Comparing Prompt Engineering Techniques

Anum Ahmed
aqa2001

Tramy Dong
td2748

Kyra Ramesh Krishna
kr3026

Abstract

This project systematically evaluates the impact of different prompt engineering strategies on the performance of language models on sentiment classification tasks. We compared four prompting techniques—zero-shot, few-shot, chain-of-thought, and meta—using the same datasets and evaluation metrics. Our experiments revealed several unexpected findings: the FlanT5 large model outperformed larger models like ChatGPT in predicting poetry sentiment, challenging the assumption that larger models would perform better. Additionally, zero-shot prompting achieved the highest accuracy, contrary to our expectation that chain-of-thought would be more effective. These results suggest that the size of a model does not necessarily correlate with better performance, and more complex prompting techniques may not always be suited for tasks like poetry sentiment analysis. Our findings have implications for optimizing prompt engineering strategies, particularly for sentiment classification tasks, and highlight the need for further research into the relationship between model size, prompt complexity, and task-specific performance.

1 Introduction

In-context learning has emerged as a defining capability of pre-trained large language models (LLMs), enabling them to perform tasks solely based on prompt input, without requiring any gradient updates. (Shah, 2023) This makes LLMs adaptable to a wide range of tasks when prompts are engineered well. Exploring this area can help us improve model performance, reduce computational costs, and inform better design principles, making LLMs more accessible and effective for broader applications.

While prior studies have explored techniques like few-shot and chain-of-thought prompting (Schulhoff et al., 2024; Zhao et al., 2021), the factors contributing to prompt effectiveness remain an underexplored area in the natural language processing (NLP) community. This project seeks to systematically evaluate the impact of different prompt engineering strategies on model performance. Specifically, we investigate one-shot, few-shot, meta, and chain-of-thought prompting on a semantic classification task to address the central research question: How do different prompt engineering techniques (one-shot, few-shot, meta, and chain-of-thought) influence the performance of an LLM?

In addition to our primary research question, we would like to explore supplemental questions to deepend our understanding of prompt engineering:

1. Can prompts be programmatically generated (e.g., using techniques like Auto-CoT) to achieve comparable performance to manually crafted prompts?

2. How do small and large models perform relative to each other to different prompting techniques?

3. Are certain prompting techniques more effective for specific data types like poetry versus tweets?

2 Methods

2.1 Datasets

The original and main dataset we chose to use was the poem sentiment dataset. This dataset is from google research, and was available on Huggingface where we used their API to collect the data. Originally, the data was used in (Sheng and Uthus, 2020) to investigate bias from poetry and is a collection of verses from the Gutenberg poetry set. Lines of poetry were classified into four distinct categories, (0) Negative, (1) Positive, (2) No Impact, (3) Mixed. Here are the train, validation, and test splits of the data. Here is an example line from each category.

Text	Label
with pale blue berries. in these peaceful shades–	Positive
it flows so long as falls the rain,	No Impact
and that is why, the lonesome day,	Negative
when i peruse the conquered fame of heroes, and the victories of mighty generals, i do not envy the generals,	Mixed

Table 1: Example Data with Labels for Poem Sentiment

Dataset	Number of Rows
Train	892
Validation	105
Test	104
Total	1101

Table 2: Dataset Splits

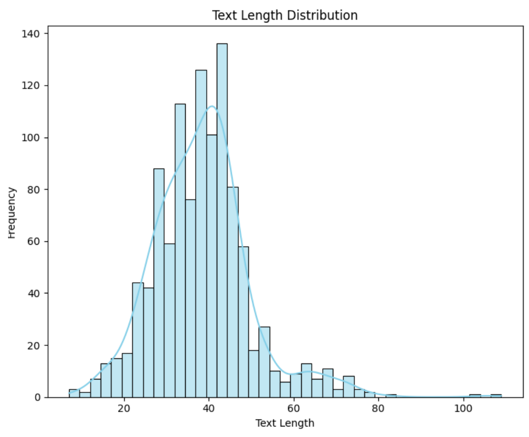

Figure 1: Input Length Distribution [Poems]

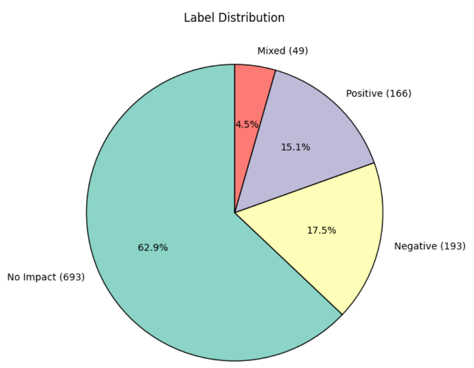

Figure 2: Label Distribution [Poems]

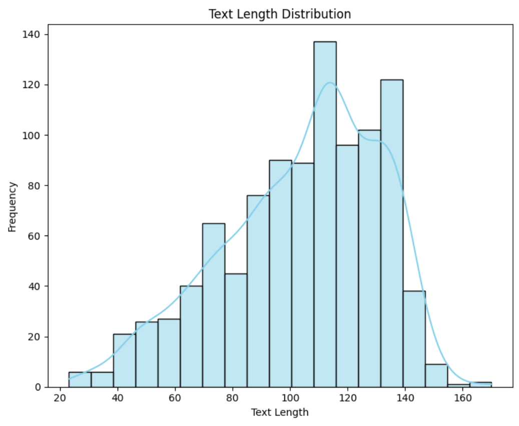

Figure 3: Input Length Distribution [Tweets]

Additionally, there was a broad distribution of input length for the lines of poetry. Figure 1 is a breakdown of the different lengths and their frequency in the combined set. Please note that because in-context learning does not require fine tuning and our goal in the experiment was to compare prompting types and their impact, we combined the training. test, and validation datasets to give us the most data to work with. We truncated the data at 1000 rows to ensure that the models would be able to run on the interference API through huggingface without using up all of our credits.

We also wanted to see how what the distribution was like for data between labels. Figure 2 shows a pie chart with this distribution.

