Assignment 3

Saujanya Acharya Eric Ohemeng Bozhou(Peter) Tan

2024-04-08

Introduction

In this project, we use the Reddit comments collected in Assignment 2 and apply machine learning (ML) techniques and Large Language Models (LLMs) to analyze the stance of Reddit posts. In th first part is conducted in R, we begin from the basic steps of text pre-processing, including removing punctuation, stop words and removing rarely used words. We then tokenize the text and create a document-term matrix. We use several different models to predict the stance of the comments, including K-Nearest Neighbors, Neural Network and XGBoost Model. We evaluate the performance of these models using accuracy, sensitivity and specificity. In the second part, we use the Large Language Model (LLM) to anaylze the stance of the Reddit comments. We use GreatLakes HPC cluster to run the LLM. All data and code are available in the GitHub repository.

Machine Learning Models in R

We begin the analysis by importing the cleaned data in Assignment 3. All comments come from the subreddit r/economy from March 16th to March 23rd, 2024. There are 838 relevent comments in the dataset with labels of "Favor", "Opposed" and "Neutral" after hand coding by each member. We then tokenize the text and remove stop words. The most frequent words in the comments are shown in Figure 1. The figure shows that the most frequent words are "people", "tax", "rates", "market", "home", "housing", "prices", "economy", "money" and "inflation". They cover almost all main topics in the economy and indicate people's concerns when evaluating the current economic situation. We also remove the words that appear only once in the comments to reduce the dimensionality of the data, for infrequent words are less likely to be useful in predicting the stance of the comments. The distribution of the number of words in the comments after removing infrequent words is shown in Figure 2. The distribution is right-skewed, with most words appearing less than 10 times in the comments.

After the descriptive analysis, we then make more text processing to prepare the data for the machine learning models. We make all the letters lowercase and get rid of numbers and punctuation. Stopwords are also excluded in the dataset, for they are common words that do not carry much meaning. We then create a matrix that keeps track of how often each cleaned-up word appears in

