DSC Endsem Project Report

Aditya Gupta
IIITD
Delhi, India
aditya22031@iiitd.ac.in

Sahil Gupta
IIITD
Delhi, India
sahil22430@iiitd.ac.in

Rishabh Jay
IIITD
Delhi, India
rishabh22401@iiitd.ac.in

Avi Sharma
IIITD
Delhi, India
avi22119@iiitd.ac.in

1. Brief overview of the dataset and your problem statement

1.1. Dataset

In this project, we employ two combined datasets: the U.S. 2020 Presidential Election Dataset and a labeled dataset with sentiment annotations.
1) The dataset for the U.S. 2020 Presidential Election has been sourced from Kaggle, containing approximately 900,000 tweets associated with the hashtag #DonaldTrump and 700,000 tweets with the hashtag #JoeBiden (source: Kaggle).This collection captures public discourse around the two primary candidates during the 2020 election cycle.
2) Additionally, to train our model, we obtained a labeled dataset annotated with sentiment information from Kaggle (source: Kaggle). This dataset consists of 1.6 million tweets labeled with corresponding sentiment categories, providing a robust foundation for training sentiment analysis models on social media data .

Therefore, we are utilizing a dataset comprising over one million entries, meeting the required scale for our analysis.

1.2. Problem Statement

In recent years, sentiment analysis has become a crucial tool for understanding public opinion and customer feedback, especially as individuals express their opinions more openly on social media platforms. This research aims to leverage sentiment analysis techniques to compare public sentiments between two major political figures, Donald Trump and Joe Biden, during the 2020 U.S. presidential election. Using a dataset of over 1.6 million annotated tweets and 1.6 million additional tweets tagged with #DonaldTrump and #JoeBiden, this study will employ fine-grained sentiment analysis, emotion detection,and aspect-based sentiment analysis to gauge differences in public sentiment and emotional expressions associated with each candidate.

2. Challenges faced & your solutions while working with the dataset

The various challenges faced during working with the dataset is as follows :

- **Problem:** Missing data in numerical columns like likes and user_followers_count, and categorical columns.

  Solution: Imputed missing numerical values with the mean and categorical values with the mode. Dates were reformatted to ensure temporal consistency.

- **Problem:** Presence of outliers in the text dataset, particularly languages with low representation, causing sparsity and bias in multilingual text analysis.

  Solution: Removed under-represented languages to enhance uniformity and reduce computational overhead, ensuring better model performance.

- **Problem:** The large scale of the dataset posed memory and computational constraints during preprocessing and training.

  Solution:

  - Processed the dataset in smaller chunks (500 subsets) using memory-efficient methods.
  - Used techniques like HashingVectorizer for feature extraction, reducing the need for storing large vocabularies.
  - Implemented dimensionality reduction methods like the Johnson-Lindenstrauss Lemma (JL Lemma), Structured Low Rank Approximation, and coresets to optimize training time while managing a trade-off between efficiency and accuracy.

- **Problem:** Need to remove noise (e.g., mentions, hashtags, links) and standardize text for model input.

  Solution:

  - Employed regex-based text cleaning to remove unnecessary elements and normalize repeated characters.
  - Used spaCy for lemmatization to reduce word variations and improve consistency.

- **Problem:** The HashingVectorizer produced data with 1,048,576 dimensions, causing computational inefficiencies.

  Solution:

  - Applied Sparse JL matrix for dimensionality reduction to 1,000 dimensions.
  - Used Structured Low Rank Approximation and coresets to further reduce the size and computational cost.

These were the challenges faced along with their solutions .

3. Hypothesis test and their conclusion

The following hypothesis tests were conducted on the datasets :

1. We applied hypothesis testing to test the fact that average polarity of joe biden is greater than the overall average polarity . We got the polarity labels using model training and then testing on the joe_biden and trump tweets .

   H0 = Average polarity of Joe Biden is greater than overall average polarity

H1 = Average polarity of Joe Biden is lesser than overall average polarity

We used the z-test for the same . We take a sample size of 50 tweets from the Jobidden tweets and take all the combined tweets of Joe Biden and Trump as the population . Now , we get the mean of the sample as 0.594 . We get the population mean as 0.573 and population standard deviation as 0.2023 . We use the formulae for z_test to get the value of z_calculated as 0.733 . We are assuming that the level of confidence is 0.05 . This is a one tailed z test ,thus looking it's value in the table we get z_tabular = 1.65 . Thus z_calculated $<$ z_tabular . Thus , we accept the Ho(Null Hypothesis) that the average polarity of Joe Biden is greater than the overall average polarity .

