 ± 0.06
- w/ generic goal	0.30 ± 0.02	0.36 ± 0.03	0.24 ± 0.03	0.25 ± 0.06
Ours (llama-3-70b)	0.33 ± 0.02	0.52 ± 0.04	0.36 ± 0.01	0.33 ± 0.01
- non-diverse follow-ups	0.29 ± 0.01	0.51 ± 0.03	0.32 ± 0.01	0.28 ± 0.03
- w/ generic goal	0.27 ± 0.03	0.35 ± 0.01	0.23 ± 0.03	0.23 ± 0.02

8.3 Results

Main Results. Table 8.1 shows the performance of different data analytics agents on *InsightBench*. First, we note that *AgentPoirot* outperforms Pandas Agent in terms of ROUGE-1 and LLaMA-3-Eval scores. We note that using **gpt-4o** consistently achieves the best overall performance. We further see that *AgentPoirot* with **LLaMA-3-70b** outperforms *AgentPoirot* with **gpt-3.5-turbo** and is close to **gpt-4-turbo** in terms of LLaMA-3-Eval scores.

Figure 8.4a shows the performance of the agents by different dataset categories. Agents achieve the highest scores in “Asset Management,” “Goal Management,” and “Finance Management” categories. Notably, *AgentPoirot* with **LLaMA-3-70b-Instruct** struggles when we combine two datasets (“Finance Management & User Management” and “Asset Management & User Management”), which might be due to the small context size of **LLaMA-3-70b-Instruct**. On the other hand, Pandas Agent surprisingly performs the best in those dataset categories. Figure 8.4b shows the performance of agents by the difficulty level of the dataset. Intuitively, agents achieve strong performance on “Easy” and “Medium” level datasets, and their performance drops for the “Hard” level datasets. We also show some qualitative examples in Table B.2, where we compare the outputs generated by *AgentPoirot* and PandasAgent.



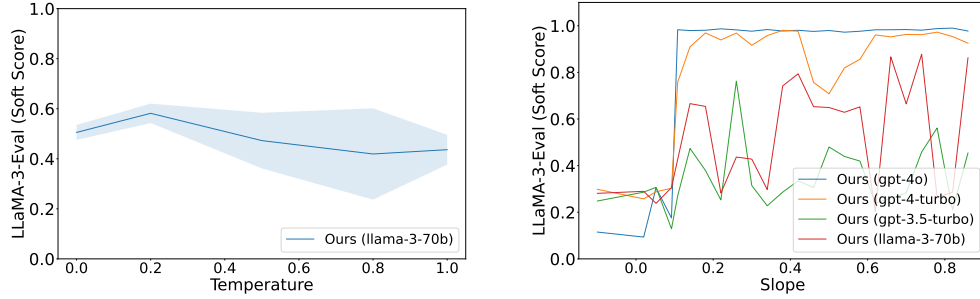
(a) Different agents’ performance by dataset (b) Different agents’ performance by dataset difficulty.

Figure 8.4: Performance of different agents on *InsightBench* grouped by difficulty and dataset category.

Effect of Using Generic Goals. Table 8.1 also shows the performance of *AgentPoirot* that uses a generic goal. We notice a drastic overall decrease in the ROUGE-1 and LLaMA-3-Eval scores for all the backbones. For instance, the insight-level LLaMA-3-Eval score for *AgentPoirot* (gpt-4o) drops from $0.59 (\pm 0.01)$ to $0.39 (\pm 0.02)$ and the summary-level LLaMA-3-Eval score drops from $0.41 (\pm 0.02)$ to $0.31 (\pm 0.13)$ (see “Ours (gpt-4o)” v/s “Ours (gpt-4o) w/ generic goal” rows in Table 8.1). This highlights the importance of including a meaningful goal in the agent’s performance.

Effect of Exploring Diverse Questions. Table 8.1 includes the performance of *AgentPoirot* when generating only a single type of follow-up questions (see “Ours (llama-3-70b) non-diverse follow-ups”). We observe a notable decrease in ROUGE-1 scores and a slight decrease in LLaMA-3-Eval scores, confirming that discovering a diverse set of questions during data analysis leads to better performance.

Effect of Trend Intensity Figure 8.5b shows the performance of *AgentPoirot* with different backbones on variations of dataset 2 in *InsightBench*. First, we note that all the models fail to discover insight when the slope is less than 0.1 (including negative slopes). This suggests that our model did not generate false positives (we assume the flag is discovered if LLaMA-3-Eval is greater than 0.5). Overall, for slope greater than 0.1, gpt-4o and gpt-4-turbo discovered the insight every time, LLaMA-3-70b-Instruct discovered it



(a) Effect of varying sampling temperature (performance reported across the 100 datasets in *InsightBench*) (b) Effect of varying trend intensity (for a specific insight in dataset 2).

Figure 8.5: Sensitivity Analysis.

nearly half the times, and `gpt-3.5-turbo` had a particularly hard time discovering the insight. While both `LLaMA-3-70b-Instruct` and `gpt-3.5-turbo` have difficulty discovering the trend sometimes (as showcased by the fluctuations in `LLaMA-3-Eval` scores), `LLaMA-3-70b-Instruct` shows a better overall discovery rate of the insight compared to `gpt-3.5-turbo`.

Effect of Sampling Temperature. Figure 8.5a shows the performance of *AgentPoirot* with `LLaMA-3-70b-Instruct` for different sampling temperatures. The performance of the agent peaks at temperature 0.2 before it starts to decrease. Higher temperatures also lead to unstable performance, as indicated by bigger standard deviations in the plot. Overall, we note that the temperature should be low but not completely 0 for optimal agent performance on *InsightBench*.

Agent	Soft Scores			Summary Scores	
	G-Eval	LLaMA-3-Eval	LLaMA-3-Eval (many-to-many)	G-Eval	LLaMA-3-Eval
Ours (gpt-4o)	0.53 $\pm$ 0.01	0.57 $\pm$ 0.02	0.43 $\pm$ 0.03	0.55 $\pm$ 0.01	0.58 $\pm$ 0.03
Ours (gpt-4-turbo)	0.52 $\pm$ 0.02	0.55 $\pm$ 0.02	0.40 $\pm$ 0.02	0.54 $\pm$ 0.03	0.58 $\pm$ 0.02
Ours (gpt-3.5-turbo)	0.46 $\pm$ 0.01	0.50 $\pm$ 0.01	0.35 $\pm$ 0.02	0.48 $\pm$ 0.01	0.50 $\pm$ 0.01
Ours (llama-3-70b)	0.45 $\pm$ 0.01	0.50 $\pm$ 0.02	0.34 $\pm$ 0.03	0.47 $\pm$ 0.01	0.49 $\pm$ 0.01

Table 8.2: Comparison of different evaluation methods on the first 10 datasets of *InsightBench*.

G-Eval v/s LLaMA-3-Eval Table 8.2 shows results for a small-scale study where we

compare G-Eval and LLaMA-3-Eval. We note that both scores are similar in range and follow the same trend for all agents. This suggests that LLaMA-3-70b is a good alternative to using gpt-4o for evaluation.

Ground Truth Insight	Prediction Matched by Many-to-Many Evaluation Scheme	Prediction Matched by One-to-Many Evaluation Scheme
The time to resolution of incidents is increasing over time	The average resolution time for incidents is 1677.27 hours for '1 - Critical' priority, 1554.63 hours for '2 - High' priority, and 1584.00 hours for '3 - Moderate' priority.	Yes, there is a clear trend of increasing incident resolution times over the months, with the average resolution time rising from 113.03 hours in January 2023 to 3150.86 hours in June 2024.
The time to resolution of incidents is uniform over time	Beth Anglin has shown the highest efficiency in resolving incidents with a mean resolution time of 153.90 minutes, and her workload has included handling 105 incidents over the given period.	There are no noticeable patterns in the types of incidents (categories) that are being resolved more quickly or slowly over time.
The volume of Hardware incidents is elevated during specific time windows	Yes, certain 'caller_id' individuals such as 'Don Goodliffe' report more incidents in the 'Hardware' category, and there is a trend showing that 'Bud Richman' and 'David Loo' also frequently report incidents in 'Hardware' and 'Software' categories.	Yes, there are patterns in the types of incidents reported over time, with 'Hardware' incidents peaking significantly in July and August.
There is a specific agent, Fred Luddy, who is assigned significantly more incidents than others	The assignment group with the highest number of incidents assigned to them is 'Network' with 310 incidents.	Fred Luddy consistently handles a higher number of incidents compared to others , with 385 incidents assigned to him.
There are higher expense rejection rates for employees with shorter tenure	Yes, the 'Assets' and 'Travel' categories have a higher rate of submission errors or rejections, with counts of 91 and 72 respectively, compared to 'Services' and 'Miscellaneous' which have 16 and 6.	The rejection rate of employees tends to be higher for employees with shorter employment durations , with many instances of 1.0 rejection rates observed in the initial days of employment, and it gradually decreases as employment duration increases.
There are higher success rates in Critical and High priority goals within the IT department	Yes, there are correlations between the metric types and the success rates of projects, as indicated by the plot statistics showing a range of data points for percent complete and target percentage of projects.	The dataset reveals a trend where projects with 'High' and 'Critical' priority levels generally have higher completion rates , often exceeding 80%, compared to those with 'Low' and 'Medium' priorities, which show more variability and lower completion rates overall.

Table 8.3: Mismatched examples by many-to-many compared to one-to-many evaluation scheme. Parts **in bold** indicate why the prediction is appropriate to compare against the ground truth insight.

One-to-Many v/s Many-to-Many Evaluation. We conduct another study to compare our approach with an alternative many-to-many version, where, instead of computing a score for each ground truth and prediction pair, we first prompt the LLM to match every ground truth response with an appropriate prediction. We perform this experiment because the one-to-many evaluation approach can be time-consuming as it needs to compute a LLaMA-3-Eval score for *every* ground truth-prediction pair to determine the most appropriate prediction to evaluate against a given ground truth and using many-to-many matching may result in a faster evaluation process.

Table 8.3 shows the quantitative results of our experiments. We note that using many-

to-many evaluation generally leads to a lower score; however, upon inspection, we find that many-to-many prompt often mismatches a ground truth insight with a wrong prediction (even though a more appropriate prediction was present in the pool of predicted insights). Table 8.3 shows some mismatches generated by the many-to-many evaluation approach compared to the proposed one-to-many evaluation approach that intuitively does not miss a good prediction if it is present in the list of generated insights.

8.4 Limitations

Benchmark Diversity. While *InsightBench* covers a spectrum of business areas and themes, the datasets are synthesized based on patterns observed in widely used enterprises. Consequently, the synthetic “flags” are designed to mimic real-world anomalies and trends but may not capture the full complexity or unexpected behavior of genuine enterprise data. The dataset currently focuses on structured tabular data, primarily in English, reflecting typical business records. This focus may not fully represent the rich diversity of global business practices or the variety of data interactions in different linguistic contexts.

Evaluation Metrics. We use ROUGE-1 and LLaMA-3-Eval as evaluation metrics for assessing the quality of insights generated by the agents. While these metrics provide a quantitative measure of the generated insights’ quality, they may not fully capture the nuances of the insights’ relevance or depth. Future work could explore additional evaluation metrics or human evaluations to provide a more comprehensive assessment of the agents’ performance.

Reliance on LLMs. We rely completely on the instruction following capabilities of the LLMs to generate insights. While this approach has shown promising results, it may not fully capture the complexity of the data or the context in which the insights are generated. Future work could explore more sophisticated models or hybrid approaches that combine LLMs with other techniques to improve the quality of the generated insights.

8.5 Conclusion

In this chapter, we introduced *AgentPoirot* that can perform exploratory data analytics from high-level goals using both open-source and closed-source models and showed that it

outperforms existing approaches like Pandas Agent. Moving forward, we consider evaluating the more diverse use cases, such as analytics for healthcare, anomaly detection for time series data, and more complex data structures like graphs. Another interesting research direction would be to design an adaptive agent that can autonomously learn new skills depending the task at hand.

8.6 Key Takeaways

To summarize the key takeaways from this chapter:

- Capabilities of LLMs in single-step systems translate well to multi-step systems for data analytics.
- LLM-based evaluation is still a feasible alternative to using the closed-source G-Eval for evaluating complex agentic pipelines.

With this chapter, we conclude our exploration of the capabilities of LLMs in various single-step and multi-step setups and establish their potential in a wide range of applications. In the next chapter, we present some prospective research directions to further our understanding of LLMs.

Chapter 9

Future Work

In this thesis, we have focused on adapting LLMs to diverse NLP tasks, from generating new utterances for data augmentation to performing end-to-end data analysis tasks as part of a multi-agent system. Through those studies, we have established the versatility of LLMs for various tasks and present key guidelines on how to successfully adapt an LLM for diverse downstream tasks.

In this chapter, we discuss the possible future directions for further research, with a key focus on improving the interpretability and trustworthiness of LLM-based systems. Our suggestions for future research include progressing to multi-modal LLMs, developing attribution frameworks to understand the decision-making process of LLMs, going beyond standard benchmarks to evaluate the “real” suitability of LLMs for real-world tasks, and exploring the potential of LLMs in more complex agentic workflows.

9.1 Progressing to Multi-Modal LLMs

We start the discussion in this chapter with the most natural progression for LLMs, which is to move towards multi-modal LLMs. LLMs have demonstrated remarkable capabilities in a wide range of NLP tasks, but we have evaluated them solely on text-based tasks. Multi-modal LLMs, which can process both text and other modalities such as images, audio, and video, can significantly enhance the performance of LLMs across a variety of real-world tasks, such as image captioning, visual question answering, video summarization, and online hate speech detection (Li et al., 2023b; Lin et al., 2024; Sahu et al., 2021; Hebert et al., 2024). Multi-modal LLMs can leverage the complementary information

provided by different modalities to improve the quality of their predictions and generate more informative and contextually relevant outputs. For example, multi-modal LLMs can be used to generate image captions, answer questions about images, and perform other tasks that require the integration of text and visual information.

The first generation of multi-modal LLMs includes VisualBERT (Li et al., 2020), a multi-modal LLM that combines the power of BERT with the visual understanding of ResNet to perform a wide range of tasks, including image captioning, visual question answering, and image-text matching. Similarly, Lu et al. (2019) proposed ViLBERT, a multi-modal LLM that can process both text and images to perform tasks such as visual question answering and visual grounding. These developments highlight the potential of multi-modal LLMs to revolutionize the field of NLP by enabling agents to process and generate information across multiple modalities. But the most recent generation of multimodal LLMs, such as LLaMA-3.2, Claude-3.5, and GPT-4 have been trained on an unprecedented amount of *multi-modal* data (AI, 2024; Anthropic, 2024; OpenAI, 2023b). This has opened a new foray of possibilities for multi-modal LLMs to perform complex tasks that require the integration of text, images, audio, and video data. In the context of agentic workflows, integrating multi-modal LLMs can significantly enhance the capabilities of agents like AgentPoirot. For example, an agent equipped with visual understanding can interpret graphical data, allowing it to analyze and generate insights from images, charts, and diagrams. This ability can mitigate the limitations posed by the context window size by enabling the agent to process visual summaries instead of lengthy textual descriptions. Or, we can imagine more complex agentic workflows that involve processing multiple modalities of data, such as browsing the internet (Zhou et al., 2023).