After recieving feedback from the TAs and instructor, we also wanted to see if these models may perform better on non-poem data. This would help provide context to our results– do these models perform better with more colloquial english compared to the more challenging prose of poetry? Are there nuances in the poem sentiment dataset that may lead to different results? To explore this question, we also ran the experiment with the TweetEval dataset found in (Rosenthal et al., 2017). Table 2, Figure 3, and Figure 4 show the truncated data that we used for the twitter sentiment dataset. We also used the yelp polarity dataset found in (Zhang et al., 2015). Table 3, Figure 5, and Figure 6 show the breakdown from the yelp data. By using these datasets, we will also be able to see how varying input lengths and more balanced data can impact results.

Text	**Label**
Happy Birthday Nick J May you live long and Happy :)	Positive
Who's going to get them Gucci foams tomorrow,	Neutral
So disappointed in wwe summerslam! I want to see john cena wins his 16th title,	Negative

Table 3: Example Data with Labels for Tweet Sentiment

Text	**Label**
Awesome drink specials during happy hour. Fantastic wings that are crispy and delicious, wing night on Tuesday and Thursday!	Positive
Very disappointed in the customer service. We ordered Reuben's and wanted coleslaw instead of kraut. They charged us $3.00 for the coleslaw. We will not be back . The iced tea is also terrible tasting.,	Negative

Table 4: Example Data with Labels for Yelp Sentiment

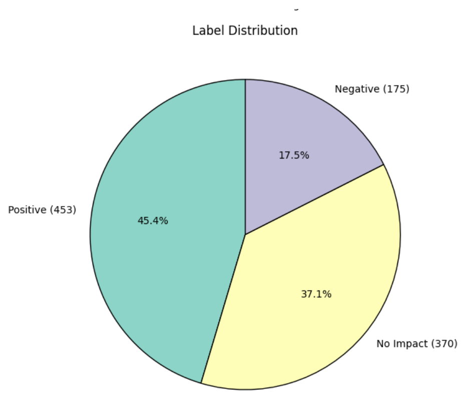

Figure 4: Label Distribution [Tweets]

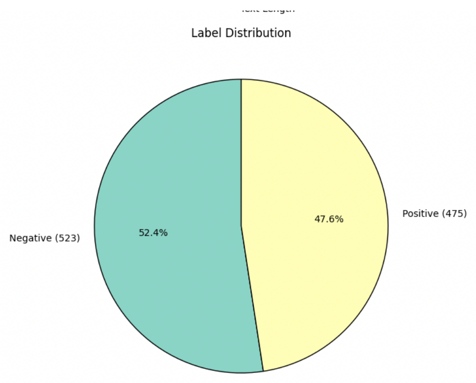

Figure 6: Label Distribution [Yelp]

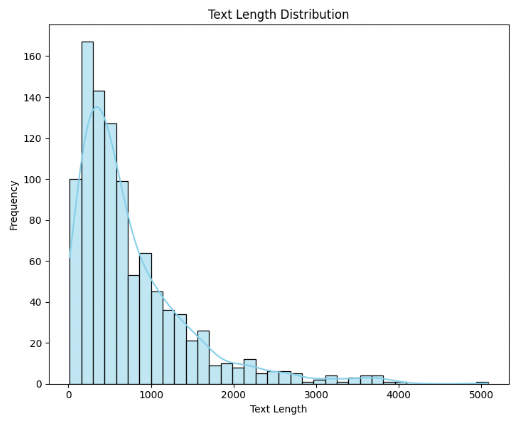

Figure 5: Input Length Distribution [Yelp]

2.2 Models

The final collection of models we will import and use for our prompt-based experiments are Flan-T5 (small and large) and GPT (4o-mini and 3.5). These models were chosen for their distinct advantages and functionalities, making them valuable for a broad range of natural language processing (NLP) tasks. Each model brings unique strengths to the table, allowing us to explore the versatility and effectiveness of each of the four prompt engineering strategies.

Flan-T5 is the instruction fine-tuned version of the Text-to-Text Transfer Transformer (T5) model developed by Google Research (Chung et al., 2022). The core innovation of T5 lies in its ability to reframe NLP tasks into a unified text-to-text format, enabling it to handle tasks such as translation, summarization, question answering, and classification using a single framework (Matricardi, 2024). Flan-T5 builds on this foundation by incorporating instruction fine-tuning, a specialized process that involves training the model on datasets specifically designed to improve its understanding of instruction-based prompts. This fine-tuning significantly enhances the model's ability to follow instructions accurately and perform well across diverse tasks. One of Flan-T5's key strengths is its proficiency in few-shot prompting, where it can deliver precise outputs with as few as 3–4 examples. Despite being smaller in scale compared to massive models like PaLM 62B, Flan-T5 achieves a remarkable balance between computational efficiency and performance, making it a practical choice for many applications.

In our initial experiment, we utilized GPT-3.5-turbo-instruct, which is another modern and up to date model that is part of the GPT-3.5 series. The GPT-3.5-turbo-instruct is a fine-tuned version of the 3.5 model that has been optimized for versatility and can handle both chat-based interactions and more traditional NLP tasks (Jeremiah, 2024). We hypothesized that it would be helpful for our project because it had been further enhanced to improve its understanding of user instructions and generate human-like responses with high coherence. However, we later transitioned to the GPT-4o-mini model instead, because the 3.5 chat completion is outdated (deprecated) so we opted for the newest 4.0 mini version. The model is powered by the Chat Completions API, which enables it to excel in tasks requiring dynamic interaction and context awareness. Compared to earlier versions of GPT, GPT-4o-mini demonstrates significant improvements in its alignment with complex reasoning tasks, and user intent, allowing for greater control over the outputs it generates (Starks, 2024). This model's ability to deliver high-quality results in both structured and open-ended tasks makes it an indispensable tool in our work.

By integrating these two models into our workflow, we aim to harness their complementary strengths to push the boundaries of prompt engineering. Each model contributes unique capabilities: Flan-T5 excels in text-to-text task generalization and few-shot learning and GPT-4o-mini offers versatility and strong alignment with user intent. Together, they form a solid foundation for exploring advanced prompting techniques, enabling us to investigate the full potential of combining diverse AI models to solve a wide range of NLP tasks. This integration will not only deepen our understanding of these models but also pave the way for innovative solutions in the field of natural language processing.