2. We applied hypothesis testing to test the fact that sentiments of Joe Biden is independent (not related) to states . We got the sentiments labels using model training and then testing on the joe_biden and trump tweets .

   Ho = Sentiments of Joe Biden is Independent or not related to states

   H1 = Sentiments of Joe Biden is dependent or related to states

   We used chi-squared testing for independence for testing the null hypothesis . We first get the observed data which is matrix of (3 * 24) and we find the value of chi_square constant which comes out to be 376.55229795512605 . This value is greater than the chi_square critical value which is 62.83 . Thus , we reject the null hypothesis in this case . This implies that sentiments of Joe Biden is dependent or related to states

3. We applied hypothesis testing to test the fact that average polarity among high profile users of Joe Biden is greater than overall average polarity among all high profile users . This is important test as these users influence the most amount of audience due to their high reach .

   H0 = Average polarity among high profile users of Joe Biden is greater than overall average polarity among high profile users.

   Ha = Average polarity among high profile users of Joe Biden is lesser than overall average polarity among high profile users.

   We used Z-test for the same . We took high profile users as the ones with more than 100000 followers . We took size of the sample as 50 and total high profile users as the population .We get the sample mean as 0.59 . The population mean as 0.57 and population standard deviation as 0.202 . We calculated the value of Z constant as 0.73 and considered significance level as 5% . Thus we get z_calculated $<$ z_tabular . Thus we accept the null hypothesis , which implies that the average polarity among high profile users of Joe Biden is greater than overall average polarity among high profile users . This was the result of the hypothesis test .

4. We applied hypothesis testing (F test) to test the whether the variance of the number of likes on twitter posts about Biden , and the variance of the number of likes on twitter posts about Trump are roughly the same or not.

   Ho = Variance of number of Likes in tweets about Biden and tweets about Trump are nearly same .

   H1 = Variance of number of likes in tweets about and tweets about Trump are significantly different.

   We used F test for testing this difference , we took two samples size as 50 from each Trump dataset and Biden dataset.The significance level was set to be 5% , and we found the F statistic value as 65.80. The p-value was found out to be 1.102e-16, which is much lower than the significance , so we reject the null hypothesis , and conclude that the variances are significantly different.

5. We applied hypothesis testing (Z test for two proportions) to Compare the proportion of positive sentiments between Biden and Trump tweets to see if there is a significant difference in the positivity rate.

   Ho = There is no significant difference in the proportion of positive sentiments in Trump and Biden Tweets .

   H1 = There is significant difference in the proportion of positive sentiments in Trump and Biden Tweets .

   We applied Z test for two proportions . We get the calculated z_value as 104.94 , while the z_tabular was 1.96 . Thus abs(z_value) $>$ z_tabular . Thus we reject the null hypothesis implying that there is a significant difference in positive sentiment proportions

4. Compare and discuss the performance of trained models on scaled datasets and without scaled datasets

4.1. Machine Learning Modelling

For sentiment analysis, the dataset consisted of tweets that were preprocessed and converted into numerical features using HashingVectorizer. This method is efficient for handling large-scale datasets as it uses a fixed-size hashing function to map features, avoiding the need to store the entire vocabulary in memory. This approach was particularly useful given the high dimensionality of our data, which included a large number of features and rows.

Once the features were extracted, the dataset was split into training (80%) and testing (20%) subsets to ensure that the model could be evaluated on unseen data . This split provided a balance between having enough data for training while maintaining a significant portion for validation.

4.2. Model Selection and Performance

1. Logistic Regression: Logistic Regression achieved an accuracy of 77-78% on the test set. This algorithm was selected for its strong performance in binary classification tasks like sentiment analysis and its ability to handle high-dimensional data efficiently.

2. Naive Bayes: The Naive Bayes classifier achieved an accuracy of 50%, which is close to random guessing. This performance indicated that the algorithm struggled to capture the complex relationships in the data.