Despite the significant advancements in multi-modal LLMs, there are still several challenges that need to be addressed to fully harness their potential. One primary challenge is the effective alignment of information across different modalities. Ensuring that textual, visual, and any other modalities' representations are coherently integrated is crucial for tasks such as image captioning and visual question answering. Models like CLIP (Radford et al., 2021) have made strides in this area by learning joint embeddings of images and text, but there is room for improvement in handling more complex and abstract concepts across modalities. Another challenge lies in the resource-intensive nature of training large-scale multi-modal models. The necessity for vast amounts of annotated multi-modal data and substantial computational resources can be prohibitive. Research into more efficient training methods, such as transfer learning and few-shot learning for multi-modal tasks, could alleviate these issues and make the technology more accessible.

Future research directions may include exploring the incorporation of additional modalities such as haptic feedback, leading to more immersive and interactive AI systems. Ad-

ditionally, addressing ethical considerations like bias and fairness in multi-modal data is crucial as models become more integrated into everyday applications.

Overall, progressing towards multi-modal LLMs holds great promise for developing AI systems that interact with the world in a more human-like and comprehensive manner, and we hope to see more research in this area in the future.

9.2 Attribution in LLMs

We now turn our focus to the interpretability and trustworthiness of LLMs, which are critical for their successful deployment in real-world applications. Interpretability refers to the ability to understand and explain the decisions made by an AI system, while trustworthiness refers to the reliability and accountability of the system.

Throughout this thesis, we have seen that modern LLM agents demonstrate complex reasoning capabilities across diverse application domains; however, they are also black boxes whose decision-making process can be hard to understand, making it challenging to determine when the responses of the LLM can be trusted. A key component of building more trustworthy LLM agents is attribution, i.e., the ability to trace and interpret model outputs by identifying which parts of the input data or internal parameters contribute to specific outputs. Specifically, attribution in LLM agents can help **a) enhance agent accountability:** by revealing erroneous or unintended behavior in LLM agents that are being increasingly entrusted with significant decision-making roles; **b) enhance agent trustworthiness:** by providing insights into the reasoning process of LLM agents, which can enhance trust in sensitive domains such as healthcare and finance; and **c) promote ethical AI:** by helping detect and mitigate biases in agent outputs, ensuring that their actions align with ethical standards and societal values. We believe that building reliable attribution frameworks for LLMs to better understand the decision-making process of LLM agents at inference time will be a key research direction in the future.

Brief Overview of Attribution in LLMs. Recent work by [Yue et al. \(2023\)](#) focuses on automatically evaluating whether generated statements are fully supported by cited references. [Cohen-Wang et al. \(2024\)](#) present ContextCite, a method for attributing model generation to context, which can be applied on top of any existing language model to help verify generated statements, improve response quality, and detect poisoning attacks. Gradient-based techniques, such as Integrated Gradients ([Sundararajan et al., 2017](#)) and Saliency Maps ([Simonyan et al., 2014](#)), measure the contribution of each input token by

computing gradients of the output with respect to input features; advances like SmoothGrad (Smilkov et al., 2017) and DeepLIFT (Shrikumar et al., 2017) improve these methods by reducing noise and enhancing accuracy. Attention mechanisms provide insights into which parts of the input the model focuses on during generation (Bahdanau et al., 2014), though their interpretability is debated (Jain and Wallace, 2019; Wiegrefe and Pinter, 2019; Serrano and Smith, 2019). Perturbation techniques involve modifying or occluding parts of the input and observing changes in the output to infer input significance (Li et al., 2016), utilizing methods like Meaningful Perturbations (Fong and Vedaldi, 2017) and LIME (Ribeiro et al., 2016). Despite advancements, challenges such as attribution leakage (Adebayo et al., 2018), unreliability of saliency methods (Kindermans et al., 2019), and complexities in attributing outputs in large models (Ghorbani et al., 2019) persist. Surveys like Li et al. (2023a) discuss current methodologies and inherent challenges, while research by Keeling and Street (2024) examines the theoretical basis for attributing confidence to LLMs, raising concerns about the reliability of experimental assessment techniques. In contrast to the discussed gradient-based attribution methods that are challenging to scale and perturbation-based approaches that need multiple passes through the model, we propose a straightforward prompting-based attribution approach that can be applied to any LLM agent, is readily scalable and does not require multiple passes through the model.

We believe attribution is a critical aspect of building more trustworthy LLM agents, and are actively exploring the application of attribution frameworks to enhance the interpretability and trustworthiness of LLM agents in various application domains. Since most task-specific information used by LLMs is provided in the context either via few-shot prompts or via RAG-based systems, we are exploring a *reason-then-cite* attribution framework for LLM agents at inference time that first prompts an LLM agent to provide reasons for its actions and then cite excerpts from its input context that justify different outputs.

Attribution is a powerful tool that can be applied to any workflow to detect erroneous agent actions and hallucinations to better enable a human user to determine when to trust the outputs of an LLM agent. For instance, we can potentially prevent jailbreaks in a prompting-based system. For instance, let’s say we have a prompt that summarizes calendar events. A bad actor might try to manipulate the agent to, say, forward private information to a specific email by adding some irrelevant text to the description of a calendar event. In such cases, an attribution framework can help the user prevent the agent from taking action by showing them the parts of the input context that led to the agent’s suggested action (in this case, the manipulative text in a calendar invite). Similarly, attribution can be used to detect and prevent hallucinations in LLM agents, where the agent generates text that is not supported by the input context. The following text discusses one such use case that innately requires citing and reasoning – literature

review agents.

Literature Review Agents as Testbed We test our attribution framework on literature review agents, a task that requires finding, citing, and contextualizing relevant prior work, which is a scientific writing requirement. There is a growing interest in using LLMs as writing assistants for conducting literature reviews of scientific papers. For example, Agarwal et al. (2024) introduced LitLLM, a toolkit that leverages Retrieval Augmented Generation (RAG) and specialized prompting techniques to automate literature reviews, addressing challenges like hallucinations and outdated information in LLMs. Similarly, Skarlinski et al. (2024) demonstrate that language agents can achieve superhuman synthesis of scientific knowledge, matching or exceeding human experts in tasks such as information retrieval, summarization, and contradiction detection. Zheng et al. (2024) present OpenResearcher, an innovative platform that accelerates scientific research by integrating LLMs with up-to-date, domain-specific knowledge using RAG principles and various tools for query understanding, information retrieval, and answer generation. Furthermore, Lu et al. (2024) propose the AI Scientist, a comprehensive framework for fully automated scientific discovery, enabling LLMs to independently perform research tasks, including idea generation, experiment execution, and scientific writing. These developments highlight the transformative potential of AI agents in automating and enhancing the scientific research process.

As the subtasks of assisting a human user with writing a literature review become more automated, the type of architecture that one might use to create a literature review becomes more suitable for an agentic formulation. Some recent work has even focused upon the dynamics of agents operating in information markets (Weiss et al.), but we focus our analysis and experiments here on agents that track both the provenance of articles that have been identified for inclusion in a literature review and the rationale behind why articles are included.

Improving Literature Review Agents with Attribution. In this section, we discuss why attribution is helpful in literature review agents. We propose a *reason-then-cite* attribution framework, where we prompt an LLM agent to **first** provide a rationale for its action and **then** identify specific parts from the input context that explains the provided rationale. By prompting the agent to provide a rationale before identifying supporting evidence from the context, we aim to bridge the gap between the agent’s output and its underlying reasoning process.

Specifically, we explore *debate-style attribution*, where we ask the agent to provide arguments *for* and *against* taking the action (that it took) and extract sentences and then ask it to generate a score between 0-100 based on the arguments that denotes the probability of the agent taking that action. Importantly, we instruct the LLM to provide excerpts from its input that support the arguments, and we re-prompt the LLM if the extracted sentences are hallucinated (i.e., not part of the model input). The generic attribution method can be applied to any step of an LLM agent by appropriately describing the action.

We demonstrate the application of our attribution framework on an agentic variant of LitLLM (Agarwal et al., 2024), based on an LLM for automated literature review generation. LitLLM takes a query abstract as input and then employs RAG to obtain a list of candidate papers. After retrieval, it ranks the candidates according to their relevance to the query abstract and considers top- k candidates when writing the literature review. We apply our attribution method to the ranking step of a larger agentic workflow. Specifically, we pass the query abstract and the abstract of candidate papers one at a time and prompt the agent to generate arguments for and against including a candidate paper in the literature review and a final score between 0-100 that is based on the generated arguments and is proportional to the probability of the candidate paper being included in the literature review. We use this score to rank the candidate papers and select the top- k papers for the literature review.

We now discuss the findings of our small-scale experiments with the agentic LitLLM agent with the *debate-style* attribution framework. We use LLaMA-3.1-70b-Instruct as the base model for the agentic LitLLM agent. Table 9.1 shows outputs from the ranking step of agentic LitLLM with *debate-style* attribution, where the agent was tasked with ranking k candidate papers on whether or not to include them in the literature review of a given query paper (for brevity, we only show two candidate papers in the Table). The highlighted sentences in the reasonings indicate the parts that justify the agent’s provided ranking for the two candidate papers. We note that the reasoning aptly justifies the generated ranking. Additionally, we also perform an automated check to confirm that none of the extracted sentences in Table 9.1 are hallucinated and are indeed present in the respective candidate papers. The consistency in the agent’s output and the attribution indicates higher reliability of the agent output in this scenario.

We report some preliminary quantitative results in Figure 9.1, where we plot the *normalized recall* and *precision* curves for agentic LitLLM with the two attribution variants in Figure 9.1. We compute normalized recall and precision for a given query abstract as follows:

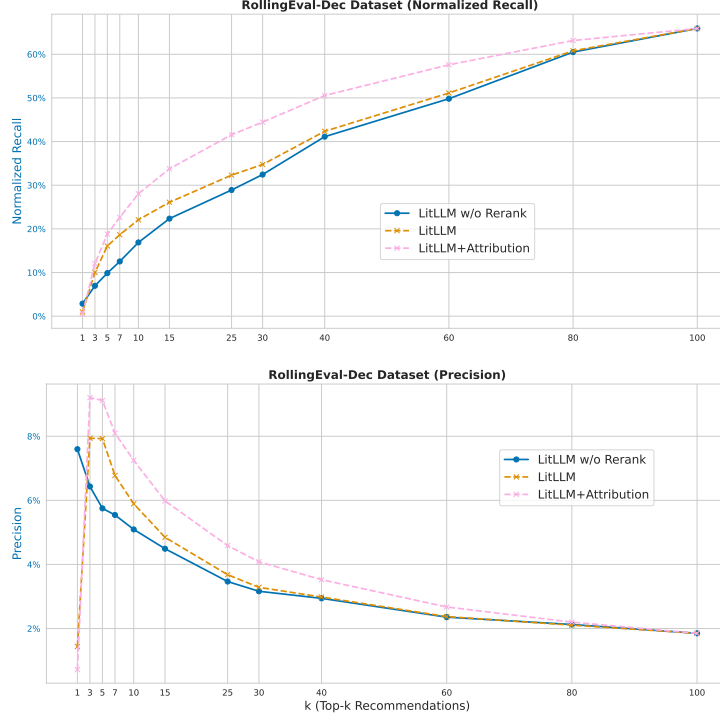


Figure 9.1: Effect of attribution on the precision and recall of the agentic LitLLM agent.

$$precision@k = \frac{n_{rel}}{N_{candidates}}; \quad normalized_recall@k = \frac{n_{rel}}{c},$$

where, $N_{candidates}$ is the total number of candidate papers, c is the number of candidate papers also present in the ground truth list of citations for the query abstract, and n_{rel} is the number of papers in first $k \in [1, 2, \dots, N_{candidates}]$ candidate papers that are present in the ground truth citations.

Although the attribution step was added to primarily understand the agent’s decision-making process, as shown in Table 9.1, we also observe improvements in the precision and recall of the agent for the retrieval step, as shown in Figure 9.1. This suggests that attribution also helps the agent in making better decisions by verbalizing a rationale for its actions.

We find the results promising and plan to further investigate the impact of attribution on the performance of other agentic workflows. For instance, we can apply the attribution framework to a data analytics agent like AgentPoirot to gain insights into the agent’s

decision-making process, or we could apply the framework to PromptMix or MixSumm to better understand the working of the models in context, or we could apply it on top of the LLM-based feature extractors in our study to model artistic preferences.

LLMs for Improving Reviews in Peer-Review Systems. In addition to the use case of improving literature review agents, we also identify another useful application: improving conference review agents. With the improving capabilities of LLMs, we can imagine a tool that can significantly reduce the time and effort researchers spend reviewing papers. A tool that, for instance, summarizes the discussion between authors and reviewers, augmented with attribution, can be a huge help for meta-reviewers and program chairs. Similarly, we can imagine a framework that first evaluates reviews based on key metrics like constructiveness, helpfulness, and thoroughness (similar to how we measure stylistic properties for lyric lines in Chapter 7), and then provides feedback to reviewers on how to improve their reviews. We believe such a tool could democratize access to high-quality research synthesis and prove to be a valuable asset for the scientific community.

9.3 Beyond Standard Benchmarks

In this section, we discuss the importance of moving beyond standard benchmarks to evaluate the “real” suitability of LLMs for real-world tasks. While standard benchmarks like GLUE, SuperGLUE, SQuAD, and MMLU provide a useful starting point for evaluating LLMs, they often do not fully capture the complexity and diversity of real-world tasks. Real-world tasks are often more nuanced, ambiguous, and context-dependent than standard benchmarks, requiring models to exhibit a deeper understanding of language and reasoning. Moreover, real-world tasks often involve multi-step reasoning, long-term memory retention, and commonsense knowledge, which are challenging for LLMs to handle effectively.

To bridge this gap between standard benchmarks and real-world tasks, we propose a more comprehensive evaluation of LLMs that reflects the complexity and diversity of real-world tasks. One approach is to develop new benchmarks that are more representative of real-world tasks, such as the recently proposed RealToxicityPrompts (Gehman et al., 2020) benchmark, which evaluates LLMs on their ability to generate non-toxic responses to toxic prompts. Another approach is to conduct more extensive evaluations on real-world datasets and tasks, such as analyzing the performance of LLMs on large-scale datasets like CommonCrawl or OpenWebText, which contain diverse and unfiltered data from the web. Additionally, we can evaluate LLMs on more complex tasks that require multi-step rea-

Table 9.1: Example use case where attribution helps understand reasoning behind an agent’s decision-making process.