2.3 Approach

To set up our project environment, we implemented custom Python scripts that imported datasets, cleaned and parsed the data, and set them up for prompt engineering. We then wrote code for each of the different prompting techniques. For example, *zero_shot_prompt*(text) simply returns the prompt as the instruction and text, while *few_shot_prompt*(text) returns a prompt with the instructions and text in addition to some examples of acceptable answers. Chain of thought prompting's function is set up to return the instruction and examples of answer and reasoning. Finally, *meta_prompt*(text) returns a prompt that lists step-by-step solution structures for the model to reference when generating responses.

2.4 Experiments

The primary aim of this project was sentiment classification using three different datasets: `google-research-datasets/poem_sentiment`, `cardiffnlp/tweet_eval`, and `fancyzhx/yelp_polarity`. To investigate our main research question, we applied four different prompting techniques (`zero-shot`, `few-shot`, `chain-of-thought`, and `meta`) to each sentiment classification task. We ran this with the four models mentioned above (`gpt-4o-mini`, gpt-3.5-turbo, flan-t5-large, flan-t5-small).

Taking inspiration from Sentiment Analysis in the Era of Large Language Models: A Reality (Zhang et al., 2023), we decided to write two key programs, `predict.py`, and `evaluate.py`. The first (`predict.py`) takes in the model name, data source, and setting (type of prompting technique) as parameters, sends a request to the respective OpenAI and HuggingFace Inference APIs, and writes the predictions to .csv to an output directory. The latter program (`evaluate.py`) takes in the same arguments and evaluates the predictions in the output directory against the original dataset with true labels.

When we began this project, we had decided to only run poem sentiment classification. However, on successfully running `flan-t5-small` on poem sentiment data, we realized that we could generalize our code to apply several other sentiment classification tasks as well. Our `predict.py` function will parse and validate its command line arguments, process the locally stored dataset, generate and send a query to the given model, and then write the prediction out to a dataset uniquely named for that model and task combination. To evaluate the results of each prediction, we built out the `evaluate.py` program. Similar to `predict.py`, this began as a program custom built for poem sentiment classification. We noticed that some models predict a number, while others predict the label by its text label, and others still generate a sentence explanation alongside the label. We built a custom `process_predictions` function that will parse the predicted text and map it to a number associated with that task's label set. For example, the following prediction texts: `'0'`, `'negative'`, and `'The sentiment of the text "he says he's hungry,—he would rather have" can be classified as` **`0 (negative)`**`. The expression of hunger implies a sense of lack or dissatisfaction.'`, will all get mapped to the label number 0. After processing the prediction data, it extracts the true labels from the given .csv files, calculates performance metrics like the F1 scores, and generates a classification report. It also saves errors (misclassifications) in a .csv file for further analysis. Additionally, it ensures that the necessary input parameters are provided and valid, with default values for optional arguments. To complete all experiments within the allocated time frame and stay within the free tiers of OpenAI's API and HuggingFace's Inference API, we sampled fewer than 1000 datapoints from each dataset.

2.5 Automation

This project automates several processes related to data processing, prediction running, and evaluation, leveraging programming to streamline each step:

1. **Data Preparation**: `make_data.py`: The script automates the process of extracting data from an external source (in this case, the `google-research-datasets/poem_sentiment` dataset on Hugging Face) and preparing it for use. It combines training, validation, and test data into a cohesive dataset, saving the output in a CSV file (`poem_sentiment_data.csv`) located in the `poem_data` folder. This script reduces manual effort by automating data retrieval and organization, ensuring that the data is ready for model training and evaluation.

2. **Generalizable Predict and Evaluation Programs**: `predict.py` and `evaluate.py` automate the prediction and evaluation process for sentiment classification tasks using different models (e.g., FlanT5 and ChatGPT).

3. **Batch Processing and Optimization**: We created bash scripts in the `/scripts` folder (`run_gpt.sh`, `eval_gpt.sh`, `run_flant5.sh`, `eval_flant5.sh`) to automate the running of predictions and evaluations for multiple configurations at once. By editing the script to include API keys and running them in one go, users can automatically process multiple data sources and models, reducing repetitive tasks and increasing productivity.

2.6 Reproducibility

A key challenge we faced in this project was the lack of reproducibility in several related works we aimed to emulate. To address this, we prioritized reproducibility, ensuring that other researchers can easily replicate, build upon, and validate our experiments. We implemented the following strategies:

1. **Setup Function**: We created a `setup_venv.sh` script to automate the setup of the virtual environment and package installation, ensuring quick and consistent environment setup across systems without dependency conflicts.

2. **Usage Messages**: The `predict.py` and `evaluate.py` scripts feature clear usage messages that explain the required arguments, available options, and how to specify models, tasks, and data sources. This helps users understand how to run the experiments and reproduce results.

3. **Clear Documentation**: We provide comprehensive documentation detailing how to prepare data, run predictions, and evaluate results. It includes examples and explanations of configurations, making the project accessible for researchers with various levels of expertise.

4. **Inclusion of Data Sources**: The project uses publicly available datasets, such as `google-research-datasets/poem_sentiment` from Hugging Face, which is automatically processed through the `make_data.py` script. This ensures that users can easily access and

use the same data for replication without manual preprocessing.

Throughout the project, we encountered difficulties in emulating the experiments from existing research due to inadequate details in many papers. This motivated us to focus on reproducibility, ensuring that all aspects of the experiment, from setup to data access, are clear and automated. We believe this effort contributes to advancing NLP research by promoting transparency and enabling others to build on our work.

3 Results/Analyses

Results are displayed in table 5, 6, and 7.

