Assignment 3

Saujanya Acharya Eric Ohemeng Bozhou(Peter) Tan

2024-04-07

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 Support Vector Machines. 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.

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

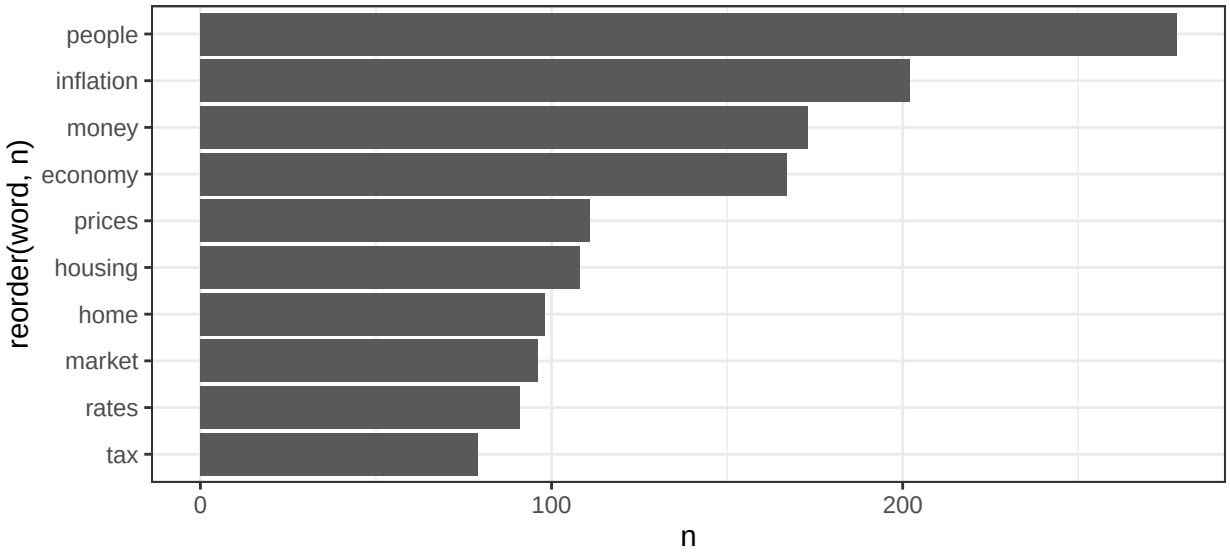

Figure 1: Most frequent words in the comments

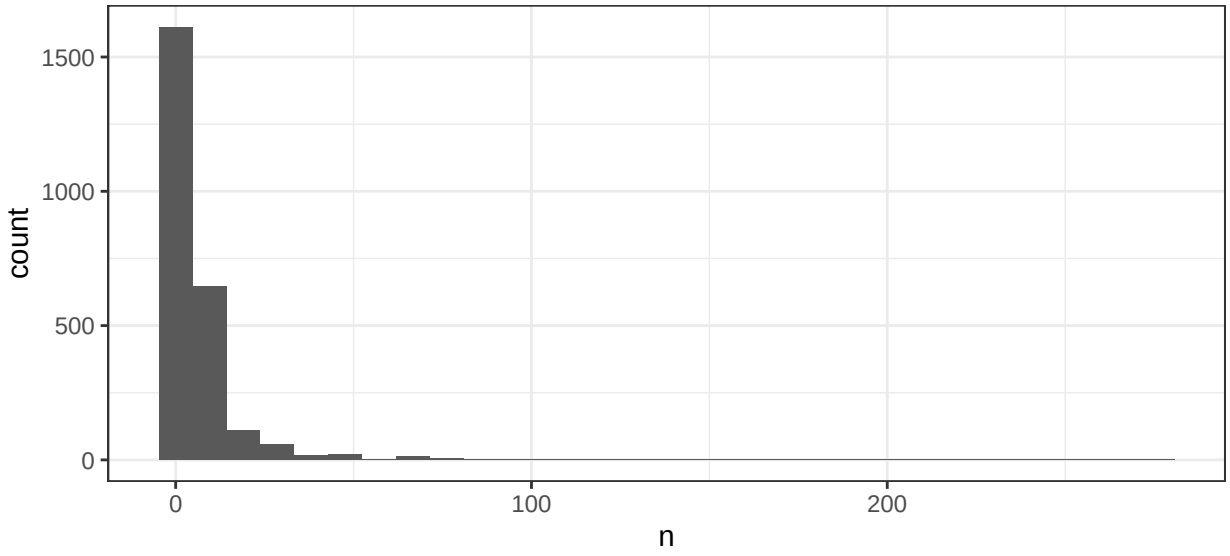

Figure 2: Distribution of the number of words in the comments after removing infrequent words

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

LLM Model

Conclusion