Query Abstract [Paper Link] Current driven domain wall motion in curved Heavy Metal/Ferrimagnetic/Oxide multilayer strips is investigated using systematic micromagnetic simulations which account for spin-orbit coupling phenomena. Domain wall velocity and characteristic relaxation times are studied as functions of the geometry, curvature and width of the strip, at and out of the angular momentum compensation. Results show that domain walls can propagate faster and without a significant distortion in such strips in contrast to their ferromagnetic counterparts. Using an artificial system based on a straight strip with an equivalent current density distribution, we can discern its influence on the wall terminal velocity, as part of a more general geometrical influence due to the curved shape. Curved and narrow ferrimagnetic strips are promising candidates for designing high speed and fast response spintronic circuitry based on current-driven domain wall motion.
LLM Output: [1] > [2]
Reasoning for [1] [Paper Link] Arguments for including [1]: This paper focuses on the current-driven domain wall dynamics in ferrimagnetic strips, which is the same topic as the query abstract. It also uses a two-sublattice model to describe the dynamics, which is similar to the approach used in the query abstract. Extracted Sentences from Paper [1]: • <i>The current-driven domain wall dynamics along ferrimagnetic elements are here theoretically analyzed as a function of temperature by means of micromagnetic simulations and a one dimensional model.</i> • <i>Contrarily to conventional effective approaches, our model takes into account the two coupled ferromagnetic sublattices forming the ferrimagnetic element.</i> Arguments for not including [1]: This paper does not specifically mention curved strips or the effect of curvature on domain wall motion, which is a key aspect of the query abstract. Extracted Sentences from Paper [1]: • <i>Here we provide a theoretical description of DW dynamics in FiM strips based on an extended collective coordinates model (1DM).</i> • <i>Importantly, our approaches allow to infer results not achievable from effective models, and to provide insights and interesting predictions of the current-driven dynamics of DWs along FiM films.</i>
Reasoning for [2] [Paper Link] Arguments for including [2]: This paper investigates the dynamics of domain walls in ferrimagnets, specifically focusing on the angular momentum compensation point (T_A). The paper highlights the potential of ferrimagnets for high-speed spintronic devices due to the fast domain wall motion at T_A. This aligns with the query abstract’s focus on current-driven domain wall motion in curved heavy metal/ferrimagnetic/oxide multilayer strips and the potential for high-speed spintronic circuitry. Extracted Sentences from Paper [2]: • <i>Recent theories indeed predicted faster dynamics of antiferromagnetic domain walls (DWs) than ferromagnetic DWs⁴⁻⁶.</i> • <i>Here we show that fast field-driven antiferromagnetic spin dynamics is realized in ferrimagnets at the angular momentum compensation point T_A.</i> • <i>Our finding allows us to investigate the physics of antiferromagnetic spin dynamics and highlights the importance of tuning of the angular momentum compensation point of ferrimagnets, which could be a key towards ferrimagnetic spintronics.</i> Arguments for not including [2]: While the paper focuses on ferrimagnets and domain wall dynamics, it doesn’t explicitly address the impact of geometry, curvature, or width on the domain wall motion. The query abstract specifically investigates these factors in curved multilayer strips. Extracted Sentences from Paper [2]: • <i>Recently, it was reported that the DW speed boosts up significantly in antiferromagnets due to the suppression of the angular precession.</i> • <i>However, the immunity of antiferromagnets to magnetic fields yields notorious difficulties in creating, manipulating, and detecting antiferromagnetic DWs, compared to ferromagnetic ones.</i>

soning and long-term memory retention, such as question answering, dialogue generation, and document summarization.

Chapter 10

Conclusion

In this thesis, we have explored how to effectively harness the knowledge of generalist LLMs for diverse tasks. Our experiments with PromptMix, MixSumm, and PPSL suggest that generalist LLMs can be excellent data generators and data labelers for both a discriminative task like few-shot text classification as well as a generative task such as text summarization. This implies that they can mitigate a lot of human-labeling costs while producing high-quality data samples. Next, we explored how we can utilize the knowledge of LLMs to extract key linguistic and poetic features from lyrics that can allow us to model the artistic preferences of creative individuals. Lastly, our experiments with Agent-Poirot on InsightBench suggest that we can also employ LLMs for more complex tasks like exploratory data analysis as part of a bigger agentic workflow. We also teased a bit into how attribution methods can be used to understand the inner workings of LLMs, and potentially also improve their performance for a downstream task.

The possibilities of utilizing LLMs are vast, and we believe that this thesis is just a small step in exploring the potential of LLMs for more complex in-the-wild tasks. We hope that the methods and insights presented in this thesis can be a stepping stone for future research in this direction.

10.1 Implications of Our Work

The methods and insights presented in this thesis have several implications for the NLP community. **First**, we believe that data augmentation as used in PromptMix and MixSumm can be a game-changing tool for NLP researchers to generate high-quality data

samples while simultaneously performing knowledge distillation from a GPT-scale model to smaller models with a few million parameters. However, when using these methods, it is critical to ensure human moderation to avoid any biases or errors in the generated data that can be detrimental to the downstream task. **Second**, we note that while LLMs struggle with performing a subjective task like artistic preference modeling end-to-end, they can still be useful when used at a more fundamental level to measure key linguistic and stylistic features that can then be effectively used to predict subjective artistic preferences. We believe our work that provides a quantitative evidence of the strengths and shortcomings of LLMs for a subjective task can be a major stepping stone for future research on how to utilize LLMs for subjective tasks. **Third**, our insights from AgentPoirot suggest that LLMs can be used as part of a bigger agentic workflow to perform multi-step tasks like exploratory data analysis; however, current LLMs still struggle with more complex steps like generating code to answer a question based on the dataset. This suggests that there is still a gap to bridge between the capabilities of LLMs and the requirements of more complex tasks.

10.2 Recommendations for NLP Researchers

Based on the insights from this thesis, we provide the following recipe to determine whether LLMs could be helpful for their task:

- **Data Augmentation:** If working on a task that requires a lot of labeled data, consider using data augmentation methods like PromptMix and MixSumm to generate high-quality data samples; however, if the task involves generating or understanding long-form text, LLMs might not be able to aid in attaining performant models. In such cases, consider breaking down the input text into smaller chunks or develop a more sophisticated data method to improve the understanding of LLMs.
- **Feature Extraction:** If working on a subjective task like artistic preference modeling, do not expect LLMs to perform well end-to-end. Instead, consider using LLMs at a more fundamental stage, such as to extract key features from the data that can be used to model the task. As a rule of thumb, we find that the more objective the task, the better LLMs perform end-to-end, but the more subjective the task, the more LLMs struggle to perform well end-to-end and need to be used at a more fundamental stage.
- **Agentic Workflow:** If working on a multi-step task like exploratory data analysis, consider using LLMs as part of a bigger agentic workflow to perform simpler tasks

like data summarization or data filtering. As of writing this, LLMs still struggle with more complex tasks like generating code to answer a question on the dataset, which requires the LLM to generate code that is congruent to the question while also combining its understanding of the dataset.

- **Attribution Methods:** If working on a task where interpretability is critical, consider using attribution methods to understand the inner workings of LLMs and potentially improve their performance. We find that attribution methods can provide insights into how the LLMs perceive a given task and can be used to debug and improve the performance of LLMs for a downstream task; however, we still find that these insights are trustworthy only when working with short to medium-length text.
- **Human Moderation:** Regardless of the task, always ensure human moderation when using LLMs to generate data samples or perform a task to avoid any biases or errors in the generated data. There are still many unknowns about how LLMs work, and it is critical to ensure human moderation to avoid any biases or errors in the generated data that can be detrimental to the downstream task.
- **Model Selection:** For any task, while older open-source LLMs like LLaMA-2 and Vicuna do not result in satisfactory performance for real-world scenarios, the latest open-source models like LLaMA-3 and LLaMA-3.1 achieve comparable performance to closed-source models like GPT-3.5 and GPT-4o. We recommend using the latest open-source models for any task, as they are more cost-effective and provide more control to the end-user without any hidden operations or costs in closed-source models like GPT.

References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, pages 9505–9515, 2018.
- Vaibhav Adlakha, Shehzaad Dhuliawala, Kaheer Suleman, Harm de Vries, and Siva Reddy. Topiocqa: Open-domain conversational question answering with topic switching. *Transactions of the Association for Computational Linguistics*, 10:468–483, 2022.
- Shubham Agarwal, Issam H. Laradji, Laurent Charlin, and Christopher Pal. Litllm: A toolkit for scientific literature review, 2024. URL <https://arxiv.org/abs/2402.01788>.
- Meta AI. Llama 3.2 connects to 2024 vision for edge and mobile devices, 2024. URL <https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/>. Accessed: 2024-04-27.
- Neel Alex, Eli Lifland, Lewis Tunstall, Abhishek Thakur, Pegah Maham, C. Jess Riedel, Emmie Hine, Carolyn Ashurst, Paul Sedille, Alexis Carlier, Michael Noetel, and Andreas Stuhlmüller. RAFT: A real-world few-shot text classification benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL <https://openreview.net/forum?id=bgWHz41FMB7>.
- Niek Althuisen. Revisiting berlyne’s inverted u-shape relationship between complexity and liking: The role of effort, arousal, and status in the appreciation of product design aesthetics. *Psychology & Marketing*, 38(3):481–503, 2021. doi: <https://doi.org/10.1002/mar.21449>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/mar.21449>.

- Teresa M Amabile. Social psychology of creativity: A consensual assessment technique. *Journal of personality and social psychology*, 43(5):997, 1982.
- Massih-Reza Amini and Patrick Gallinari. Automatic text summarization using unsupervised and semi-supervised learning. In *Principles of Data Mining and Knowledge Discovery: 5th European Conference, PKDD 2001, Freiburg, Germany, September 3–5, 2001 Proceedings 5*, pages 16–28. Springer, 2001.
- Ateret Anaby-Tavor, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. Do Not Have Enough Data? Deep Learning to the Rescue! volume 34, pages 7383–7390, April 2020. doi: 10.1609/aaai.v34i05.6233. URL <https://aaai.org/ojs/index.php/AAAI/article/view/6233>.
- AI Anthropic. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 1, 2024.
- John Baer. How divergent thinking tests mislead us: Are the torrance tests still relevant in the 21st century? the division 10 debate. *Psychology of Aesthetics, Creativity, and the Arts*, 5(4):309–313, November 2011. doi: 10.1037/a0025210.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL <https://api.semanticscholar.org/CorpusID:11212020>.
- Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.
- Regina Barzilay and Kathleen R McKeown. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328, 2005.
- Randy Bean. Why becoming a data-driven organization is so hard. *Harvard Business Review*, feb 2022.
- Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.
- Masha Belyi, Robert Friel, Shuai Shao, and Atindriyo Sanyal. Luna: An evaluation foundation model to catch language model hallucinations with high accuracy and low cost. *arXiv preprint arXiv:2406.00975*, 2024.

- D. E. Berlyne. Studies in the new experimental aesthetics: Steps toward an objective psychology of aesthetic appreciation. *Journal of Aesthetics and Art Criticism*, 34(1): 86–87, 1975. doi: 10.2307/428656.
- Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. Inpars: Data augmentation for information retrieval using large language models. *arXiv preprint arXiv:2202.05144*, 2022.
- Gerlof Bouma. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, 30:31–40, 2009.
- Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.
- Jonathan Bragg, Arman Cohan, Kyle Lo, and Iz Beltagy. Flex: Unifying evaluation for few-shot nlp. *Advances in Neural Information Processing Systems*, 34:15787–15800, 2021.
- Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.
- Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a” siamese” time delay neural network. *Advances in neural information processing systems*, 6, 1993.
- Cleanth Brooks. Warren, robert penn. *Understanding poetry*, 4, 1976.
- Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 1877–1901, 2020.
- Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior research methods*, 46:904–911, 2014.
- Cristian Bucila, Rich Caruana, Alexandru Niculescu-Mizil, et al. Model compression. In *Kdd*, volume 6, pages 1150402–1150464, 2006.
- W.S. Burroughs. *The Job: Interviews with William S. Burroughs*. Penguin Modern Classics. Penguin Books Limited, 2012. ISBN 9780141903583. URL <https://books.google.ca/books?id=DivGjW-qEv8C>.

- Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 38–45, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.nlp4convai-1.5. URL <https://aclanthology.org/2020.nlp4convai-1.5>.
- Ernst Cassirer. *Language and myth*, volume 51. Courier Corporation, 1946.
- Tuhin Chakrabarty, Philippe Laban, Divyansh Agarwal, Smaranda Muresan, and Chien-Sheng Wu. Art or artifice? large language models and the false promise of creativity. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–34, 2024.
- Louis C Charland. The heat of emotion: Valence and the demarcation problem. *Journal of consciousness studies*, 12(8-9):82–102, 2005.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021a.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, David W. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, S. Arun Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374, 2021b.
- Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023a.
- Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

- Zhutian Chen, Chenyang Zhang, Qianwen Wang, Jakob Troidl, Simon Warchol, Johanna Beyer, Nils Gehlenborg, and Hanspeter Pfister. Beyond generating code: Evaluating GPT on a data visualization course, 2023b.
- Liyang Cheng, Xingxuan Li, and Lidong Bing. Is GPT-4 a good data analyst? In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL <https://openreview.net/forum?id=PxEhoPiBB0>.
- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL <https://lmsys.org/blog/2023-03-30-vicuna/>.
- Bharath Chintagunta, Namit Katariya, Xavier Amatriain, and Anitha Kannan. Medically aware GPT-3 as a data generator for medical dialogue summarization. In Chaitanya Shivade, Rashmi Gangadharaiah, Spandana Gella, Sandeep Konam, Shaoqing Yuan, Yi Zhang, Parminder Bhatia, and Byron Wallace, editors, *Proceedings of the Second Workshop on Natural Language Processing for Medical Conversations*, pages 66–76, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.nlpmc-1.9. URL <https://aclanthology.org/2021.nlpmc-1.9>.
- Kyunghyun Cho. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.
- Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2097. URL <https://aclanthology.org/N18-2097>.
- Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Madry. Contextcite: Attributing model generation to context, 2024. URL <https://arxiv.org/abs/2409.00729>.
- Eric Colson. What ai-driven decision making looks like. *Harvard Business Review*, jul 2019.
- Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 1995.

- Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. Snips Voice Platform: an embedded Spoken Language Understanding system for private-by-design voice interfaces. *arXiv:1805.10190 [cs]*, December 2018. URL <http://arxiv.org/abs/1805.10190>. arXiv: 1805.10190.
- Charles R. Cross. *Heavier Than Heaven: A Biography of Kurt Cobain*. Hyperion, New York, 2001.
- Haixing Dai, Zhengliang Liu, Wenxiong Liao, Xiaoke Huang, Yihan Cao, Zihao Wu, Lin Zhao, Shaochen Xu, Wei Liu, Ninghao Liu, et al. Auggpt: Leveraging chatgpt for text data augmentation. *arXiv preprint arXiv:2302.13007*, 2023.
- Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. Goemotions: A dataset of fine-grained emotions. *arXiv preprint arXiv:2005.00547*, 2020.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Tamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019a. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL <https://aclanthology.org/N19-1423>.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2019b. URL <https://arxiv.org/abs/1810.04805>.
- Bosheng Ding, Chengwei Qin, Ruochen Zhao, Tianze Luo, Xinze Li, Guizhen Chen, Wenhan Xia, Junjie Hu, Anh Tuan Luu, and Shafiq Joty. Data augmentation using llms: Data perspectives, learning paradigms and challenges. *arXiv preprint arXiv:2403.02990*, 2024.
- Xingping Dong and Jianbing Shen. Triplet loss in siamese network for object tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 459–474, 2018.
- Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. Gsum: A general framework for guided neural abstractive summarization. In *Proceedings of the*

- 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4830–4842, 2021.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori Hashimoto. AlpacaFarm: A simulation framework for methods that learn from human feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=4hturzLcKX>.
- Sarah M. Dunnigan. *7 Poetic Imagery*, pages 67–77. Edinburgh University Press, Edinburgh, 2014. ISBN 9780748691333. doi: doi:10.1515/9780748691333-008. URL <https://doi.org/10.1515/9780748691333-008>.
- Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jian, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D Hwang, et al. Faith and fate: Limits of transformers on compositionality. *arXiv preprint arXiv:2305.18654*, 2023.
- Günes Erkan and Dragomir R. Radev. Lexrank: graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479, dec 2004. ISSN 1076-9757.
- Alexander R Fabbri, Simeng Han, Haoyuan Li, Haoran Li, Marjan Ghazvininejad, Shafiq Joty, Dragomir Radev, and Yashar Mehdad. Improving zero and few-shot abstractive summarization with intermediate fine-tuning and data augmentation. *arXiv preprint arXiv:2010.12836*, 2020.
- Guy Feigenblat, Chulaka Gunasekara, Benjamin Sznajder, Sachindra Joshi, David Konopnicki, and Ranit Aharonov. TWEETSUMM - a dialog summarization dataset for customer service. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 245–260, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.24. URL <https://aclanthology.org/2021.findings-emnlp.24>.
- Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. A survey of data augmentation approaches for nlp. *arXiv preprint arXiv:2105.03075*, 2021.

- Ruth C. Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3429–3437, 2017.
- Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. GPTScore: Evaluate as you desire. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6556–6576, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.365. URL <https://aclanthology.org/2024.naacl-long.365>.
- Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020.
- Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. Real-ToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.301. URL <https://aclanthology.org/2020.findings-emnlp.301>.
- Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3681–3688, 2019.
- Benjamin P. Gold, Marcus T. Pearce, Ernest Mas-Herrero, Alain Dagher, and Robert J. Zatorre. Predictability and uncertainty in the pleasure of music: A reward for learning? *Journal of Neuroscience*, 39(47):9397–9409, 2019. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.0428-19.2019. URL <https://www.jneurosci.org/content/39/47/9397>.
- Chen Gong, Dacheng Tao, Stephen J Maybank, Wei Liu, Guoliang Kang, and Jie Yang. Multi-modal curriculum learning for semi-supervised image classification. *IEEE Transactions on Image Processing*, 25(7):3249–3260, 2016.
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.
- Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

- Tanya Goyal, Junyi Jessy Li, and Greg Durrett. News summarization and evaluation in the era of gpt-3. *arXiv preprint arXiv:2209.12356*, 2022.
- Significant Gravitas. AutoGPT, 2023. URL <https://github.com/Significant-Gravitas/Auto-GPT>.
- Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Multimodal semi-supervised learning for image classification. In *2010 IEEE Computer society conference on computer vision and pattern recognition*, pages 902–909. IEEE, 2010.
- Yue Guo, Yi Yang, and Ahmed Abbasi. Auto-debias: Debiasing masked language models with automated biased prompts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1012–1023, 2022.
- Helia Hashemi, Jason Eisner, Corby Rosset, Benjamin Van Durme, and Chris Kedzie. Llm-rubric: A multidimensional, calibrated approach to automated evaluation of natural language texts. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13806–13834, 2024.
- Tatsunori B. Hashimoto, Hugh Zhang, and Percy Liang. Unifying human and statistical evaluation for natural language generation. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1689–1701, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1169. URL <https://aclanthology.org/N19-1169>.
- Liam Hebert, Gaurav Sahu, Yuxuan Guo, Nanda Kishore Sreenivas, Lukasz Golab, and Robin Cohen. Multi-modal discussion transformer: Integrating text, images and graph transformers to detect hate speech on social media. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 22096–22104, 2024.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.
- Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. URL <http://arxiv.org/abs/1503.02531>.

- Namgyu Ho, Laura Schmid, and Se-Young Yun. Large language models are reasoning teachers. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14852–14882, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.830. URL <https://aclanthology.org/2023.acl-long.830>.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Li Zhang, Lingyao Zhang, Min Yang, Mingchen Zhuge, Taicheng Guo, Tuo Zhou, Wei Tao, Wenyi Wang, Xiangru Tang, Xiangtao Lu, Xiaowu Zheng, Xinbing Liang, Yaying Fei, Yuheng Cheng, Zongze Xu, and Chenglin Wu. Data interpreter: An llm agent for data science, 2024.
- Xueyu Hu, Ziyu Zhao, Shuang Wei, Ziwei Chai, Qianli Ma, Guoyin Wang, Xuwu Wang, Jing Su, Jingjing Xu, Ming Zhu, Yao Cheng, Jianbo Yuan, Jiwei Li, Kun Kuang, Yang Yang, Hongxia Yang, and Fei Wu. Infiagent-dabench: Evaluating agents on data analysis tasks. In *The Twelfth International Conference on Machine Learning (ICML)*, 2024.
- Sarthak Jain and Byron C. Wallace. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 3543–3556. Association for Computational Linguistics, 2019.
- Kate M Johnson-Grey, Reihane Boghrati, Cheryl J Waksalak, and Morteza Dehghani. Measuring abstract mind-sets through syntax: Automating the linguistic category model. *Social Psychological and Personality Science*, 11(2):217–225, 2020.
- Dylan Jones. *David Bowie: A Life*. Crown Archetype, New York, first edition edition, 2017.
- Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, 2nd edition, 2000. URL <https://www.speech.cs.cmu.edu/cgi-bin/cmudict>.
- Justine Kao and Dan Jurafsky. A computational analysis of style, affect, and imagery in contemporary poetry. In *Proceedings of the NAACL-HLT 2012 workshop on computational linguistics for literature*, pages 8–17, 2012.

- Geoff Keeling and Winnie Street. On the attribution of confidence to large language models, 2024. URL <https://arxiv.org/abs/2407.08388>.
- Kashif Khan, Gaurav Sahu, Vikash Balasubramanian, Lili Mou, and Olga Vechtomova. Adversarial learning on the latent space for diverse dialog generation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5026–5034, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.441. URL <https://www.aclweb.org/anthology/2020.coling-main.441>.
- Tushar Khot, Ashish Sabharwal, and Peter Clark. Scitail: A textual entailment dataset from science question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- Yekyung Kim, Seohyeong Jeong, and Kyunghyun Cho. Linda: Unsupervised learning to interpolate in natural language processing. *arXiv preprint arXiv:2112.13969*, 2021.
- Yoon Kim. Convolutional neural networks for sentence classification proceedings of the 2014 conference on empirical methods in natural language processing, emnlp 2014, october 25-29, 2014, doha, qatar, a meeting of sigdat, a special interest group of the acl. *Association for Computational Linguistics, Doha, Qatar*, 2014.
- Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T. Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280, 2019.
- Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. URL <https://arxiv.org/abs/1312.6114>.
- Mahnaz Koupaee and William Yang Wang. Wikihow: A large scale text summarization dataset. *arXiv preprint arXiv:1810.09305*, 2018.
- Varun Kumar, Hadrien Glaude, Cyprien de Lichy, and Wlliam Campbell. A Closer Look At Feature Space Data Augmentation For Few-Shot Intent Classification. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 1–10, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-6101. URL <https://www.aclweb.org/anthology/D19-6101>.

- Varun Kumar, Ashutosh Choudhary, and Eunah Cho. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pages 18–26, 2020.
- Farshad Kyoomarsi, Hamid Khosravi, Esfandiar Eslami, Pooya Khosravayan Dehkordy, and Asghar Tajoddin. Optimizing text summarization based on fuzzy logic. In *Seventh IEEE/ACIS International Conference on Computer and Information Science (icis 2008)*, pages 347–352. IEEE, 2008.
- Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Scott Wen tau Yih, Daniel Fried, Sida Wang, and Tao Yu. DS-1000: A natural and reliable benchmark for data science code generation. *ArXiv*, abs/2211.11501, 2022.
- LangChain. Pandas dataframe. <https://python.langchain.com/v0.2/docs/integrations/toolkits/pandas/>, 2024. Accessed: 2024-06-03.
- Issam Laradji, Perouz Taslakian, Sai Rajeswar, Valentina Zantedeschi, Alexandre Lacoste, Nicolas Chapados, David Vazquez, Christopher Pal, and Alexandre Drouin. Capture the flag: Uncovering data insights with large language models. *arXiv preprint arXiv:2312.13876*, 2023.
- Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. An Evaluation Dataset for Intent Classification and Out-of-Scope Prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1131. URL <https://www.aclweb.org/anthology/D19-1131>.
- Kenton Lee, Kelvin Guu, Luheng He, Tim Dozat, and Hyung Won Chung. Neural Data Augmentation via Example Extrapolation. *arXiv:2102.01335 [cs]*, February 2021. URL <http://arxiv.org/abs/2102.01335>. arXiv: 2102.01335.
- David D Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *European conference on machine learning*, pages 4–15. Springer, 1998.
- 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, 2020a.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020b.
- Dongfang Li, Zetian Sun, Xinshuo Hu, Zhenyu Liu, Ziyang Chen, Baotian Hu, Aiguo Wu, and Min Zhang. A survey of large language models attribution, 2023a. URL <https://arxiv.org/abs/2311.03731>.
- Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in nlp. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 681–691. Association for Computational Linguistics, 2016.
- Lei Li, Yuwei Yin, Shicheng Li, Liang Chen, Peiyi Wang, Shuhuai Ren, Mukai Li, Yazheng Yang, Jingjing Xu, Xu Sun, et al. M3it: A large-scale dataset towards multi-modal multilingual instruction tuning. *arXiv preprint arXiv:2306.04387*, 2023b.
- Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. What does BERT with vision look at? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5265–5275, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.469. URL <https://aclanthology.org/2020.acl-main.469>.
- Wenjie Li, Mingli Wu, Qin Lu, Wei Xu, and Chunfa Yuan. Extractive summarization using inter-and intra-event relevance. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 369–376, 2006.
- Kevin J Liang, Weituo Hao, Dinghan Shen, Yufan Zhou, Weizhu Chen, Changyou Chen, and Lawrence Carin. Mixkd: Towards efficient distillation of large-scale language models. In *International Conference on Learning Representations*, 2021.
- Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL <https://aclanthology.org/W04-1013>.

- Ji Lin, Hongxu Yin, Wei Ping, Pavlo Molchanov, Mohammad Shoeybi, and Song Han. Vila: On pre-training for visual language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26689–26699, 2024.
- Yen-Ting Lin, Alexandros Papangelis, Seokhwan Kim, Sungjin Lee, Devamanyu Hazarika, Mahdi Namazifar, Di Jin, Yang Liu, and Dilek Hakkani-Tur. Selective in-context data augmentation for intent detection using pointwise v-information. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1455–1468, 2023.
- Hanmeng Liu, Zhiyang Teng, Leyang Cui, Chaoli Zhang, Qiji Zhou, and Yue Zhang. Logi-CoT: Logical chain-of-thought instruction tuning. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2908–2921, Singapore, December 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.191. URL <https://aclanthology.org/2023.findings-emnlp.191>.
- Quande Liu, Lequan Yu, Luyang Luo, Qi Dou, and Pheng Ann Heng. Semi-supervised medical image classification with relation-driven self-ensembling model. *IEEE transactions on medical imaging*, 39(11):3429–3440, 2020.
- Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1387. URL <https://aclanthology.org/D19-1387>.
- Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: NLG evaluation using gpt-4 with better human alignment. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore, December 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.153. URL <https://aclanthology.org/2023.emnlp-main.153>.
- Yinhan Liu. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.
- Yixin Liu, Alexander Fabbri, Jiawen Chen, Yilun Zhao, Simeng Han, Shafiq Joty, Pengfei Liu, Dragomir Radev, Chien-Sheng Wu, and Arman Cohan. Benchmarking generation and evaluation capabilities of large language models for instruction controllable

- summarization. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4481–4501, Mexico City, Mexico, June 2024a. Association for Computational Linguistics. URL <https://aclanthology.org/2024.findings-naacl.280>.
- Yixin Liu, Kejian Shi, Katherine He, Longtian Ye, Alexander Fabbri, Pengfei Liu, Dragomir Radev, and Arman Cohan. On learning to summarize with large language models as references. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8647–8664, Mexico City, Mexico, June 2024b. Association for Computational Linguistics. URL <https://aclanthology.org/2024.naacl-long.478>.
- Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- Matthew W. Lowder, Wonil Choi, Fernanda Ferreira, and John M. Henderson. Lexical predictability during natural reading: Effects of surprisal and entropy reduction. *Cognitive Science*, 42(S4):1166–1183, 2018. doi: <https://doi.org/10.1111/cogs.12597>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1111/cogs.12597>.
- Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery, 2024. URL <https://arxiv.org/abs/2408.06292>.
- Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. ViLBERT: pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, number 2, pages 13–23. Curran Associates Inc., Red Hook, NY, USA, December 2019.
- Pingchuan Ma, Rui Ding, Shuai Wang, Shi Han, and Dongmei Zhang. Demonstration of InsightPilot: An LLM-empowered automated data exploration system. *CoRR*, abs/2304.00477, 2023.
- Andrew McAfee and Erik Brynjolfsson. Big data: The management revolution. *Harvard Business Review*, oct 2012. From the Magazine (October 2012).
- Nicholas Meade, Elinor Poole-Dayana, and Siva Reddy. An empirical survey of the effectiveness of debiasing techniques for pre-trained language models. *arXiv preprint arXiv:2110.08527*, 2021.