Below are the results gathered from implementing the four prompting techniques on the three datasets: poem, tweet, yelp. This chart shows the weighted average row across all of the classes, but we have added all of the results in the appendix as well for clarity. An interesting detail we noticed was that Flant5 large and GPT-3.5 performed the best on precision, while GPT-4o-mini has the best written explanations. Altough it doesn't guess the label correctly, GPT-40-mini was able to provide a meaningful justification for it's prediction. Semantic texts are subjective by nature, which explains why models like GPT might make more creative assumptions about emotion compared to Flan-T5. This may be why the smaller models performed better than the two gpt models. For the poem sentiment, all of the models performed average– none of them were able to give an F1 score over .5. A higher F1 score is better. The F1 score is the harmonic mean of precision and recall, and it balances the trade-off between them. A higher F1 score indicates better overall performance in terms of both precision (how many selected items are relevant) and recall (how many relevant items are selected). The F1 score ranges from 0 to 1, with 1 being the best possible score.

Additionally, for the Tweet tests, the model did significantly better than the poem sentiment. All four models did around the same, but the best F1 results came from flant-5 using meta prompting. It seems that overall the smaller models continued on the trend of doing better than the larger models.

Additionally, for Yelp, the datset doing better accuracy overall because its data was super obviously negative or positive, using curse words and powered adjectives. This likely made it easier to classify, and for some of the models it even got a 1.00 precision which was really good. Overall, the smaller models still did better than the larger models. These values were all extremely high, aide from the chain of thought for flant5 small. This may be because flant5 is not tuned for prompting like that.

4 Related Works

4.0.1 Prompt Engineering and In-Context Learning

The paper, "Automatic Chain of Thought Prompting in Large Language Models" (Zhang et al., 2022) compared zero-shot chain of prompting (where you just write "Let's think step by step") and manual chain of thought prompting (where you provide a few examples showing how to break down a type of problem) on a math word-problem database. It aimed to show that diversity in chain of thought and providing a breakdown of the rationale can improve results. It showed that by initially clustering the data and then retrieving the few-shot example prompts based on similarity (a technique called Automatic chain of thought) they were able to match or exceed performance of having to manually find similar examples. This paper inspired the zero shot and meta prompting used in our experiment.
The paper "Language Models are Few-Shot Learners" (Brown et al., 2020) highlights that GPT-3, when applied to various tasks, performs well without requiring any gradient updates or fine-tuning. Tasks are specified purely through text interaction, with examples provided in a few-shot manner. GPT-3 demonstrates strong performance across tasks like translation, question-answering, and including word unscrambling. The paper also discusses how bias present in these models can influence the results, emphasizing the need to be aware of potential biases in model behavior and performance. Our paper is similar because we also used GPT 3 without any tuning and used the different prompting types to explore how the model reacts.
This paper "Text classification via Large Language Models" (Sun et al., 2023) explores a method to try and make LLM's better at text classification problems using a method called Clue And Reasoning Prompting (CARP). CARP is a way of prompting that first tries to use keywords, tones, semantic relations, and references, and then uses this for final decisions. This method worked well when tested on Rotten Tomatoes and Yelp data. This

		precision	**recall**	**f1-score**
flan-t5-small	zero-shot	0.22	0.35	0.21
	few-shot	0.22	0.23	0.21
	chain-of-thought	0.25	0.24	0.09
	meta-prompting	0.11	0.27	0.11
flan-t5-large	zero-shot	0.43	0.74	0.54
	few-shot	0.22	0.23	0.21
	chain-of-thought	0.25	0.24	0.09
	meta-prompting	0.11	0.27	0.11
gpt-4o-mini	zero-shot	0.27	0.28	0.27
	few-shot	0.29	0.30	0.28
	chain-of-thought	0.28	0.29	0.27
	meta-prompting	0.04	0.25	0.07
gpt-3.5	zero-shot	0.52	0.44	0.40
	few-shot	0.26	0.26	0.23
	chain-of-thought	0.30	0.30	0.27
	meta-prompting	0.19	0.27	0.22

Table 5: Poem Sentiment

		precision	**recall**	**f1-score**
flan-t5-small	zero-shot	0.57	0.41	0.32
	few-shot	0.36	0.40	0.37
	chain-of-thought	0.40	0.46	0.33
	meta-prompting	0.48	0.46	0.29
flan-t5-large	zero-shot	0.28	0.40	0.33
	few-shot	0.46	0.27	0.26
	chain-of-thought	0.34	0.31	0.28
	meta-prompting	0.29	0.40	0.34
gpt-4o-mini	zero-shot	0.29	0.27	0.28
	few-shot	0.36	0.30	0.32
	chain-of-thought	0.28	0.26	0.26
	meta-prompting	0.20	0.45	0.28
gpt-3.5	zero-shot	0.32	0.31	0.30
	few-shot	0.39	0.22	0.28
	chain-of-thought	0.25	0.26	0.25
	meta-prompting	0.20	0.45	0.28

Table 6: Tweet Sentiment

		precision	**recall**	**f1-score**
flan-t5-small	zero-shot	0.93	0.93	0.93
	few-shot	0.25	0.22	0.18
	chain-of-thought	0.18	0.31	0.23
	meta-prompting	0.74	0.56	0.47
flan-t5-large	zero-shot	0.99	0.82	0.89
	few-shot	0.99	0.90	0.94
	chain-of-thought	0.99	0.94	0.96
	meta-prompting	0.99	0.91	0.95
gpt-4o-mini	zero-shot	1.00	0.65	0.79
	few-shot	1.00	0.71	0.83
	chain-of-thought	1.00	0.76	0.86
	meta-prompting	0.33	0.57	0.42
gpt-3.5	zero-shot	1.00	0.81	0.89
	few-shot	0.95	0.78	0.86
	chain-of-thought	0.65	0.54	0.59
	meta-prompting	0.33	0.57	0.42

Table 7: Yelp Sentiment

experiment tested if out of domain training effects performance. It found that the supervised model did worse when being tested on a different domain, while the LLM was able to adapt and not have as much of a decrease in accuracy when the domain was changed. Our experiment will also be testing how the models compare with different domains, and this paper was a good introduction on how to make LLM's work well for classification.
In "A Survey of In Context Learning", (Dong et al., 2024) authors noted that in-context learning can reduce cost to run models and make them faster since the training step will not be as computationally heavy and parameters do not need to updated for each new problem set. However, it also noted that this type of learning may be difficult due to many LLM having a token limit for the input. Additionally, for languages that might not have as many examples or sparse datasets to provide examples it can prove difficult to apply this method. Our experiment takes this notion– editing the prompt rather than fine tuning- and uses it to form the basis of our experiment.