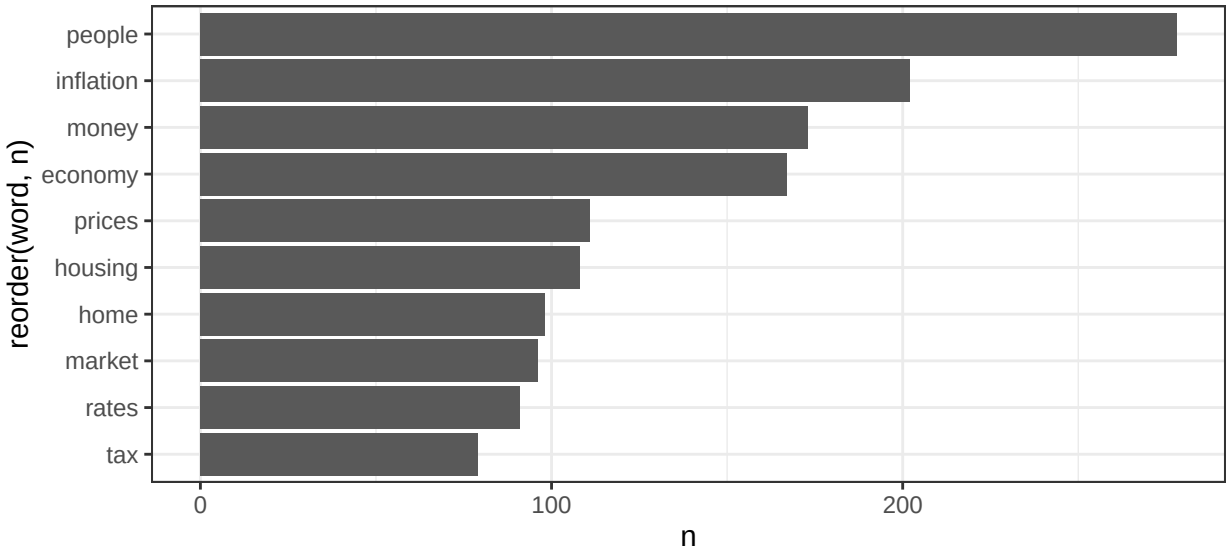

Figure 1: Most frequent words in the comments

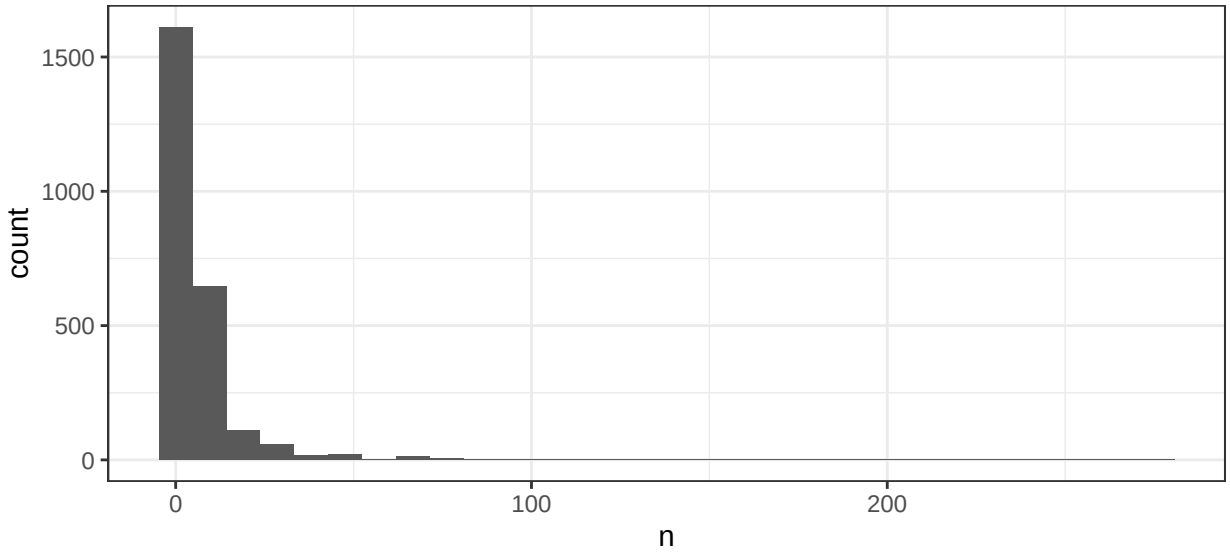

Figure 2: Distribution of the number of words in the comments after removing infrequent words

each comment. We make this table more manageable by removing words that hardly ever show up at 0.99 level. We split the data into training and testing sets with a ratio of 0.8. We use train data with 10 fold validation to estimate different machine learning models and use the test data to evaluate the performance of the models. There are 672 cases in the training set and 166 cases in the testing set.

We first use the K-Nearest Neighbors (KNN) model to predict the stance of the comments. We use the caret package to tune the hyperparameter of the KNN model. We set the K-value from 3 to 9 to test the evaluation performance of the model. The accuracy of the KNN model with different K values is shown in Table 1. The accuracy of the model is highest when K = 9, with an accuracy of 0.578. However, the acuracy from K = 3 to K = 9 is very close, with a difference of 0.03 and there is some fluctuation in the accuracy. Table 2 to Table 4 provide a comprehensive overview of the performance of the KNN model with varying values of K, focusing on sensitivity, specificity, precision, and recall for each class. In Table 2 (K = 3), the model demonstrates varying performance across classes, with particularly high sensitivity for class "Opposed" but lower values for "Favor" and "Neutral". However, specificity is notably higher for classes "Favor" and "Neutral", indicating better identification of true negative cases. Table 3 (K = 6) showcases a stark decline in sensitivity across all classes, with specificity generally remaining high, suggesting a trade-off between sensitivity and specificity as K increases. Conversely, Table 4 (K = 9) highlights a similar trend of low sensitivity, especially for classes "Favor" and "Neutral", alongside extremely high specificity, implying a tendency to correctly identify true negative cases but at the expense of missing positive ones. Overall, the models are better at predicting "Opposed" than "Favor" and "Neutral".

Table 1: Accuracy of KNN model with different K values

K	accuracy
3	0.548
4	0.554
5	0.566
6	0.572
7	0.566
8	0.572
9	0.578

Table 2: Performance of KNN model with K = 3

	Sensitivity	Specificity	Precision	Recall
Class: Favor	0.111	0.914	0.200	0.111
Class: Neutral	0.136	0.926	0.400	0.136
Class: Opposed	0.863	0.239	0.603	0.863

Table 3: Performance of KNN model with K = 6

	Sensitivity	Specificity	Precision	Recall
Class: Favor	0.037	0.978	0.250	0.037
Class: Neutral	0.023	0.967	0.200	0.023
Class: Opposed	0.979	0.099	0.592	0.979

Table 4: Performance of KNN model with K = 9

	Sensitivity	Specificity	Precision	Recall
Class: Favor	0.037	0.993	0.500	0.037
Class: Neutral	0.000	1.000	NA	0.000
Class: Opposed	1.000	0.028	0.579	1.000