- Yu Meng, Jiaxin Huang, Yu Zhang, and Jiawei Han. Generating training data with language models: Towards zero-shot language understanding. *arXiv preprint arXiv:2202.04538*, 2022.
- Yu Meng, Martin Michalski, Jiaxin Huang, Yu Zhang, Tarek Abdelzaher, and Jiawei Han. Tuning language models as training data generators for augmentation-enhanced few-shot learning. In *International Conference on Machine Learning*, pages 24457–24477. PMLR, 2023.
- Sewon Min, Jordan Boyd-Graber, Chris Alberti, Danqi Chen, Eunsol Choi, Michael Collins, Kelvin Guu, Hannaneh Hajishirzi, Kenton Lee, Jennimaria Palomaki, et al. Neurips 2020 efficientqa competition: Systems, analyses and lessons learned. In *NeurIPS 2020 Competition and Demonstration Track*, pages 86–111. PMLR, 2021.
- Nishant Mishra, Gaurav Sahu, Iacer Calixto, Ameen Abu-Hanna, and Issam Laradji. Llm aided semi-supervision for efficient extractive dialog summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10002–10009, 2023.
- Takeru Miyato, Andrew M Dai, and Ian Goodfellow. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*, 2016.
- Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gülçehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, 2016.
- John Ashworth Nelder and Robert WM Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 135(3):370–384, 1972.
- Joel Larocca Neto, Alex A Freitas, and Celso AA Kaestner. Automatic text summarization using a machine learning approach. In *Advances in Artificial Intelligence: 16th Brazilian Symposium on Artificial Intelligence, SBIA 2002 Porto de Galinhas/Recife, Brazil, November 11–14, 2002 Proceedings 16*, pages 205–215. Springer, 2002.
- NVIDIA. What is agentic ai? <https://blogs.nvidia.com/blog/what-is-agentic-ai/>, 2023. Accessed: September 1, 2024.
- OpenAI. Introducing ChatGPT. <https://openai.com/index/chatgpt/>, 2022.
- OpenAI. GPT-4 technical report. *ArXiv*, abs/2303.08774, 2023a.

- OpenAI. Code interpreter. <https://platform.openai.com/docs/assistants/tools/code-interpreter>, 2024. Accessed: 2024-06-03.
- R OpenAI. Gpt-4 technical report. *arXiv*, pages 2303–08774, 2023b.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.
- C. Palahniuk. *Consider This: Moments in My Writing Life after Which Everything Was Different*. Grand Central Publishing, 2020. ISBN 9781538717967. URL <https://books.google.ca/books?id=p3ibDwAAQBAJ>.
- Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *arXiv preprint cs/0409058*, 2004.
- 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, 2002.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.
- Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*, 2023.
- Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep Contextualized Word Representations. In

- Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics, NAACL-HLT 2018*, February 2018. URL <http://arxiv.org/abs/1802.05365>. arXiv: 1802.05365.
- Jonathan Pilault, Raymond Li, Sandeep Subramanian, and Christopher Pal. On extractive and abstractive neural document summarization with transformer language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9308–9319, 2020.
- Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. Mauve: measuring the gap between neural text and human text using divergence frontiers. In *Proceedings of the 35th International Conference on Neural Information Processing Systems, NIPS '21*, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9781713845393.
- Ezra Pound. A few don'ts by an imagiste. *Poetry*, 1(6):200–206, 1913.
- Daniel Preotiu-Pietro, Mihaela Gaman, and Nikolaos Aletras. Automatically identifying complaints in social media. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5008–5019, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1495. URL <https://aclanthology.org/P19-1495>.
- Dongqi Pu and Vera Demberg. Chatgpt vs human-authored text: Insights into controllable text summarization and sentence style transfer. *arXiv preprint arXiv:2306.07799*, 2023.
- PyTorch Seq2Seq Tutorials. Translation with a sequence to sequence network and attention. https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html, 2024. Accessed: October 31, 2024.
- Chen Qian, Xin Cong, Wei Liu, Cheng Yang, Weize Chen, Yusheng Su, Yufan Dang, Jiahao Li, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. Communicative agents for software development, 2023.
- Chen Qian, Jiahao Li, Yufan Dang, Wei Liu, YiFei Wang, Zihao Xie, Weize Chen, Cheng Yang, Yingli Zhang, Zhiyuan Liu, and Maosong Sun. Iterative experience refinement of software-developing agents, 2024.
- Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *Science China technological sciences*, 63(10):1872–1897, 2020.

- J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1:81–106, 1986.
- Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
- Dragomir Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Çelebi, Stanko Dimitrov, Elliott Franco Drabek, Ali Hakim, Wai Lam, Danyu Liu, et al. Mead: a platform for multidocument multilingual text summarization. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC’04)*, 2004.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- 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.
- 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. *arXiv preprint arXiv:1910.10683*, 2019. URL <https://arxiv.org/abs/1910.10683>.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016.
- Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. Collecting image annotations using amazon’s mechanical turk. In *Proceedings of the NAACL HLT 2010 workshop on creating speech and language data with Amazon’s Mechanical Turk*, pages 139–147, 2010.
- 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 (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL <https://aclanthology.org/D19-1410>.

- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.
- Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “why should i trust you?”: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016.
- Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.437. URL <https://aclanthology.org/2020.emnlp-main.437>.
- Rick Rubin. *The creative act: a way of being*. Penguin, 2023.
- David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.
- Gaurav Sahu and Issam H Laradji. Mixsumm: Topic-based data augmentation using llms for low-resource extractive text summarization. *arXiv preprint arXiv:2407.07341*, 2024.
- Gaurav Sahu and Olga Vechtomova. Adaptive fusion techniques for multimodal data. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3156–3166, 2021.
- Gaurav Sahu and Olga Vechtomova. Computational modeling of artistic inspiration: A framework for predicting aesthetic preferences in lyrical lines using linguistic and stylistic features. *arXiv preprint arXiv:2410.02881*, 2024.
- Gaurav Sahu, Robin Cohen, and Olga Vechtomova. Towards a multi-agent system for online hate speech detection. *Second Workshop on Autonomous Agents for Social Good (AASG), AAMAS, 2021*, 2021.
- Gaurav Sahu, Pau Rodriguez, Issam Laradji, Parmida Atighehchian, David Vazquez, and Dzmitry Bahdanau. Data augmentation for intent classification with off-the-shelf large language models. In Bing Liu, Alexandros Papangelis, Stefan Ultes, Abhinav Rastogi, Yun-Nung Chen, Georgios Spithourakis, Elnaz Nouri, and Weiyan Shi, editors, *Proceedings of the 4th Workshop on NLP for Conversational AI*, pages 47–57, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.nlp4convai-1.5. URL <https://aclanthology.org/2022.nlp4convai-1.5>.

- Gaurav Sahu, Olga Vechtomova, Dzmitry Bahdanau, and Issam Laradji. PromptMix: A class boundary augmentation method for large language model distillation. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5316–5327, Singapore, December 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.323. URL <https://aclanthology.org/2023.emnlp-main.323>.
- Gaurav Sahu, Olga Vechtomova, and Issam H Laradji. Prompt-based pseudo-labeling strategy for sample-efficient semi-supervised extractive summarization. *arXiv preprint arXiv:2311.09559*, 2023b.
- Gaurav Sahu, Abhay Puri, Juan Rodriguez, Alexandre Drouin, Perouz Taslakian, Valentina Zantedeschi, Alexandre Lacoste, David Vazquez, Nicolas Chapados, Christopher Pal, et al. Insightbench: Evaluating business analytics agents through multi-step insight generation. *arXiv preprint arXiv:2407.06423*, 2024.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.
- Yogesh Sankarasubramaniam, Krishnan Ramanathan, and Subhankar Ghosh. Text summarization using wikipedia. *Information Processing & Management*, 50(3):443–461, 2014.
- Klaus R Scherer. On the nature and function of emotion: a component process approach. approaches to emotion, 1984.
- Timo Schick and Hinrich Schütze. It’s not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, 2021.
- Timo Schick and Hinrich Schütze. Exploiting Cloze Questions for Few Shot Text Classification and Natural Language Inference. *arXiv:2001.07676 [cs]*, January 2021. URL <http://arxiv.org/abs/2001.07676>. arXiv: 2001.07676.
- Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

- Gün R Semin and Klaus Fiedler. The cognitive functions of linguistic categories in describing persons: Social cognition and language. *Journal of personality and Social Psychology*, 54(4):558, 1988.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1009. URL <https://aclanthology.org/P16-1009>.
- Sofia Serrano and Noah A. Smith. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951. Association for Computational Linguistics, 2019.
- Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.
- Xiaoqian Shen and Mohamed Elhoseiny. Large language models as consistent story visualizers. *arXiv preprint arXiv:2312.02252*, 2023.
- Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 614–622, 2008.
- Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. Distilling multi-step reasoning capabilities of large language models into smaller models via semantic decompositions. *arXiv preprint arXiv:2212.00193*, 2022.
- Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153. PMLR, 2017.
- Kashun Shum, Shizhe Diao, and Tong Zhang. Automatic prompt augmentation and selection with chain-of-thought from labeled data. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12113–12139, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.811. URL <https://aclanthology.org/2023.findings-emnlp.811>.

- Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. 2014. URL <https://arxiv.org/abs/1312.6034>.
- Michael D. Skarlinski, Sam Cox, Jon M. Laurent, James D. Braza, Michaela Hinks, Michael J. Hammerling, Manvitha Ponnampati, Samuel G. Rodrigues, and Andrew D. White. Language agents achieve superhuman synthesis of scientific knowledge. 2024. URL <https://arxiv.org/abs/2409.13740>.
- Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: Removing noise by adding noise. 2017. URL <https://arxiv.org/abs/1706.03825>.
- Leslie N Smith. Cyclical learning rates for training neural networks. arxiv. *Preprint at https://arxiv.org/abs/1506.01186*, 2015.
- Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.
- Ladda Suanmali, Mohammed Salem Binwahlan, and Naomie Salim. Sentence features fusion for text summarization using fuzzy logic. In *2009 Ninth International Conference on Hybrid Intelligent Systems*, volume 1, pages 142–146. IEEE, 2009.
- Siqi Sun, Zhe Gan, Yuwei Fang, Yu Cheng, Shuohang Wang, and Jingjing Liu. Contrastive distillation on intermediate representations for language model compression. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 498–508, 2020.
- Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017.
- Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

- Krysta Svore, Lucy Vanderwende, and Christopher Burges. Enhancing single-document summarization by combining ranknet and third-party sources. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 448–457, 2007.
- Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.746. URL <https://aclanthology.org/2020.emnlp-main.746>.
- Ellis Paul Torrance. *Torrance Tests of Creative Thinking: Norms-Technical Manual: Verbal Tests, Forms A and B: Figural Tests, Forms A and B*. Personal Press, Incorporated, 1966.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. Efficient few-shot learning without prompts. *arXiv preprint arXiv:2209.11055*, 2022.
- Robert Vacareanu, Vlad-Andrei Negru, Vasile Suciuc, and Mihai Surdeanu. From words to numbers: Your large language model is secretly a capable regressor when given in-context examples, 2024.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- Olga Vechtomova and Gaurav Sahu. Lyricjam sonic: a generative system for real-time composition and musical improvisation. In *International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar)*, pages 292–307. Springer, 2023.
- Olga Vechtomova, Gaurav Sahu, and Dhruv Kumar. Lyricjam: A system for generating lyrics for live instrumental music. In *Proceedings of the 12th Conference on Computational Creativity*, 2021.

- Ellen M Voorhees, Dawn M Tice, et al. The trec-8 question answering track evaluation. In *TREC*, volume 1999, page 82, 1999.
- Dazhen Wan, Zheng Zhang, Qi Zhu, Lizi Liao, and Minlie Huang. A unified dialogue user simulator for few-shot data augmentation. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3788–3799, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.277. URL <https://aclanthology.org/2022.findings-emnlp.277>.
- Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. <https://github.com/kingoflolz/mesh-transformer-jax>, May 2021.
- Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv*, abs/2305.04091, 2023.
- Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents, 2024.
- Yiming Wang, Qianren Mao, Junnan Liu, Weifeng Jiang, Hongdong Zhu, and Jianxin Li. Noise-injected consistency training and entropy-constrained pseudo labeling for semi-supervised extractive summarization. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6447–6456, 2022.
- Jason Wei and Kai Zou. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1670. URL <https://aclanthology.org/D19-1670>.
- Jason Wei, Chengyu Huang, Soroush Vosoughi, Yu Cheng, and Shiqi Xu. Few-shot text classification with triplet networks, data augmentation, and curriculum learning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5493–5500, 2021.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

- Martin Weiss, Nasim Rahaman, Manuel Wuthrich, Yoshua Bengio, Li Erran Li, Bernhard Schölkopf, and Christopher Pal. Redesigning information markets in the era of language models. In *First Conference on Language Modeling*.
- Luoxuan Weng, Xingbo Wang, Junyu Lu, Yingchaojie Feng, Yihan Liu, and Wei Chen. Insightlens: Discovering and exploring insights from conversational contexts in large-language-model-powered data analysis, 2024.
- Peter West, Chandra Bhagavatula, Jack Hessel, Jena Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. Symbolic knowledge distillation: from general language models to commonsense models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4602–4625, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.341. URL <https://aclanthology.org/2022.naacl-main.341>.
- Sarah Wiegrefe and Yuval Pinter. Attention is not not explanation. In *Conference on Empirical Methods in Natural Language Processing*, pages 11–20. Association for Computational Linguistics, 2019.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, oct 2020. Association for Computational Linguistics. URL <https://www.aclweb.org/anthology/2020.emnlp-demos.6>.
- Kam-Fai Wong, Mingli Wu, and Wenjie Li. Extractive summarization using supervised and semi-supervised learning. In *Proceedings of the 22nd international conference on computational linguistics (Coling 2008)*, pages 985–992, 2008.
- Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework, 2023.
- Xing Wu, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. Conditional bert contextual augmentation. In *Computational Science–ICCS 2019: 19th International*

- Conference, Faro, Portugal, June 12–14, 2019, Proceedings, Part IV 19*, pages 84–95. Springer, 2019.
- Tianbao Xie, Fan Zhou, Zhoujun Cheng, Peng Shi, Luoxuan Weng, Yitao Liu, Toh Jing Hua, Junning Zhao, Qian Liu, Che Liu, et al. Openagents: An open platform for language agents in the wild. *arXiv preprint arXiv:2310.10634*, 2023.
- Pawel Swietojanski Xingkun Liu, Arash Eshghi and Verena Rieser. Benchmarking natural language understanding services for building conversational agents. In *Proceedings of the Tenth International Workshop on Spoken Dialogue Systems Technology (IWSDS)*, pages xxx–xxx, Ortigia, Siracusa (SR), Italy, April 2019. Springer. URL <http://www.xx.xx/xx/>.
- Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. Variational autoencoder for semi-supervised text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.
- Yinfei Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernandez Abrego, Steve Yuan, Chris Tar, Yun-hsuan Sung, Brian Strope, and Ray Kurzweil. Multilingual universal sentence encoder for semantic retrieval. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 87–94, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-demos.12. URL <https://aclanthology.org/2020.acl-demos.12>.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.
- Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyoung Park. GPT3Mix: Leveraging large-scale language models for text augmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2225–2239, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.192. URL <https://aclanthology.org/2021.findings-emnlp.192>.
- Weizhe Yuan, Graham Neubig, and Pengfei Liu. Bartscore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*, 34:27263–27277, 2021.