4.0.2 Sentiment Analysis

"Sentiment Analysis in the Era of Large Language Models: A Reality Check" (Zhang et al., 2023) was a paper thatwas trying to solve a very similar problem as we are, comparing sentiment analysis with smaller models and LLMs. In this paper they also compared different types of few shot learning (1,5,10 examples). They found that the LLMs did outperform the smaller models, but it was not by that much. Additionally, they did note that in some cases, where tasks requiring structured sentiment output LLMs actually performed worse than small models like T5 in both automatic and human evaluations. Our experiment expanded this to include poetry data in order to add another layer of complexity and see how models would compare.
The paper "Comprehensive Study on Sentiment Analysis: From Rule based to modern LLMs" (Gupta et al., 2024) was a recent paper discussing the benefits and challenges of using LLM's for sentiment analysis. With improved contextual understanding, LLM's provided stronger results when it comes to classifying text based on sentiment compared to traditional approaches. Additionally, it highlights that LLM's may have better cross-linguistic or cross-cultural analysis. This paper was extremely helpful in understanding at a high level how LLM's understand semantics and why they might be better than a simpler model.
We initially found the paper, "Journalists, Emotions, and the Introduction of Generative AI Chatbots (Lewis et al., 2024) when exploring how our experiment might test with other datasets like TweetEval. This paper highlighted how AI has impacted the language used in tweets, mainly highlighting two key findings. First, there was a large increase in words relating to AI (models, natural language processing, etc). Additionally, there were some words that seemed to spike in popularity after ChatGPT was introduced. This was an interesting paper that helped better understand and explore the other datasets we were testing with, especially because the paper also tracked how emotional sentiment had shifted over time.
Similar to the last paper, "Linguistic Landscape of Generative AI Perception: A Global Twitter Analysis Across 14 Languages" (Murayama et al., 2024) also analyzed different tweets and aimed to form an understanding of how language differences may have shaped sentiment. While our experiment did not test different languages, this paper raised interesting points of how language as a whole can impact the data, and how future considerations of our project could expand to include other languages. Additionally, the paper raised how local events may still affect Tweet sentiment in other languages, which is an interesting consideration to keep in mind as we use the tweet data in our project.

5 Conclusion

All in all, the goal of the project was to systematically explore the differences between prompting techniques (zero-shot, few-shot, chain-of-thought, meta) across different datasets (poetry, tweets, Yelp reviews) and different types of models (Flan-T5 and GPT). We wanted to understand how these prompting methods influence the performance of language models in sentiment classification tasks.

Based on our evaluation values, we found that zero-shot prompting did better than expected and outperformed the other techniques in accuracy, especially in poetry sentiment classification. This contradicts the hypothesis that more complex prompting techniques like meta-prompting are inherently better for all tasks. The simplicity of zero-shot may have aligned better with a direct and minimal task like sentiment classification. Additionally, the Flan-T5 models outperformed the GPT models despite being smaller in scale, demonstrating that model size is not always the sole determinant of performance. Our findings highlight the importance of task-specific optimization when it comes to prompting techniques and models.

On a different note, our project lacked dataset scope as it was only conducted on English data, resulting in a missing representation of languages, cultures, and domains. Although our models, Flan-T5 and GPT provided useful contrast, it does not account for the diversity of the thousands of existing models. Furthermore, our code was mainly implemented on static prompts and datasets, making the scalability of findings difficult. Because of this, a future direction would include enhancing our code and incorporating more variations of data and model.

6 Contribution Statement

We had multiple coding sessions where we all joined a zoom call and coded the pipeline together. Additionally, Kyra ran the models on her laptop and wrote the experiment sections, Anum wrote the related works and datasets portion, and Tramy wrote the models and introductions. The rest of the sections were evenly distrubuted between all of the group members.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *Preprint*, arXiv:2210.11416.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A survey on in-context learning. *Preprint*, arXiv:2301.00234.

Shailja Gupta, Rajesh Ranjan, and Surya Narayan Singh. 2024. Comprehensive study on sentiment analysis: From rule-based to modern llm based system. *Preprint*, arXiv:2409.09989.

Jeremiah. 2024. How developers can use gpt-3.5 turbo to be more productive.

Seth C. Lewis, David M. Markowitz, and Jon Benedik Bunquin. 2024. Journalists, emotions, and the introduction of generative ai chatbots: A large-scale analysis of tweets before and after the launch of chatgpt. *Preprint*, arXiv:2409.08761.

Fabio Matricardi. 2024. Past, present, perfect: Flan-t5 stands the test of time.

Taichi Murayama, Kunihiro Miyazaki, Yasuko Matsubara, and Yasushi Sakurai. 2024. Linguistic landscape of generative ai perception: A global twitter analysis across 14 languages. *Preprint*, arXiv:2405.20037.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)*, pages 502–518.

Sander Schulhoff, Michael Ilie, Nishant Balepur, Konstantine Kahadze, Amanda Liu, Chenglei Si, Yinheng Li, Aayush Gupta, HyoJung Han, Sevien Schulhoff, Pranav Sandeep Dulepet, Saurav Vidyadhara, Dayeon Ki, Sweta Agrawal, Chau Pham, Gerson Kroiz, Feileen Li, Hudson Tao, Ashay Srivastava, Hevander Da Costa, Saloni Gupta, Megan L. Rogers, Inna Goncearenco, Giuseppe Sarli, Igor Galynker, Denis Peskoff, Marine Carpuat, Jules White, Shyamal Anadkat, Alexander Hoyle, and Philip Resnik. 2024. The prompt report: A systematic survey of prompting techniques. *Preprint*, arXiv:2406.06608.

Deval Shah. 2023. What is in-context learning, and how does it work: The beginner's guide. *Lakera.AI*.

Emily Sheng and David Uthus. 2020. Investigating societal biases in a poetry composition system. *Preprint*, arXiv:2011.02686.