Based on the evaluation results, Logistic Regression was selected as the final model due to its superior accuracy and scalability to handle the large dataset effectively. Link to GitHub

The model metrics showcasing accuracy , confusion matrix and classification report on the test dataset is as follows:

```
ACCURACY:
77.78715269766981

CONFUSSION MATRIX
[[116210  36339]
 [ 31057 119804]]

CLASSIFICATION REPORT
              precision    recall  f1-score   support

           0       0.79      0.76      0.78    152549
           4       0.77      0.79      0.78    150861

    accuracy                           0.78    303410
   macro avg       0.78      0.78      0.78    303410
weighted avg       0.78      0.78      0.78    303410
```

Figure 1. model_metrics

RUNNING TIME OF THE MODEL : HashingVectorizer transforms the input data into a sparse matrix of fixed size . The time complexity of transforming n samples , each containing m words is O(n.m) , where m is the average length of a sample(tweet) . After this , the Logistic Regression's time complexity is O(n.d.k) , where

n : number of training samples

d : number of features (fixed by HashingVectorizer)

k : number of iterations (given as max_iter = 1000)

Overall Time Complexity for Training the Model : O(n.m + n.d.k)

.

The HashingVectorizer runs once , and since d is fixed by the vectorizer , the dominant factor is O(n.d.k) .

Sentiment Analysis Application - After training, the Logistic Regression model was used to predict the sentiments of tweets from both participants in the dataset. The model's predictions were analyzed to identify patterns and insights into the sentiment trends associated with each participant. The results provided valuable information for understanding public sentiment and the effectiveness of communication strategies.

4.3. Randomized Scaling Techniques

The Randomized Scaling Techniques we are using are as follows :

1. **JL Lemma** : The Johnson-Lindenstrauss Lemma is ideal for reducing the dimensionality of high-dimensional data while preserving the pairwise distances between data points . This is useful in NLP tasks where the feature space can be large due to the use of text vectorization methods like HashingVectorizer which we are using in the modelling pipeline .

   The Output size of the training dataset after Hashing Vectorization is as follows :

   The shape of the data after Hashing Vectorization is: (1213636, 1048576) .

   Thus we can observe that the number of dimensions is very large of the output data after the processing .

   We project the d-dimensional data(in this case 1048576) to some lower k-dimensional space using a random matrix while preserving pairwise distance . We are choosing Sparse JL matrix with the dimension reduction to 1000 dimensions . Thus this reduce the time complexity to O(n.p.k) where :

   n : number of training samples

   p : number of reduced features after JL Lemma dimension reduction .

   k : number of iterations (given as max_iter = 1000)

   This fastens up the process of logistic regression on the expense of some accuracy .

   Comparison of Model Performance

Method	Training Time(seconds)	Accuracy(%)	Improvement In Time(%)	Accuracy Loss(%)
Without JL Lemma (Full d)	25.74	77.76	—------	—------
With JL Lemma (Reduced k)	19.58	67.74	~24%	~10%

Figure 2. JL_Lemma_comparision

   Efficiency: JL Lemma effectively reduced the training time by a significant margin.

   Accuracy Trade-off: The reduction in dimensionality led to a noticeable drop in accuracy, highlighting the trade-off between computational efficiency and model performance. Link to GitHub

2. **Structured Low Rank Approximation** -

   1. **Downsampling** - First , the original matrix was downsampled from 1213636 samples to 50,000 samples using K stratified fold in order to handle the RAM constraint.

   2.**Methodology** -

   It begins by calculating the squared l2-norms of all rows and columns in the train matrix which are then normalized to derive probabilities for sampling. Rows and columns with larger norms, which contribute more significantly to the matrix, are sampled with higher probability. Using these probabilities, k rows and k columns are sampled from the training matrix , forming the matrices B (sampled rows) and C (sampled columns). Here k was chosen arbitrarily to be 500.

   Next, it computes B.BT the row-row similarity matrix, and adds a small regularization term epsilon to ensure numerical stability. The inverse of this regularized matrix is used to construct a projection matrix P, which represents the contribution of the sampled rows. Finally, W a reduced representation of the original matrix, is formed by selecting k rows of P corresponding to the sampled columns. The output matrices C and W together approximate the original matrix A, enabling efficient computations in a lower-dimensional space. This method is particularly suited for large, sparse matrices where direct computation would be computationally expensive. We used the matrix C for reducing the dimension of training matrix.

   The final reduced matrix has dimensions (50000, 500) which was used for training.

   The time complexity of the sampling-based low-rank approximation is primarily influenced by the sparsity of the matrix A and the rank k. Computing row and column norms takes $O(\text{nnz}(A))$, where $\text{nnz}(A)$ is the number of non-zero elements in A. Sampling rows and columns requires $O(n + d)$, and extracting them from A is $O(k \cdot (n + d))$. Matrix multiplications involving B and its transpose contribute $O(k^2 \cdot d)$, while the inversion of $B \cdot B^T$ is $O(k^3)$. The dominant terms are

$O(\text{nnz}(A) + k \cdot d^2)$, making the method efficient for large sparse matrices when $k < \min(n, d)$.