- Xiang Yue, Boshi Wang, Zirui Chen, Kai Zhang, Yu Su, and Huan Sun. Automatic evaluation of attribution by large language models, 2023. URL <https://arxiv.org/abs/2305.06311>.
- Chen Zhang, Luis D’Haro, Chengguang Tang, Ke Shi, Guohua Tang, and Haizhou Li. xdialog-eval: A multilingual open-domain dialogue evaluation benchmark. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5579–5601, 2023a.
- Haopeng Zhang, Xiao Liu, and Jiawei Zhang. Extractive summarization via chatgpt for faithful summary generation. *arXiv preprint arXiv:2304.04193*, 2023b.
- Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. URL <https://openreview.net/forum?id=r1Ddp1-Rb>.
- Jianguo Zhang, Trung Bui, Seunghyun Yoon, Xiang Chen, Zhiwei Liu, Congying Xia, Quan Hung Tran, Walter Chang, and Philip Yu. Few-shot intent detection via contrastive pre-training and fine-tuning. *arXiv preprint arXiv:2109.06349*, 2021.
- Lei Zhang, Yuge Zhang, Kan Ren, Dongsheng Li, and Yuqing Yang. Mlcopilot: Unleashing the power of large language models in solving machine learning tasks. *arXiv preprint arXiv:2304.14979*, 2023c.
- Shaokun Zhang, Jieyu Zhang, Jiale Liu, Linxin Song, Chi Wang, Ranjay Krishna, and Qingyun Wu. Training language model agents without modifying language models. *ICML’24*, 2024a.
- Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.
- Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics*, 12:39–57, 2024b.
- Wenqi Zhang, Yongliang Shen, Weiming Lu, and Yueting Zhuang. Data-copilot: Bridging billions of data and humans with autonomous workflow. *arXiv preprint arXiv:2306.07209*, 2023d.
- Yuxiang Zheng, Shichao Sun, Lin Qiu, Dongyu Ru, Cheng Jiayang, Xuefeng Li, Jifan Lin, Binjie Wang, Yun Luo, Renjie Pan, Yang Xu, Qingkai Min, Zizhao Zhang, Yiwen Wang, Wenjie Li, and Pengfei Liu. Openresearcher: Unleashing ai for accelerated scientific research, 2024. URL <https://arxiv.org/abs/2408.06941>.

- Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuan-Jing Huang. Extractive summarization as text matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6208, 2020.
- Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.
- Banghua Zhu, Hiteshi Sharma, Felipe Vieira Frujeri, Shi Dong, Chenguang Zhu, Michael I Jordan, and Jiantao Jiao. Fine-tuning language models with advantage-induced policy alignment. *arXiv preprint arXiv:2306.02231*, 2023.
- Yingying Zhuang, Jiecheng Song, Narayanan Sadagopan, and Anurag Beniwal. Self-supervised pre-training and semi-supervised learning for extractive dialog summarization. In *Companion Proceedings of the ACM Web Conference 2023*, pages 1069–1076, 2023.

APPENDICES

Appendix A

Worked Out Examples

A.1 HMM Worked Out Example

In this example, we will predict the probability of the word “mat” in the sequence “The cat sat on the mat” using a Hidden Markov Model (HMM). We will construct the HMM parameters based on the provided corpus and perform step-by-step calculations to illustrate the process.

Corpus:

*“The cat sat on the mat. The dog sat on the rug. The cat chased the mouse.
The dog chased the ball.”*

Objective:

Determine the probability of the word “mat” occurring in the sequence “The cat sat on the mat” using an HMM trained on the given corpus.

Step 1: Define States and Observations

- **States (Hidden States):** Parts of Speech (POS) tags
 - Determiner (DT)
 - Noun (NN)
 - Verb (VB)

– Preposition (IN)

- **Observations:** Words from the corpus

{the, cat, dog, sat, on, chased, mat, rug, mouse, ball}

Step 2: Annotate the Corpus with POS Tags

- **Sentence 1:** The/DT cat/NN sat/VB on/IN the/DT mat/NN.
- **Sentence 2:** The/DT dog/NN sat/VB on/IN the/DT rug/NN.
- **Sentence 3:** The/DT cat/NN chased/VB the/DT mouse/NN.
- **Sentence 4:** The/DT dog/NN chased/VB the/DT ball/NN.

Step 3: Calculate Initial Probabilities (π)

Since all sentences start with the Determiner (DT), we have:

$$\pi(\text{DT}) = 1.0 \quad \text{and} \quad \pi(\text{NN}) = \pi(\text{VB}) = \pi(\text{IN}) = 0$$

Step 4: Calculate Transition Probabilities (A)

Count the transitions between states from the annotated corpus.

From State	To State	Count
DT	NN	8
NN	VB	4
VB	IN	2
IN	DT	2
VB	DT	2
Total		
From DT		8
From NN		4
From VB		4
From IN		2

Compute Transition Probabilities:

$$A(\text{NN} \mid \text{DT}) = \frac{\text{Count}(\text{DT} \rightarrow \text{NN})}{\text{Total transitions from DT}} = \frac{8}{8} = 1.0$$

$$A(\text{VB} \mid \text{NN}) = \frac{4}{4} = 1.0$$

$$A(\text{IN} \mid \text{VB}) = \frac{2}{4} = 0.5, \quad A(\text{DT} \mid \text{VB}) = \frac{2}{4} = 0.5$$

$$A(\text{DT} \mid \text{IN}) = \frac{2}{2} = 1.0$$

All other transitions are not observed and are assigned a probability of zero.

Step 5: Calculate Emission Probabilities (B)

Count the emissions (word occurrences) for each state.

State DT:

$$B(\text{the} \mid \text{DT}) = \frac{8}{8} = 1.0$$

State NN:

Total emissions from NN: 8

$$B(\text{cat} \mid \text{NN}) = \frac{2}{8} = 0.25$$

$$B(\text{dog} \mid \text{NN}) = \frac{2}{8} = 0.25$$

$$B(\text{mat} \mid \text{NN}) = \frac{1}{8} = 0.125$$

$$B(\text{rug} \mid \text{NN}) = \frac{1}{8} = 0.125$$

$$B(\text{mouse} \mid \text{NN}) = \frac{1}{8} = 0.125$$

$$B(\text{ball} \mid \text{NN}) = \frac{1}{8} = 0.125$$

State VB:

Total emissions from VB: 4

$$B(\text{sat} \mid \text{VB}) = \frac{2}{4} = 0.5, \quad B(\text{chased} \mid \text{VB}) = \frac{2}{4} = 0.5$$

State IN:

$$B(\text{on} \mid \text{IN}) = \frac{2}{2} = 1.0$$

Step 6: Observed Sequence

The observed sequence is:

$$O = (\text{the}, \text{cat}, \text{sat}, \text{on}, \text{the}, \text{mat})$$

Step 7: Computing Desired Probabilities

We will compute $\alpha_t(s)$, the probability of being in state s after observing the first t observations.

Initialization ($t = 1$):

$$\alpha_1(\text{DT}) = \pi(\text{DT}) \times B(\text{the} \mid \text{DT}) = 1.0 \times 1.0 = 1.0$$

All other states are zero:

$$\alpha_1(\text{NN}) = \alpha_1(\text{VB}) = \alpha_1(\text{IN}) = 0$$

Recursion:

For $t = 2$ (Observation: *cat*)

$$\alpha_2(\text{NN}) = (\alpha_1(\text{DT}) \times A(\text{NN} \mid \text{DT})) \times B(\text{cat} \mid \text{NN}) = (1.0 \times 1.0) \times 0.25 = 0.25$$

Other states remain at zero.

For $t = 3$ (Observation: *sat*)

$$\alpha_3(\text{VB}) = (\alpha_2(\text{NN}) \times A(\text{VB} \mid \text{NN})) \times B(\text{sat} \mid \text{VB}) = (0.25 \times 1.0) \times 0.5 = 0.125$$

For $t = 4$ (Observation: *on*)

$$\alpha_4(\text{IN}) = (\alpha_3(\text{VB}) \times A(\text{IN} \mid \text{VB})) \times B(\text{on} \mid \text{IN}) = (0.125 \times 0.5) \times 1.0 = 0.0625$$

For $t = 5$ (Observation: *the*)

$$\alpha_5(\text{DT}) = (\alpha_4(\text{IN}) \times A(\text{DT} \mid \text{IN})) \times B(\text{the} \mid \text{DT}) = (0.0625 \times 1.0) \times 1.0 = 0.0625$$

For $t = 6$ (Observation: *mat*)

$$\alpha_6(\text{NN}) = (\alpha_5(\text{DT}) \times A(\text{NN} \mid \text{DT})) \times B(\text{mat} \mid \text{NN}) = (0.0625 \times 1.0) \times 0.125 = 0.0078125$$

Step 8: Termination

The probability of the observed sequence is:

$$P(O) = \alpha_6(\text{NN}) = 0.0078125$$

Step 9: Conclusion

The probability of the sequence “The cat sat on the mat” occurring, as per our HMM, is approximately 0.0078. Specifically, the probability of “mat” being observed as a Noun (NN) at position $t = 6$ is:

$$P(\text{mat at } t = 6, \text{state} = \text{NN} \mid O) = \frac{\alpha_6(\text{NN})}{P(O)} = \frac{0.0078125}{0.0078125} = 1.0$$

This indicates that, given the model and the corpus, the word “mat” at position 6 is predicted to be a Noun with probability 1. It is important to note again that the probabilities in this example are based on the limited corpus and a simplified model, and in a real-world scenario, smoothing techniques and larger corpora would be used to provide more accurate estimates.

A.2 Transformer Worked Out Example

To illustrate how a Transformer model processes natural language, we will walk through a simplified example using the sentence “The cat sat on the mat.” This example demonstrates the key components of the Transformer architecture, focusing on the self-attention mechanism.

Step 1: Input Tokenization

The input sentence is tokenized into individual words (tokens):

Input tokens : [The, cat, sat, on, the, mat]

Note that tokenization in Transformers typically happens at a subword level using techniques like Byte Pair Encoding (BPE) or WordPiece, but for simplicity, we consider word-level tokens here.

Step 2: Token Embedding

Each token is converted into a vector representation using an embedding matrix. Let’s denote the embeddings as $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_6$, where $\mathbf{x}_i \in \mathbb{R}^d$, and d is the embedding dimension.

Step 3: Positional Encoding

Positional encodings are added to the token embeddings to retain positional information since Transformers do not inherently capture sequence order. The input to the encoder becomes:

$$\mathbf{z}_i = \mathbf{x}_i + \mathbf{p}_i \quad \text{for } i = 1, \dots, 6$$

where $\mathbf{p}_i$ is the positional encoding vector for position i .

Step 4: Self-Attention Mechanism

The self-attention mechanism allows the model to weigh the relevance of each token with respect to others.

Computing Queries, Keys, and Values:

The input vectors $\mathbf{z}_i$ are projected into queries $\mathbf{q}_i$, keys $\mathbf{k}_i$, and values $\mathbf{v}_i$ using learned weight matrices $\mathbf{W}^Q$, $\mathbf{W}^K$, and $\mathbf{W}^V$:

$$\mathbf{q}_i = \mathbf{W}^Q \mathbf{z}_i, \quad \mathbf{k}_i = \mathbf{W}^K \mathbf{z}_i, \quad \mathbf{v}_i = \mathbf{W}^V \mathbf{z}_i$$

Calculating Attention Scores:

The attention score between token i and token j is computed using the scaled dot-product:

$$\alpha_{ij} = \frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{d_k}}$$

where d_k is the dimension of the key vectors.

Applying Softmax to Obtain Attention Weights:

The attention weights are obtained by normalizing the scores using the softmax function:

$$a_{ij} = \text{softmax}(\alpha_{ij}) = \frac{\exp(\alpha_{ij})}{\sum_{j=1}^6 \exp(\alpha_{ij})}$$

Computing the Output:

Each output vector $\mathbf{h}_i$ is calculated as a weighted sum of the value vectors:

$$\mathbf{h}_i = \sum_{j=1}^6 a_{ij} \mathbf{v}_j$$

Simplified Example Calculation

For illustration, suppose $d_k = 1$, and we have simplified scalar values for queries, keys, and values:

Token i	The	cat	sat	on	the	mat
$\mathbf{q}_i$	1.2	0.8	1.0	0.5	1.1	0.9
$\mathbf{k}_i$	1.0	0.9	1.1	0.6	1.0	0.8
$\mathbf{v}_i$	0.5	0.6	0.7	0.4	0.5	0.6

Calculating Attention for Token “The” ($i = 1$):

$$\alpha_{1j} = \frac{\mathbf{q}_1 \mathbf{k}_j}{\sqrt{1}} = \mathbf{q}_1 \mathbf{k}_j$$

Compute the scores:

$$\alpha_{1j} = [1.2 \times 1.0, 1.2 \times 0.9, 1.2 \times 1.1, 1.2 \times 0.6, 1.2 \times 1.0, 1.2 \times 0.8] = [1.2, 1.08, 1.32, 0.72, 1.2, 0.96]$$

Applying Softmax:

$$a_{1j} = \frac{\exp(\alpha_{1j})}{\sum_{j=1}^6 \exp(\alpha_{1j})}$$

Compute exponentials (approximate values):

$$\exp(\alpha_{1j}) \approx [3.32, 2.95, 3.74, 2.05, 3.32, 2.61]$$

Sum of exponentials:

$$\text{Total} = 3.32 + 2.95 + 3.74 + 2.05 + 3.32 + 2.61 = 17.99$$

Attention weights:

$$a_{1j} = \left[\frac{3.32}{17.99}, \frac{2.95}{17.99}, \frac{3.74}{17.99}, \frac{2.05}{17.99}, \frac{3.32}{17.99}, \frac{2.61}{17.99} \right] \approx [0.184, 0.164, 0.208, 0.114, 0.184, 0.145]$$

Computing Output $\mathbf{h}_1$:

$$\mathbf{h}_1 = \sum_{j=1}^6 a_{1j} \mathbf{v}_j = (0.184)(0.5) + (0.164)(0.6) + (0.208)(0.7) + (0.114)(0.4) + (0.184)(0.5) + (0.145)(0.6)$$

Calculate:

$$\mathbf{h}_1 = 0.092 + 0.098 + 0.146 + 0.046 + 0.092 + 0.087 \approx 0.561$$

Step 5: Position-Wise Feedforward Network

Each $\mathbf{h}_i$ is passed through a feedforward network:

$$\mathbf{y}_i = \text{ReLU}(\mathbf{h}_i \mathbf{W}_1 + \mathbf{b}_1) \mathbf{W}_2 + \mathbf{b}_2$$

Step 6: Output of the Encoder

The outputs $\mathbf{y}_i$ represent context-aware embeddings of the input tokens, capturing the relationships between words.