Austin Starks. 2024. Gpt-4o mini vs gpt-3.5 turbo. i just tried out the new model and am beyond impressed.

Xiaofei Sun, Xiaoya Li, Jiwei Li, Fei Wu, Shangwei Guo, Tianwei Zhang, and Guoyin Wang. 2023. Text classification via large language models. *Preprint*, arXiv:2305.08377.

Wenxuan Zhang, Yue Deng, Bing Liu, Sinno Jialin Pan, and Lidong Bing. 2023. Sentiment analysis in the era of large language models: A reality check. *Preprint*, arXiv:2305.15005.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. *arXiv:1509.01626 [cs]*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *Preprint*, arXiv:2210.03493.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. *Preprint*, arXiv:2102.09690.

A Example Appendix

Model Evaluations

A.1 Poem Sentiment

Model: flan-t5-small, Prompting Style: Zero-shot

Class	Precision	Recall	F1-Score
0	0.19	1.00	0.32
1	0.68	0.41	0.51
2	0.00	0.00	0.00
3	0.00	0.00	0.00
Macro avg	0.22	0.35	0.21
Weighted avg	0.14	0.24	0.13

Accuracy: 0.24
F1 Score: 0.2091

Model: flan-t5-large, Prompting Style: Zero-shot

Class	Precision	Recall	F1-Score
0	0.43	0.74	0.54
1	0.24	0.96	0.38
2	0.80	0.01	0.01
3	0.02	0.04	0.03
Macro avg	0.37	0.44	0.24
Weighted avg	0.61	0.28	0.16

Accuracy: 0.28
F1 Score: 0.2412

Model: flan-t5-small, Prompting Style: Few-shot

Class	Precision	Recall	F1-Score
0	0.10	0.05	0.07
1	0.12	0.27	0.16
2	0.65	0.59	0.62
3	0.00	0.00	0.00
Macro avg	0.22	0.23	0.21
Weighted avg	0.44	0.42	0.42

Accuracy: 0.42
F1 Score: 0.2128

Model: flan-t5-large, Prompting Style: Few-shot

Class	Precision	Recall	F1-Score
0	0.43	0.82	0.56
1	0.30	0.87	0.44
2	0.91	0.03	0.06
3	0.02	0.10	0.04
Macro avg	0.42	0.46	0.28
Weighted avg	0.69	0.30	0.21

Accuracy: 0.30
F1 Score: 0.2781

Model: flan-t5-small, Prompting Style: Chain-of-thought

Class	Precision	Recall	F1-Score
0	0.17	0.03	0.05
1	0.15	0.91	0.25
2	0.68	0.03	0.06
3	0.00	0.00	0.00
Macro avg	0.25	0.24	0.09
Weighted avg	0.48	0.16	0.08

Accuracy: 0.16
F1 Score: 0.0896

Model: flan-t5-large, Prompting Style: Chain-of-thought

Class	Precision	Recall	F1-Score
0	0.42	0.88	0.57
1	0.34	0.94	0.50
2	0.94	0.16	0.28
3	0.03	0.06	0.04
Macro avg	0.43	0.51	0.35
Weighted avg	0.71	0.40	0.35

Accuracy: 0.40
F1 Score: 0.3455

Model: flan-t5-small, Prompting Style: Meta

Class	Precision	Recall	F1-Score
0	0.30	0.11	0.16
1	0.16	0.97	0.27
2	0.00	0.00	0.00
3	0.00	0.00	0.00
Macro avg	0.11	0.27	0.11
Weighted avg	0.08	0.16	0.07

Accuracy: 0.16
F1 Score: 0.1069

Model: flan-t5-large, Prompting Style: Meta

Class	Precision	Recall	F1-Score
0	0.57	0.75	0.65
1	0.26	0.97	0.42
2	0.67	0.01	0.01
3	0.03	0.14	0.05
Macro avg	0.38	0.47	0.28
Weighted avg	0.56	0.29	0.19

Accuracy: 0.29
F1 Score: 0.2827

Model: gpt-4o-mini, Prompting Style: Zero-shot

Class	Precision	Recall	F1-Score
0	0.20	0.26	0.22
1	0.21	0.26	0.24
2	0.64	0.57	0.60
3	0.03	0.02	0.03
Macro avg	0.27	0.28	0.27
Weighted avg	0.47	0.44	0.45

Accuracy: 0.44
F1 Score: 0.2722

Model: gpt-3.5-turbo, Prompting Style: Zero-shot

Class	Precision	Recall	F1-Score
0	0.74	0.31	0.44
1	0.35	0.90	0.50
2	0.95	0.24	0.38
3	0.06	0.43	0.10
Macro avg	0.52	0.47	0.35
Weighted avg	0.78	0.36	0.40

Accuracy: 0.36
F1 Score: 0.3545

Model: gpt-4o-mini, Prompting Style: Few-shot

Class	Precision	Recall	F1-Score
0	0.20	0.32	0.24
1	0.20	0.31	0.24
2	0.64	0.43	0.52
3	0.11	0.14	0.12
Macro avg	0.29	0.30	0.28
Weighted avg	0.47	0.38	0.41

Accuracy: 0.38
F1 Score: 0.2805

Model: gpt-3.5-turbo, Prompting Style: Few-shot

Class	Precision	Recall	F1-Score
0	0.20	0.06	0.10
1	0.16	0.38	0.22
2	0.61	0.42	0.50
3	0.06	0.18	0.09
Macro avg	0.26	0.26	0.23
Weighted avg	0.44	0.34	0.37

Accuracy: 0.34
F1 Score: 0.2268

Model: gpt-4o-mini, Prompting Style: Chain-of-thought

Class	Precision	Recall	F1-Score
0	0.21	0.31	0.25
1	0.18	0.30	0.22
2	0.65	0.44	0.53
3	0.08	0.10	0.09
Macro avg	0.28	0.29	0.27
Weighted avg	0.47	0.38	0.41