Comparison of Model Performance

Method	Training Time(seconds)	Accuracy(%)	Improvement In Time(%)	Accuracy Loss(%)
Without Low Rank Approx (Full d)	0.4512s	74.42	———	———
With Low Rank Approx	0.06s	46.68	~86%	~30%

Figure 3. Low_rank_approximation_comparision

Efficiency: Low_rank_approximation effectively reduced the training time , it was more effective in time reduction than JL lemma .
Accuracy Trade-off: The reduction in dimensionality led to a noticeable drop in accuracy, highlighting the trade-off between computational efficiency and model performance.(more than JL lemma) . Link to GitHub

3. **Coresets** - A coreset is a small, representative subset of a large dataset that preserves the important properties of the original dataset, allowing algorithms to run faster and with reduced computational requirements. The key idea is to sample the data in a way that ensures the reduced dataset (coreset) reflects the statistical or geometric structure of the original dataset.

   Downsampling: The large training dataset (df_train) is initially downsampled to 100,000 samples using stratified sampling to ensure class balance. This reduces the computational load so that the training process can fit in RAM in the next steps.

   Training using coresets : The logistic regression model is trained on the coreset, which has only 866 data points. Training on the coreset is extremely fast (0.023 seconds) compared to the original downsampled data dataset (1.27 seconds). The significant reduction in training time although costed the accuracy to decrease from 69% to about 64%. (In both cases for fair comparison we have created 210 features using hashing vectorizer.)

   Advantages: Significant reduction in training time and computational resource usage and effective when computational efficiency is prioritized over slight accuracy loss.

   Disadvantages: Loss of some predictive accuracy compared to models trained on larger datasets Let there be n samples in training and m in testing set .

   Time Complexity - Resampling : O(n) ; Vectorization: O(n+m) ; Clustering: O(n) (worst case) ; Training and Evaluation: O(m) ; Total: O(n+m) .

   Comparison of Model Performance

Method	Training Time(seconds)	Accuracy(%)	Improvement In Time(%)	Accuracy Loss(%)
Without coresets	1.272 s	69.97	———	———
With coresets	0.023 s	64.59	~98%	~8%

Figure 4. Coresets_comparision

Efficiency: Low rank approximation effectively reduced the training time , it was the least among the three techniques used .
Accuracy Trade-off: The reduction in dimensionality led to a noticeable drop in accuracy, highlighting the trade-off between computational efficiency and model performance. There was still not very much drop in accuracy . Link to GitHub

5. Conclusion & future of the project

5.1. Conclusion

The project focused on preprocessing and analyzing the social media for sentiment analysis and understanding public opinion . The tasks included data cleaning and preprocessing , data transformation , model development and training/evaluation and visualizing the dataset and getting insights from it. A robust pipeline was developed to clean tweets by removing noise such as mentions , hashtags , links etc. Language was also detected to ensure only english tweets are analysed improving the relevance of data. The missing values were replaced and removed to ensure data consistency. A logistic regression model was trained using a pipeline incorporating feature extraction (HashingVectorizer) and classification. The model achieved reasonable accuracy of 77.87% indicating its capability to classify sentiments effectively. Metrics such as accuracy , confusion matrix , and classification reports were analyzed to evaluate the performance.

5.2. Future Work

In the future , we will be incorporating advanced NLP techniques such as named entity recognition (NER) or dependency parsing to capture more features from the text. Also , more experiments can be done using regex to give better representation. We will also be exploring deep learning models for improved sentiment classification. We will be performing hyperparameter tuning for the logistic regression model to optimize performance further. Also, we can expand the dataset to build more diverse and balanced training set ,potentially leading to better generalization. We can also extend the model to classify tweets into fine grained sentiment categories like very positive , neutral , very negative instead of binary classification. Also , at the end with these features incorporated , we will be deploying a web based dashboard or API to provide real time sentiment analysis and visualization for incoming tweets.

6. References

- Twitter Sentiment Analysis
- Twitter Sentiment Analysis using Python - GeeksforGeeks
- Hypothesis Testing of Tweet Text Using NLP — SpringerLink
- JL Lemma for Logistic Regression Reference
- Low-rank approximation reference
- Coresets and coding application reference
- Github Link