We also use neural network method to do this multi-class classification. we set size as 1 and use different decay values to tune the model. Tables 5 to 8 provide insights into the performance of a Neural Network model with varying decay values, focusing on accuracy, sensitivity, specificity, precision, and recall for each decay setting. In Table 5, which presents the accuracy of the model with different decay values, it is evident that the highest accuracy of 0.482 is achieved when the decay value is set to 0.1, indicating the impact of decay on the overall model performance. Moving to Tables 6, 7, and 8, which delve into the model's performance at specific decay values, notable trends emerge. For instance, in Table 6 (Decay = 0.001), the model demonstrates moderate sensitivity and specificity across classes, with higher values for class "Neutral" compared to the other classes. In Table 7 (Decay = 0.01), there is an improvement in sensitivity and specificity across all classes compared to the previous decay setting, with relatively balanced performance across classes. Lastly, Table 8 (Decay = 0.1) exhibits a trade-off between sensitivity and specificity, with higher sensitivity observed for class "Favor" and lower sensitivity for class "Opposed". Precision and recall metrics vary across classes but generally reflect the model's ability to correctly identify positive cases and avoid false positives. These tables collectively underscore the importance of tuning the decay parameter to optimize model performance, considering the trade-offs between sensitivity, specificity, precision, and recall for effective classification in the Neural Network model. Overall, the accuracy of the Neural Network model is lower than that of the KNN model, indicating the need for further optimization to enhance predictive performance.

Table 5: Accuracy of Neural Network model with different Decay values

Decay	accuracy
0.001	0.386
0.010	0.380
0.100	0.482

Table 6: Performance of Neural Network model with Decay = 0.001

Sensitivity	Specificity	Precision	Recall
0.407	0.705	0.212	0.407
0.000	1.000	NA	0.000
0.726	0.366	0.605	0.726

Table 7: Performance of Neural Network model with Decay = 0.01

Sensitivity	Specificity	Precision	Recall
0.444	0.698	0.222	0.444
0.364	0.656	0.276	0.364
0.368	0.732	0.648	0.368

Table 8: Performance of Neural Network model with Decay = 0.1

Sensitivity	Specificity	Precision	Recall
0.222	0.763	0.154	0.222
0.136	0.795	0.194	0.136
0.547	0.380	0.542	0.547

The last model we used is the XGBoost. Following the pre-processing procedure listed above, we fine-tuned the XGBoost model across a range of hyperparameters to accommodate our multi-class classification task,: max_depth for control over the complexity of the model, nrounds for the number of boosting iterations, and eta for the learning rate, which affects the contribution of each tree to the final model.

Table 9: Summary of XGBoost Model Performance Metrics

Metric	Class: Favor	Class: Neutral	Class: Opposed
Sensitivity	0.0370	0.0682	0.9579
Specificity	0.9784	0.9508	0.1268
Precision	0.2500	0.3333	0.5948
Recall	0.2500	0.3333	0.5947
Balanced Accuracy	0.5077	0.5095	0.5423
Accuracy	0.5113	0.5013	0.5723

The XGBoost model's performance was evaluated, as shown in the Table 9. The overall accuracy is 57.2%. We found that while the model excelled at identifying the 'Opposed' class with high sensitivity (95.79%), it struggled with the 'Favor' and 'Neutral' classes, with sensitivity scores of 3.70% and 6.82% respectively. Specificity scores were high for 'Favor' and 'Neutral' at 97.84% and 95.08%, indicating a strong ability to identify true negatives.

LLM Model

In our analysis of the Large Language Model's (LLM) annotations of the Reddit posts/comments regarding the stance on the economy's current state versus the past, the Great Lakes High Performance Clusters were utilized to run the LLM model. We decided to select mistralai/Mistral-7B-Instruct-v0.2 model, and created a one-shot prompt design. The select prompt was as follows

> "Instruction: You have assumed the role of a human annotator. In this task, you will be presented with a reddit post/comment, delimited by triple backticks, concerning whether the economy is better now than before. Please make the following assessment:
>
> (1) Determine whether the comment/post discusses the topic of whether the economy is better now than before. If so, please indicate whether the Reddit user who posted the tweet favors, opposes, or is neutral about whether the economy is better now than before. Your response should be formatted as follows and include nothing else: "Stance: [F, O, N]"
>
> For example if comment is "I feel that the average income of US households has improved in the past few years.", then your response should be "Stance: F" and include nothing before or after this response.
>
> Here F stands for Favor, O stands for Oppose and N stands for Neutral.
>
> Post/Comment: *inserted comment/post here*"

Each post/comment was then tokenized and all of them were put into a list to process them with the LLM. Based on the output generated by the model, the stances were extracted from each of the responses and were appended to the cleaned data file for comparison.

To assess the performance of the large language model, several key metrics were then evaluated: accuracy, precision, recall, F1-score, and specificity for each stance category (Favor, Oppose, Neutral) using the scikit-learn pacakage in python.