Accuracy: 0.38
F1 Score: 0.2721

Model: gpt-3.5-turbo, Prompting Style: Chain-of-thought

Class	Precision	Recall	F1-Score
0	0.25	0.21	0.23
1	0.17	0.48	0.25
2	0.68	0.40	0.51
3	0.08	0.10	0.09
Macro avg	0.30	0.30	0.27
Weighted avg	0.50	0.37	0.40

Accuracy: 0.37
F1 Score: 0.2689

Model: gpt-4o-mini, Prompting Style: Meta

Class	Precision	Recall	F1-Score
0	0.00	0.00	0.00
1	0.15	1.00	0.26
2	0.00	0.00	0.00
3	0.00	0.00	0.00
Macro avg	0.04	0.25	0.07
Weighted avg	0.02	0.15	0.04

Accuracy: 0.15
F1 Score: 0.0653

Model: gpt-3.5-turbo, Prompting Style: Meta

Class	Precision	Recall	F1-Score
0	1.00	0.01	0.01
1	0.15	1.00	0.26
2	1.00	0.00	0.01
3	0.00	0.00	0.00
Macro avg	0.54	0.25	0.07
Weighted avg	0.82	0.15	0.05

Accuracy: 0.15
F1 Score: 0.0700

A.1.1 Tweets

Model: flan-t5-small, , Setting: zero-shot

Class	Precision	Recall	F1-Score
0	0.26	0.97	0.41
1	0.60	0.01	0.01
2	0.69	0.63	0.66
Macro avg	0.52	0.54	0.36
Weighted avg	0.57	0.41	0.32

Accuracy: 0.41
F1 Score: 0.3605

Model: flan-t5-large, , Setting: zero-shot

Class	Precision	Recall	F1-Score
0	0.49	0.83	0.62
1	0.43	0.57	0.49
3	0.00	0.00	0.00
Macro avg	0.23	0.35	0.28
Weighted avg	0.28	0.40	0.33

Accuracy: 0.40
F1 Score: 0.2769

Model: flan-t5-small, , Setting: few-shot

Class	Precision	Recall	F1-Score
0	0.12	0.02	0.03
1	0.43	0.43	0.43
2	0.39	0.55	0.45
Macro avg	0.31	0.33	0.31
Weighted avg	0.36	0.40	0.37

Accuracy: 0.40
F1 Score: 0.3056

Model: flan-t5-large, , Setting: few-shot

Class	Precision	Recall	F1-Score
0.0	0.17	0.24	0.20
1.0	0.46	0.49	0.48
2.0	0.60	0.01	0.02
Macro avg	0.31	0.19	0.17
Weighted avg	0.46	0.27	0.26

Accuracy: 0.27
F1 Score: 0.1725

Model: flan-t5-small, , Setting: chain-of-thought

Class	Precision	Recall	F1-Score
0	0.00	0.00	0.00
1	0.46	0.96	0.62
2	0.52	0.07	0.13
Macro avg	0.33	0.34	0.25
Weighted avg	0.40	0.46	0.33

Accuracy: 0.46
F1 Score: 0.2494

Model: flan-t5-large, , Setting: chain-of-thought

Class	Precision	Recall	F1-Score
0.0	0.18	0.30	0.23
1.0	0.45	0.56	0.50
2.0	0.28	0.01	0.03
Macro avg	0.23	0.22	0.19
Weighted avg	0.34	0.31	0.28

Accuracy: 0.31
F1 Score: 0.1886

Model: flan-t5-small, , Setting: meta

Class	Precision	Recall	F1-Score
0	0.50	0.01	0.02
1	0.46	1.00	0.63
2	0.50	0.00	0.01
Macro avg	0.36	0.25	0.16
Weighted avg	0.48	0.46	0.29

Accuracy: 0.46
F1 Score: 0.1634

Model: flan-t5-large, , Setting: meta

Class	Precision	Recall	F1-Score
0.0	0.55	0.74	0.63
1.0	0.43	0.58	0.50
2.0	0.00	0.00	0.00
Macro avg	0.25	0.33	0.28
Weighted avg	0.29	0.40	0.34

Accuracy: 0.40
F1 Score: 0.2824

Model: gpt-4o-mini, Prompting Style: Zero-shot

Class	Precision	Recall	F1-Score
0	0.66	0.67	0.66
1	0.25	0.21	0.23
2	0.17	0.14	0.15
Macro avg	0.27	0.26	0.26
Weighted avg	0.29	0.27	0.28

Accuracy: 0.27
F1 Score: 0.2629

Model: gpt-3.5-turbo, Prompting Style: Zero-shot

Class	Precision	Recall	F1-Score
0	0.69	0.46	0.55
1	0.39	0.50	0.44
2	0.06	0.01	0.01
Macro avg	0.28	0.24	0.25
Weighted avg	0.32	0.31	0.30

Accuracy: 0.31
F1 Score: 0.2511

Model: gpt-4o-mini, Prompting Style: Few-shot

Class	Precision	Recall	F1-Score
0	0.42	0.45	0.44
1	0.38	0.31	0.34
2	0.31	0.21	0.25
Macro avg	0.12	0.11	0.11
Weighted avg	0.36	0.30	0.32

Accuracy: 0.30
F1 Score: 0.1139

Model: gpt-3.5-turbo, Prompting Style: Few-shot

Class	Precision	Recall	F1-Score
0.0	0.19	0.16	0.17
1.0	0.46	0.23	0.31
2.0	0.39	0.23	0.29
Macro avg	0.26	0.15	0.19
Weighted avg	0.39	0.22	0.28

Accuracy: 0.22
F1 Score: 0.1919

Model: gpt-4o-mini, Prompting Style: Chain-of-thought

Class	Precision	Recall	F1-Score
0	0.55	0.73	0.63
1	0.28	0.27	0.27
2	0.15	0.07	0.10
Macro avg	0.16	0.18	0.17
Weighted avg	0.27	0.26	0.26

Accuracy: 0.26
F1 Score: 0.1669

Model: gpt-3.5-turbo, Prompting Style: Chain-of-thought

Class	Precision	Recall	F1-Score
0	0.58	0.73	0.65
1	0.29	0.31	0.30
2	0.08	0.03	0.04
Macro avg	0.24	0.27	0.25
Weighted avg	0.25	0.26	0.25