Conclusion

This example demonstrates how the Transformer model processes the sentence “The cat sat on the mat” using self-attention to capture contextual relationships. By computing attention weights, the model identifies which words are most relevant to each other at each position, enabling effective encoding of the sequence.

Appendix B

Miscellaneous

Table B.1: Instances of insights from *InsightBench* for each category.

Dataset	Cate- gory	Insights	Plot
Incident	Manage- ment	Question: What is the trend in time to resolution (TTR) of incidents across categories? Insight: TTR starts to increase linearly for hardware incidents abruptly during a particular time period.	
User	Management	Question: What is the distribution of reporters per Manager by Department? Insight: The average number of reporters per manager in the IT department is significantly higher than other departments. It is 50.5 compared to 8.8 in Customer Support, 11.6 in Finance, 12.8 in HR, and 13.0 in Sales.	
Assets	Manage- ment	Question: What is the relationship between the purchase date of assets and their warranty periods? Insight: There is a significant correlation between purchase dates and warranty periods in that the later one purchases an asset the higher the warranty which is a surprising insight.	
Expense	Manage- ment	Question: What is the distribution of processing times of expense reports across different cost brackets? Insight: Counter-intuitively, expenses within lower cost brackets experience significantly longer processing times.	
Goal	Management	Question: How long do goals under 'Cost Reduction' take to achieve? Insight: As time passes, there is an increase in the time it takes to achieve 'Cost Reduction' goals.	

Ground Truth Insight	Insight Predicted by PandasAgent (gpt-4o)	Insight Predicted by AgentPoirot (gpt-4o)
Dataset Category: Incident Management		
The time to resolution of incidents is increasing over time	Network assignment group consistently has longer resolution times compared to other groups, with an average resolution time of approximately 1074 hours.	Yes, there is a clear trend of increasing incident resolution times over the months, with the average resolution time rising from 113.03 hours in January 2023 to 3150.86 hours in June 2024.
The increase in volume of incidents is seen only for one particular category i.e. Hardware	The trend in assigned incidents varies significantly across different assignment groups, with the "Hardware" group having the highest number of incidents (314), followed by "Software" (56), "Network" (48), "Database" (43), and "Service Desk" (39).	The category ' Hardware ' has shown the highest increase in the number of incidents over the given period, with a total increase of 32.0 incidents.
Dataset Category: User Management		
Fred Luddy has a significantly higher average TTR compared to other agents	There are specific types of incidents where the specified human agent, Fred Luddy, consistently performs worse than others.	The absence of Howard Johnson significantly impacts the workload and performance of other agents, as evidenced by the high mean resolution times, particularly for Fred Luddy at 410.055 hours , compared to other agents like Beth Anglin at 102.210 hours.
There is no significant change in the distribution of incident categories for Beth Anglin and Luke Wilson during the other agents' PTO	During peak business hours, Luke Wilson and Beth Anglin receive the highest number of incident assignments, while off-peak hours show a more balanced distribution among agents.	No specific periods identified where the number of incidents assigned to agents significantly increases or decreases, and no correlation with agents' leave periods has been established.
Dataset Category: Asset Management		
Servers and Web Servers are the most expensive asset categories on average, followed by computers.	Yes, there are cost outliers within specific model categories as the maximum cost of an asset is 4786, which is significantly higher than the 75th percentile cost of 2834.	Yes, certain model categories such as ' Server ' and ' Computer ' have a higher variance in asset costs compared to others, with variances of 4,441,227.02 and 1,063,157.44 respectively, and these categories also have relatively longer mean warranty periods of 730.87 and 1,228.08 days.
There is an excessive number of computers per user in HR Department	Yes, in the HR department, the 'Computer' model category has significantly higher costs compared to other models, totaling 61215.	Yes, the HR department has a disproportionately high number of Computers (19) compared to other model categories such as Web Servers (5) and Servers (4).
Dataset Category: Finance Management		
Most of the declined expenses are belong to a few specific users in IT Department	There is a correlation between the user submitting the expense and the likelihood of rejection, as some expenses are declined, indicating a rejection, while others are processed or pending.	Yes, specific users such as Vernon Engelman and Helene Iberg from the IT department have higher rates of expense rejections , particularly in the 'Assets' category.
There is a linear positive correlation between new employee's start dates and their expense rejection rates	There are differences in the approval rates of expense submissions between newer and more experienced employees , with newer employees having higher rates of submission errors and rejections compared to more experienced employees.	The rejection rate of expense submissions tends to be higher for employees with shorter employment durations , as indicated by the higher rejection rates observed in the initial days of employment.
Dataset Category: Goal Management		
There is a uniform distribution of goal priorities in the Finance department	There is no significant correlation between the owner of a goal in the Finance department and the time taken for completion.	The priority level of goals in the Finance department is relatively balanced , with counts of 58 for Low, 52 for Medium, 52 for High, and 51 for Critical priorities.
The IT department exhibits a higher number of both Critical and High priority goals compared to other departments	The Marketing department has the highest number of goals, followed by HR, IT, and Finance.	The distribution of goal priorities within the IT department is 56 High, 54 Critical, 10 Low, and 9 Medium, which shows a higher concentration of High and Critical priorities compared to other departments.

Table B.2: PandasAgent v/s AgentPoirot. **Bold text** shows why an insight is a close approximation of the ground truth insight. 186

Appendix C

Prompts Used

In this section, we show all the prompts used in this work.

Prompt C.1: Prompt used for Generating New Articles

```
### Instruction:
You are an expert data generator tasked with synthesizing new documents
for a
summarization task. {dataset_description}

Below are some example documents and their summaries from a group in the
dataset
(group 1):
{gp1_documents}

Below are some example documents and their summaries from another group
in the dataset (group 2):
{gp2_documents}

Given the above documents, follow these instructions:
* Synthesize a new document that follows a similar format to the examples
  provided.
* The document should contain {document_size}.
* The document should be coherent and relevant to the topic.
* The document should be original and not copied from the examples.
* Ensure that the document covers {alpha}% topics from group 1 and 1 - {
  alpha}% topics from group 2.
* Wrap your response in the <document></document> tags.
* Do NOT include anything else like the examples in your output.

### Response:
```

Input Prompt:

Each example in the following list contains a sentence that belongs to a category. A category is one of the following:
music_likeness, play_music, music_settings:

```
sentence: next i want to hear shinedown ;  
category: play_music  
sentence: i am the living blues ;  
category: music_likeness  
sentence: open music player settings ;  
category: music_settings  
sentence: play hopsin from my latest  
playlist ; category: play_music  
sentence: i like this song ;  
category:
```

GPT-3 Predictions:

```
play_music,music_likeness,music_settings,  
music_likeness,music_likeness,help_command
```

Figure C.1: **Using GPT-3 as a classifier.** Given a triplet of close intents, we mix and shuffle the multiple seed examples available for each of them. Then, we append an incomplete line to the prompt with just the generated sentence and feed it to GPT-3 multiple times. Among the responses, we choose the most generated in-triplet intent as the predicted intent (“music_likeness” in the above example). **Note:** For brevity, we don’t show all the seed examples and predictions.

Prompt C.2: Prompt used for Scoring a Generated Summary

Below is an instruction that describes a task. Write a response that appropriately completes the request.

Instruction:

Input Prompt:

Consider the task of classifying between the following classes (along with some examples):

1. age_limit, which is about customer inquiries on age-related restrictions for opening a bank account. Some examples of utterances include:

 - Can I get an account for my son?
 - Can my teenager have an account?

2. atm_support, which is about users asking how to use an ATM, where to find one, or any other clarifications about a transaction at an ATM. Some examples of utterances include:

 - Is the closest ATM to me within 2 miles?
 - Are there only certain ATM machines where I can use this card?

}

Part 1
- Generate a diverse set of 4 short utterances where each utterance belongs 75% to age_limit and 25% to atm_support.

}

Part 2

Example 1:

Completions:

- Can someone under 18 open an account with unlimited ATM withdrawal limit?
- Can I open an account for my teenage daughter?
- Do I need to be over a certain age to use an ATM?
- Can I use my children's debit card to withdraw money from the ATM?

Figure C.2: **PromptMix prompt.** The demonstration highlights the two main parts of our prompt: Part 1 shows the description and examples, and Part 2 shows the mixup instruction. We use GPT3.5-turbo to obtain the completions. In this example, we highlight good generations in **blue** and bad generations in **red**. **Note:** for brevity, we only include two classes in the prompt.

```
Given the document:
{document}

Provided Summary:
{summary}
```

Input Prompt:

Consider the task of classifying between the following classes(along with some examples):

1. age_limit, which is about customer inquiries on age-related restrictions for opening a bank account.

Some examples of utterances include:

- Can I get an account for my son?
- Can my teenager have an account?

2. atm_support, which is about users asking how to use an ATM, where to find one, or any other clarifications about a transaction at an ATM. Some examples of utterances include:

- Is the closest ATM to me within 2 miles?
- Are there only certain ATM machines where I can use this card?

Consider the following test sentence:

1. Do I need to be over a certain age to use an ATM?

Classify the test sentence into one of the previously described classes.

1.

Example 1:

Completions:

- atm_support

Figure C.3: **Relabeling prompt.** The generated sentence does not belong to the class `age_limit` as the main objective of the query entails an ATM support query, which is fixed after relabeling. **Note:** we relabel both good and bad examples. The color in the figure is just for demonstration purposes.

Follow these instructions when writing your response:
* On a scale of 1-10, provide a numerical rating for the provided summary
 , with 10 denoting that the provided answer perfectly surmises the
 main points of the document.

```
* Your response should contain only the numerical rating. DO NOT include
  anything else like the provided answer, the ground truth answer, or
  an explanation of your rating scale in your response.
* Wrap your numerical rating inside <rating></rating> tags.
* Check very carefully before answering.
* Follow the output format as shown in the example below:
Example response:
<rating>7</rating>

### Response:
```

Prompt C.3: Prompt used for Summarizing an Article in *MixSumm*

```
Below is an instruction that describes a task. Write a response that
  appropriately
  completes the request.

### Instruction:
You are an expert data annotator tasked with summarizing documents for a
  summarization task. {dataset_description}

Below are some example documents and their summaries from a group in the
  dataset (group 1):
{gp1_documents}

Below are some example documents and their summaries from another group
  in the dataset (group 2):
{gp2_documents}

A document is composed of the following sentences:
{sences}

Given the sentences above:
* You are to construct an extractive summary for the document by
  selecting some sentences from above.
* The summary captures the main points of the article.
* Now, output the probability of a sentence being included in the summary
  .
* Do NOT include anything else like the sentence in your output.
* Output your probabilities in the format <line id>. <probability>. Refer
  to the example below:
1. 0.73
2. 0.65
3. 0.95
etc.
```

Response:

Prompt C.4: Prompt used for Extracting Questions from Data

SYSTEM MESSAGE:

You the manager of a data science team whose goal is to help stakeholders within your company extract actionable insights from their data. You have access to a team of highly skilled data scientists that can answer complex questions about the data. You call the shots and they do the work. Your ultimate deliverable is a report that summarizes the findings and makes hypothesis for any trend or anomaly that was found.

PROMPT:

Instruction:

Given the following context:

<context>{context}</context>

Given the following goal:

<goal>{goal}</goal>

Given the following schema:

<schema>{schema}</schema>

Instructions:

- * Write a list of questions to be solved by the data scientists in your team to explore my data and reach my goal.
- * Explore diverse aspects of the data, and ask questions that are relevant to my goal.
- * You must ask the right questions to surface anything interesting (trends, anomalies, etc.)
- * Make sure these can realistically be answered based on the data schema.
- * The insights that your team will extract will be used to generate a report.
- * Each question should only have one part, that is a single '?' at the end which only require a single answer.
- * Do not number the questions.
- * You can produce at most {max_questions} questions. Stop generation after that.
- * Most importantly, each question must be enclosed within <question></question> tags. Refer to the example response below:

Example response:

<question>What is the average age of the customers?</question>

```
<question>What is the distribution of the customers based on their age?</  
question>
```

```
### Response:
```

Prompt C.5: Prompt used for Generating and Executing code for answering a Question

```
PROMPT:
Given the goal:\n
{goal}

Given the schema:\n
{schema}

Given the data path:\n
{database_path}

Given the list of predefined functions in cba.tools module and their
example usage:\n\n
{function_docs}

Give me the python code required to answer this question "{question}" and
put a comment on top of each variable.\n\n

Make a single code block for starting with ‘‘python
Do not produce code blocks for languages other than Python.
Import cba.tools at the beginning.
You must only use the predefined functions mentioned above to make the
plot.
You must generate one single simple plot and save it as a jpg file.
For the plot, save a stats json file that stores the data of the plot.
For the plot, save a x_axis.json and y_axis.json file that stores a
maximum of 50 of the most important x and y axis data points of the
plot, respectively.
Save each json file using the cba.save_json function
For the json file must have a "name", "description", and "value" field
that describes the data.
The content of the json file should be less than 4500 characters

Call the fix_fnames function in cba.tools at the end of your code.
End your code with ‘‘.

Output code:\n
```

Prompt C.6: Prompt used for Generating Diverse Follow-up Questions based on the previous answer

SYSTEM:

You the manager of a data science team whose goal is to help stakeholders within your company extract actionable insights from their data. You have access to a team of highly skilled data scientists that can answer complex questions about the data. You call the shots and they do the work. Your ultimate deliverable is a report that summarizes the findings and makes hypothesis for any trend or anomaly that was found.

PROMPT:

Hi, I require the services of your team to help me reach my goal.

<context>{context}</context>

<goal>{goal}</goal>

<schema>{schema}</schema>

<question>{question}</question>

<answer>{answer}</answer>

Instructions:

- * Produce a list of follow up questions explore my data and reach my goal.
- * Note that we have already answered <question> and have the answer at <answer>, do not include a question similar to the one above.
- * Explore diverse aspects of the data, and ask questions that are relevant to my goal.
- * You must ask the right questions to surface anything interesting (trends, anomalies, etc.)
- * Make sure these can realistically be answered based on the data schema.
- * The insights that your team will extract will be used to generate a report.
- * Each question that you produce must be enclosed in <question>content</question> tags.
- * Each question should only have one part, that is a single '?' at the end which only require a single answer.
- * Do not number the questions.
- * You can produce at most {max_questions} questions.