The model achieved an overall accuracy of only 39%, demonstrating moderate to low effectiveness in correctly aligning with manually annotated stances. Despite a relatively higher precision of 64%, recall matched the accuracy at 39%, pointing towards a significant portion of true stances being either missed or incorrectly categorized. The weighted F1-score of 34% further illustrates

the challenge the model faces in balancing precision and recall, indicating a tendency towards making incorrect predictions or failing to identify true cases. Specificity scores for the Favor, Oppose, and Neutral stances were 0.849, 0.472, and 0.903, respectively, showcasing the model's competence in identifying comments not belonging to certain categories, particularly for Favor and Neutral stances. However, the lower specificity for the Oppose category suggests difficulties in correctly dismissing non-opposing comments, potentially leading to a higher false positive rate for this stance. This analysis underscores the model's current limitations in accurately classifying stances on economic topics, highlighting a substantial need for improvements in understanding context, optimizing training data, or refining classification strategies to enhance its alignment with human judgment.

Table 10: Summary of LLM Performance Metrics

Metric	Value
Accuracy	0.39
Precision (Weighted)	0.64
Recall (Weighted)	0.39
F1-Score (Weighted)	0.34
Specificity (F)	0.849
Specificity (O)	0.472
Specificity (N)	0.903

After the initial analysis of the model we also proceeded to compare the the LLM's performance with the ML models.

Based on the performance metrics – particularly, Accuracy, precision, recall and specificity, the KNN model (with k = 6) showed to outperform both the LLM and Neural Network models in terms of accuracy, achieving an accuracy of 0.572. However, the LLM model shows the highest precision for the 'O' (Opposed) class, with a precision of 0.80, while the Neural Network model demonstrates the highest recall for the 'O' (Opposed) class, with a recall of 0.547. In terms of specificity, the KNN model shows the highest specificity for the 'F' (In Favor) class, with a specificity of 0.978, while the Neural Network model exhibits the highest specificity for the 'N' (Neutral) class, with a specificity of 0.795.

Table 11: Models Performance Metrics

Model	Metric	Value
LLM	Accuracy	0.39
	Precision (F)	0.65
	Precision (O)	0.80
	Precision (N)	0.30

Model	Metric	Value
	Recall (F)	0.11
	Recall (O)	0.21
	Recall (N)	0.94
	Specificity (F)	0.849
	Specificity (O)	0.472
	Specificity (N)	0.903
KNN Model	Accuracy	0.572
	Precision (F)	0.25
	Precision (O)	0.592
	Precision (N)	0.2
	Recall (F)	0.037
	Recall (O)	0.979
	Recall (N)	0.023
	Specificity (F)	0.978
	Specificity (O)	0.099
	Specificity (N)	0.967
Neural Network Model	Accuracy	0.482
	Precision (F)	0.154
	Precision (O)	0.542
	Precision (N)	0.194
	Recall (F)	0.222
	Recall (O)	0.547
	Recall (N)	0.136
	Specificity (F)	0.763
	Specificity (O)	0.380
	Specificity (N)	0.795
XGBoost	Accuracy	0.572
	Precision (F)	0.037
	Precision (O)	0.957
	Precision (N)	0.068
	Recall (F)	0.250
	Recall (O)	0.594
	Recall (N)	0.333
	Specificity (F)	0.839
	Specificity (O)	0.692
	Specificity (N)	0.738

The metrics indicate that all of the models struggle with identifying the “opposing” view. Moreover, the accuracy of the LLM and the Neural Network models are below 50% showing poor performance.

Conclusion

In this study, we explored various techniques in sentiment analysis by implementing different ML models alongside a large language model.

In the analysis we mainly utilized traditional ML algorithms such as K-nearest neighbors, neural networks and XGBoost. The performance metrics obtained from each model provided insights to the strengths and weaknesses in capturing nuances in sentiments expressed through Reddit comments. Leveraging the Great Lakes HPC helped us handle the computing requirements for implementing the LLM model.

Our comparative analysis of ML models has revealed insightful performance disparities across Large Language Models (LLM), K-Nearest Neighbors (KNN), Neural Networks, and XGBoost when applied to the classification of Reddit comments. The LLM exhibited modest accuracy (39%) but showed high precision in identifying favorable comments (65%). It demonstrated remarkable recall for neutral comments (94%), suggesting a strong ability to capture the majority of this sentiment class. The KNN model displayed a balanced performance with the highest overall accuracy (57.2%) among the models evaluated. This model's strength lies in its ability to identify opposed comments, as evidenced by a high recall of 97.9%. The Neural Network model, although not outperforming the KNN model in accuracy (48.2%), provided a more balanced specificity across classes. The XGBoost model matched the KNN model in accuracy (57.2%) and surpassed it in the precision of opposed comments (95.7%). This performance, alongside respectable specificity metrics, underscores XGBoost's capacity for effectively distinguishing between classes without significant overfitting.

While each model has demonstrated unique strengths, the XGBoost and KNN models have emerged as front runners in this analysis, offering promising avenues for accurate classification of economic sentiment on Reddit. The exploration of the techniques using different ML models and LLM underscored the importance of selecting proper approaches based on task specific requirements.