Accuracy: 0.26
F1 Score: 0.2465

Model: gpt-4o-mini, Prompting Style: Meta

Class	Precision	Recall	F1-Score
0	0.00	0.00	0.00
1	0.45	1.00	0.62
2	0.00	0.00	0.00
Accuracy			0.45
Macro avg	0.15	0.33	0.21
Weighted avg	0.20	0.45	0.28

Accuracy: 0.45
F1 Score: 0.2069

Model: gpt-3.5-turbo, Prompting Style: Meta

Class	Precision	Recall	F1-Score
0	0.00	0.00	0.00
1	0.45	1.00	0.62
2	0.00	0.00	0.00
Accuracy			0.45
Macro avg	0.15	0.33	0.21
Weighted avg	0.20	0.45	0.28

Accuracy: 0.45
F1 Score: 0.2069

A.1.2 Yelp

Model: flan-t5-small, Setting: Zero-Shot

Class	Precision	Recall	F1-Score
0	0.90	0.97	0.94
1	0.97	0.88	0.92
Macro avg	0.93	0.93	0.93
Weighted avg	0.93	0.93	0.93

Accuracy: 0.93
F1 Score: 0.9293

Model: flan-t5-large, Setting: Zero-Shot

Class	Precision	Recall	F1-Score
0	0.99	0.92	0.95
1	0.99	0.72	0.83
Macro avg	0.50	0.41	0.45
Weighted avg	0.99	0.82	0.89

Accuracy: 0.82
F1 Score: 0.4460

Model: flan-t5-small, Setting: Few-Shot

Class	Precision	Recall	F1-Score
0	0.20	0.00	0.01
1	0.30	0.47	0.37
Macro avg	0.08	0.08	0.06
Weighted avg	0.25	0.22	0.18

Accuracy: 0.22
F1 Score: 0.0623

Model: flan-t5-large, Setting: Few-Shot

Class	Precision	Recall	F1-Score
0	0.99	0.91	0.95
1	0.98	0.88	0.93
Macro avg	0.49	0.45	0.47
Weighted avg	0.99	0.90	0.94

Accuracy: 0.90
F1 Score: 0.4698

Model: flan-t5-small, Setting: Chain-of-Thought

Class	Precision	Recall	F1-Score
0	0.00	0.00	0.00
1	0.38	0.66	0.48
Macro avg	0.06	0.11	0.08
Weighted avg	0.18	0.31	0.23

Accuracy: 0.31
F1 Score: 0.0798

Model: flan-t5-large, Setting: Chain-of-Thought

Class	Precision	Recall	F1-Score
0	0.99	0.93	0.96
1	0.99	0.94	0.96
Macro avg	0.49	0.47	0.48
Weighted avg	0.99	0.94	0.96

Accuracy: 0.94
F1 Score: 0.4810

Model: flan-t5-small, Setting: Meta

Class	Precision	Recall	F1-Score
0	0.95	0.16	0.28
1	0.52	0.99	0.68
Macro avg	0.49	0.38	0.32
Weighted avg	0.74	0.56	0.47

Accuracy: 0.56
F1 Score: 0.3206

Model: flan-t5-large, Setting: Meta

Class	Precision	Recall	F1-Score
0	0.99	0.93	0.96
1	0.98	0.90	0.94
Macro avg	0.66	0.61	0.63
Weighted avg	0.99	0.91	0.95

Accuracy: 0.91
F1 Score: 0.6324
article amsmath

Model: gpt-4o-mini, Setting: Zero-Shot

Class	Precision	Recall	F1-Score
0	1.00	0.66	0.79
1	1.00	0.65	0.79
Macro avg	0.67	0.44	0.53
Weighted avg	1.00	0.65	0.79

Accuracy: 0.65
F1 Score: 0.5271

Model: gpt-3.5-turbo, Setting: Zero-Shot

Class	Precision	Recall	F1-Score
0	1.00	0.91	0.95
1	1.00	0.74	0.85
Macro avg	0.67	0.55	0.60
Weighted avg	1.00	0.81	0.89

Accuracy: 0.81
F1 Score: 0.6014

Model: gpt-4o-mini, Setting: Few-Shot

Class	Precision	Recall	F1-Score
0	1.00	0.66	0.79
1	1.00	0.74	0.85
Macro avg	0.67	0.47	0.55
Weighted avg	1.00	0.71	0.83

Accuracy: 0.71
F1 Score: 0.5486

Model: gpt-3.5-turbo, Setting: Few-Shot

Class	Precision	Recall	F1-Score
0.0	0.96	0.84	0.90
1.0	0.94	0.74	0.83
Macro avg	0.63	0.53	0.58
Weighted avg	0.95	0.78	0.86

Accuracy: 0.78
F1 Score: 0.5756

Model: gpt-4o-mini, Setting: Chain-of-Thought

Class	Precision	Recall	F1-Score
0	1.00	0.78	0.88
1	1.00	0.74	0.85
Macro avg	0.67	0.51	0.58
Weighted avg	1.00	0.76	0.86

Accuracy: 0.76
F1 Score: 0.5768

Model: gpt-3.5-turbo, Setting: Chain-of-Thought

Class	Precision	Recall	F1-Score
0.0	0.64	0.56	0.60
1.0	0.65	0.52	0.58
Macro avg	0.43	0.36	0.39
Weighted avg	0.65	0.54	0.59

Accuracy: 0.54
F1 Score: 0.3930

Model: gpt-4o-mini, Setting: Meta

Class	Precision	Recall	F1-Score
0	0.00	0.00	0.00
1	0.57	1.00	0.73
Macro avg	0.29	0.50	0.36
Weighted avg	0.33	0.57	0.42

Accuracy: 0.57
F1 Score: 0.3644

Model: gpt-3.5-turbo, Setting: Meta

Class	Precision	Recall	F1-Score
0	0.00	0.00	0.00
1	0.57	1.00	0.73
Macro avg	0.29	0.50	0.36
Weighted avg	0.33	0.57	0.42

Accuracy: 0.57
F1 Score: 0.3644