Prompt C.7: Prompt used for Generating only one type of Follow-up Questions based on the previous answer

```
SYSTEM:
You the manager of a data science team whose goal is to help stakeholders
    within your company extract actionable insights from their data.
You have access to a team of highly skilled data scientists that can
    answer complex questions about the data.
You call the shots and they do the work.
Your ultimate deliverable is a report that summarizes the findings and
    makes hypothesis for any trend or anomaly that was found.

PROMPT:
Hi, I require the services of your team to help me reach my goal.

<context>{context}</context>

<goal>{goal}</goal>

<schema>{schema}</schema>

<question_type>{question_type}</question_type>

<question>{question}</question>

<answer>{answer}</answer>

Instructions:
* Produce a list of follow-up questions to explore my data and reach my
  goal.
* Note that we have already answered <question> and have the answer at <
  answer>, do not include a question similar to the one above.
* Explore diverse aspects of the data, and ask questions that are
  relevant to my goal.
* You must ask the right questions to surface anything interesting (
  trends, anomalies, etc.)
* Make sure these can realistically be answered based on the data schema.
* The insights that your team will extract will be used to generate a
  report.
* The question has to adhere to the type of question that is provided in
  the <question_type> tag
* The type of question is either descriptive, diagnostic, prescriptive,
  or predictive.
* Each question that you produce must be enclosed in <question>content</
  question> tags.
* Each question should only have one part, that is a single '?' at the
```

```
end which only require a single answer.  
* Do not number the questions.  
* You can produce at most {max_questions} questions.
```

Prompt C.8: Prompt used for Interpreting the Solution

```
PROMPT:
### Instruction:
You are trying to answer a question based on information provided by a
  data scientist.

Given the context:
<context>
  You need to answer a question based on information provided by a data
  scientist.
</context>

Given the goal:
<goal>{goal}</goal>

Given the question:
<question>{question}</question>

Given the analysis:
<analysis>
  <message>
    {message}
  </message>
  {insights}
</analysis>

Instructions:
* Based on the analysis and other information provided above, write an
  answer to the question enclosed with <question></question> tags.
* The answer should be a single sentence, but it should not be too high
  level and should include the key details from justification.
* Write your answer in HTML-like tags, enclosing the answer between <
  answer></answer> tags, followed by a justification between <
  justification></justification> tags.
* Refer to the following example response for the format of the answer
  and justification.

Example response:
<answer>This is a sample answer</answer>
<justification>This is a sample justification</justification>

### Response:
```

Prompt C.9: Prompt used for Selecting the best Question based on previously asked questions and the goal

```
SYSTEM MESSAGE:
You the manager of a data science team whose goal is to help stakeholders
    within your company extract actionable insights from their data.
You have access to a team of highly skilled data scientists that can
    answer complex questions about the data.
You call the shots and they do the work.
Your ultimate deliverable is a report that summarizes the findings and
    makes hypothesis for any trend or anomaly that was found.

PROMPT:
Hi, I require the services of your team to help me reach my goal.

<context>{context}</context>

<goal>{goal}</goal>

<prev_questions>{prev_questions_formatted}</prev_questions>

<followup_questions>{followup_questions_formatted}</followup_questions>

Instructions:
* Given a context and a goal, select one follow up question from the
    above list to explore after prev_question that will help me reach my
    goal.
* Do not select a question similar to the prev_questions above.
* Output only the index of the question in your response inside <
    question_id></question_id> tag.
* The output questions id must be 0-indexed.
"""
```

Prompt C.10: Prompt used for Summarizing the Insights

SYSTEM_MESSAGE:

You the manager of a data science team whose goal is to help stakeholders within your company extract actionable insights from their data.

You have access to a team of highly skilled data scientists that can answer complex questions about the data.

You call the shots and they do the work.

Your ultimate deliverable is a report that summarizes the findings and makes hypothesis for any trend or anomaly that was found.

PROMPT:

Hi, I require the services of your team to help me reach my goal.

<context>{context}</context>

<goal>{goal}</goal>

<history>{history}</history>

Instructions:

* Given a context and a goal, and all the history of <question_i><answer_i> pairs from the above list generate the 3 top insights that will help me reach my goal.

* Output each insight within this tag <insight></insight>.

Prompt C.11: Prompt used for generating the synthetic data for non-controllable elements of an incidents dataset.

```
gpt_fields = ["short_description", "assignment_group"]

field_info = "\n".join(
    [
        "<field>"
        + f"<name>{f}</name><type>{col_info[f]['type']}</type>"
        + (
            f"<choices>{col_info[f]['choices']}</choices>"
            if "choices" in col_info[f]
            else ""
        )
        + "</field>"
        for f in gpt_fields
    ]
)

prompt = f"""
Create a fake ServiceNow incident by generating realistic values for the
following fields:
<fields>
    {gpt_fields}
</fields>

Here are the types of each of these fields and the permitted values:
<field_info>
    {field_info}
</field_info>

Produce a json dictionary with a single value for each of these fields.
Make sure to respect allowed values and use their full diversity.

Example:
{"short_description": "My laptop is broken"}

Respond with the json only.

"""

valid_data = []
while len(valid_data) < num_samples:
    try:
```

```

        completion = client.chat.completions.create(
            model="gpt-4-turbo",
            messages=[
                {
                    "role": "system",
                    "content": "You're going to help me generate
simulated data that looks like it really came from ServiceNow Glide
tables.",
                },
                {"role": "user", "content": prompt},
            ],
        )
        incident = json.loads(completion.choices[0].message.content)

        # Validate values
        assert set(gpt_fields) == set(incident.keys())
        for f in gpt_fields:
            if "choices" in col_info[f]:
                assert incident[f] in col_info[f]["choices"]

        # Accept it
        valid_data.append(incident)
        print(f"... {len(valid_data)} / {num_samples}")

    except:
        print("... invalid value, trying again")

```

Prompt C.12: Prompt used for LLaMA-3-Eval Scores

```

SYSTEM:
You are a high school teacher evaluating student responses to a question.
    You are tasked with grading the response based on how well it
    answers the question. You are to provide a numerical rating for how
    well the response answers the question based on the ground truth
    answer.

PROMPT:
Below is an instruction that describes a task. Write a response that
    appropriately completes the request.

### Instruction:
Provided Answer:
{answer}

Ground Truth Answer:
{gt_answer}

```

Follow these instructions when writing your response:

- * On a scale of 1-10, provide a numerical rating for how close the provided answer is to the ground truth answer, with 10 denoting that the provided answer is the same as ground truth answer.
- * Your response should contain only the numerical rating. DONOT include anything else like the provided answer, the ground truth answer, or an explanation of your rating scale in your response.
- * Wrap your numerical rating inside <rating></rating> tags.
- * Check very carefully before answering.
- * Follow the output format as shown in the example below:

Example response:
<rating>7</rating>

Response:

Prompt C.13: Prompt used for many-to-many LLaMA-3-Eval Scores

SYSTEM:

You are a high school teacher evaluating student responses to some questions. Before scoring their answers, you need to first match each ground truth answer with the most appropriate answer provided by the student.

PROMPT:

Below is an instruction that describes a task. Write a response that appropriately completes the request.

Instruction:

Predicted Answers:

{pred_list}

Ground Truth Answers:

{gt_list}

For each ground truth answer above, provide the index of the most appropriate predicted answer (1-indexed).

Each line must contain a single integer value denoting the id of the matched prediction.

If there is no appropriate prediction for a ground truth answer, write -1.

Check very carefully before answering.

Response:

Input Prompt:

Below is an instruction that describes a task.
Write a response that appropriately completes
the request.

Text to summarize:

1. <Line 1 of the input text>
2. <Line 2 of the input text>
3. <Line 3 of the input text>
- .
- .
- N. <Line N of the input text>

For each sentence in the input text, output
the probability of it appearing in the
summary such that the summary captures the key
points of the text. Output in the format
<line id>. <probability>.

Response:

1. 0.9
2. 0.1
3. 0.8
- .
- .
- N. 0.2

(a) **Generating pseudo-labels.** We attach a line ID to each sentence in the input document and instruct the LLM to use those line IDs in its response.

Input Prompt:

Below is an instruction that describes a task.
Write a response that appropriately completes
the request.

Text to summarize:

<Input Conversation/Text to Summarize>

Summary:

<Summary to evaluate>

Part 1

Provide a rating between 0-100 based on the
following criteria:

- 1)The summary should be concise...
- 2)The summary should capture ...
- 3)The summary should be extractive, i.e.,...
- 4)The response should be strictly numerical.

Part 2

Check very carefully before giving your rating.

Response:

85

(b) **Scoring pseudo-labels.** The two-part prompt contains a text and summary pair (Part 1), and a list the evaluation criteria (Part 2). **Note:** Refer to Section 5.1.4 for complete details on the evaluation criteria.

Figure C.4: Different prompts used in the experiments.

Input Prompt:

Below is an instruction that describes a task.
Write a response that appropriately completes
the request.

Text to summarize:

1. <Line 1 of the input text>
2. <Line 2 of the input text>
3. <Line 3 of the input text>
- .
- .
- N. <Line N of the input text>

For each sentence in the input text, output
the probability of it appearing in the
summary such that the summary captures the key
points of the text. Output in the format
<line id>. <probability>.

Response:

1. 0.9
2. 0.1
3. 0.8
- .
- .
- N. 0.2

(a) **Generating pseudo-labels.** We attach a line ID to each sentence in the input document and instruct the LLM to use those line IDs in its response.

Input Prompt:

Below is an instruction that describes a task.
Write a response that appropriately completes
the request.

Text to summarize:

<Input Conversation/Text to Summarize>

Summary:

<Summary to evaluate>

Part 1

Provide a rating between 0-100 based on the
following criteria:

- 1)The summary should be concise...
- 2)The summary should capture ...
- 3)The summary should be extractive, i.e.,...
- 4)The response should be strictly numerical.

Part 2

Check very carefully before giving your rating.

Response:

85

(b) **Scoring pseudo-labels.** The two-part prompt contains a text and summary pair (Part 1), and a list the evaluation criteria (Part 2).

Figure C.5: Different prompts used in the experiments.

```

PROMPT =
"""
You are a linguist who is analyzing the sentence. Please provide the
    number of descriptive action verbs (DAVs), interpretive action verbs
    (IAVs), state verbs (SVs), and adjectives (ADJs) in the sentence.

### Instructions:
Given a sentence:
{sentence}

### Task:
* Output the number of descriptive action verbs (DAVs), interpretive
    action verbs (IAVs), state verbs (SVs), and adjectives (ADJs).
* Do not generate anything else in your output.
* Follow the format below:
<dav>number of DAVs</dav>
<iav>number of IAVs</iav>
<sv>number of SVs</sv>
<adj>number of ADJs</adj>

### Response:
"""

```

Figure C.6: Prompt used to obtain the number of occurrences of four linguistic categories (DAVs, IAVs, SVs, and ADJs) in a lyric line.

In this study, we aim to investigate the inspirational quality of song lyrics and poetic lines. Our objective is to understand how individual lyrics or poetic lines can influence a creative person. By “inspiration,” we refer to how these lines emotionally impact individuals and stimulate new creative ideas.

You will be shown 200 lines in total. Each line will be shown separately. Please take a moment to read each line and provide your rating based on its inspirational quality. We are interested in the potential of these lines to inspire creativity, rather than their grammatical correctness.

Please note that there are no right or wrong answers; we value your subjective impressions. Your feedback will help us understand how different lines of lyrics resonate with people.

The study is expected to take approximately 30 to 40 minutes. If you have to leave before you are done, you can resume the session using the same access code. Thank you for your participation!

Figure C.7: Guidelines presented to the eight human labelers.

```

PROMPT = """
You are a linguist who studies the notion of creative inspiration in the
    context of lyric lines.

### Instructions:
Given the following lines that belong to a "inspiring" category:
{interesting_lines}

Given the following lines that belong to a "not inspiring" category:
{not_interesting_lines}

And given the test sentence:
{sentence}

### Task:
* Predict whether the test sentence belongs to the "interesting" or "not
    interesting" category.
* Predict 0 for "not inspiring" and 1 for "inspiring".
* Enclose your rating between <pred></pred> tags.
* Do not generate anything else in your output.
* By inspiring, we refer to how these lines emotionally impact an artist
    and stimulate new creative ideas.
* Follow the format below:
<pred>1</pred>

### Response:
"""

```

Figure C.8: Prompt used to classify a lyric line with LLaMA-3-70b.

```

PROMPT = """
### Instructions:
You are provided with some text and your task is to classify it into one
of the following emotions (each mapped to an ID):

<emotion_map>
{emotion_mapping}
</emotion_map>

<input_text>
{text}
</input_text>

Follow these instructions when writing your response:
* Output the ID and name of the emotion from the list above that best
fits the input text.
* Your response should contain only the numerical id and the name of the
emotion. DONOT include anything else like the input text or the
emotion map.
* Wrap the name of the predicted emotion within <emotion></emotion> tags.
* Wrap the id of the predicted emotion within <emotion_id></emotion_id>
tags.
* Check very carefully before answering.
* Follow the output format as shown in the example below:
Example response:
<emotion>confusion</emotion>
<emotion_id>7</emotion_id>

### Response:
"""

```

Figure C.9: Prompt used to classify a lyric line into one of 29 emotion categories with LLaMA-3-70b.

```
PROMPT = """
You are a linguist who is analyzing the poetic imagery of a given textual
expression.

### Instructions:
Given the expression:
"{sentence}"

### Task:
* Provide a rating between 1-5 on how much poetic imagery the expression
  has.
* You must enclose your rating between <rating></rating> tags.
* Do not generate anything else in your output.
* Follow the format below:
<rating>3</rating>

### Response:
"""
```

Figure C.10: Prompt used to measure poetic imagery of a lyric line.

```

"""
You are a linguist who is analyzing the poetics of a textual expression.

### Instructions:
PROMPT =
"""
Given the expression:
"{sentence}"

### Task:
* Provide a rating between 1-5 on how banal or cliché the poetic
  expression in the provided text is.
* A rating of 5 indicates a highly banal or cliché expression, while a
  rating of 1 indicates a unique or original expression.
* For instance, "echoes will linger" is a banal expression, compared to "
  i'll hide the answers in the sky".
* You must enclose your rating between <rating></rating> tags.
* Do not generate anything else in your output.
* Follow the format below:
<rating>3</rating>

### Response:
"""

```

Figure C.11: Prompt used to measure banality of a lyric line.

```
PROMPT = ""
You are a linguist who is analyzing the poetics of a textual expression.

### Instructions:
Given the expression:
"{sentence}"

### Task:
* Provide a rating between 1-5 on how much energy or emotional depth the
  concepts in the expression have.
# For instance, expressions with words like "sea", "sky", "mercy" usually
  tend to have higher perceived energy due to the depth in those words
.
* The notion of energy also covers diction and symbolism.
* You must enclose your rating between <rating></rating> tags.
* Do not generate anything else in your output.
* Follow the format below:
<rating>3</rating>

### Response:
""
```

Figure C.12: Prompt used to measure word energy of a lyric line.