Data Mining and Text Mining Project - CS 583
Sentiment Analysis of Presidential Debate Tweets

Davide Ettori, Angelo Zangari
Instructor: Professor Liu Bing

December 2024

Abstract

Sentiment analysis of social media content presents complex challenges in natural language processing. This research project focuses on developing a robust sentiment classification model for tweets related to presidential candidates Obama and Romney, aiming to accurately categorize them into positive, negative, and neutral sentiments. Our comprehensive approach involves text preprocessing, diverse vector embedding strategies, and a comparative analysis of both traditional and modern machine learning models, aiming to reach the best possible performance in terms of Accuracy, Precision, Recall and F1-Score. Good performance were obtained with a Feed Forward Neural Network, but in the end the best model is a SVM with RBF kernel, trained on embeddings generated by the pre-trained SentenceTransformer model: accuracy 68%, F1 on class -1 and +1 of 75% and 0.67%. LSTM networks had worse performance probably because the tweets are too short to fully exploit that architecture.

1 Introduction

Sentiment analysis plays a critical role in understanding public opinion and has become a fundamental task in data mining and natural language processing. This project focuses on building a classifier to determine the sentiment expressed in tweets related to the 2012 U.S. presidential election, specifically regarding the two candidates: Barack Obama and Mitt Romney. The primary goal of this project is to develop an effective pipeline, from data cleaning to model inference, capable of categorizing tweets into the defined classes. In our specific case, the task involves separating each tweet into one of three classes: positive (1), negative (-1), or neutral (0), where the neutral class indicates a lack of expressed opinion.

1.1 Data Overview

The training data consists of tweets collected during a presidential debate between Obama and Romney. These tweets are stored in an Excel file, with separate sheets for each candidate. Each tweet in the training data is labeled with one of four possible classes: Positive (1), Negative (-1), Neutral (0) and Mixed (2).

Mixed signifies the presence of both positive and negative sentiments within a single tweet. However, for the purposes of this project, the mixed class (2) is excluded from both training and testing. The test data, provided without class labels, follows a similar structure to the training data, with separate sheets for tweets about Obama and Romney. During the testing phase, the classifier will evaluate the sentiment of these tweets, restricted to the three target classes: positive, negative, and neutral.

1.2 Embeddings Analysis

In this project, various embedding techniques are used to transform the textual data into numerical representations suitable for machine learning and deep learning models:

- **Label Encoding**: Sentiment labels are transformed to 1-hot vectors for compatibility with cross-entropy loss functions used in neural networks. During prediction, the labels are reverted by subtracting 1 to restore the original sentiment classes.
- **Word2Vec**: This method embeds individual words into fixed-length vectors based on their co-occurrence in a large corpus. While it lacks contextual understanding of entire sentences, Word2Vec is computationally efficient.
- **Sentence Transformers**: These models encode entire sentences into dense vectors, capturing the semantic meaning of the text as a whole. Although more computationally intensive than Word2Vec, sentence transformers provide richer contextual embeddings, which can improve classification performance.
- **TensorFlow Embedding Layer**: For deep learning approaches, an embedding layer is integrated into the neural network. This layer learns task-specific embeddings during model training, allowing for end-to-end optimization.

For traditional machine learning methods such as Naïve Bayes and SVM, the same label encoding is adopted by convention, ensuring consistency across approaches.

1.3 Evaluation Metrics

Evaluation of the sentiment classifier's performance will be based on its ability to accurately assign sentiment labels to the test data. Metrics such as precision, recall, F1-score, and overall accuracy will be employed to assess the model's effectiveness. These metrics will ensure a comprehensive understanding of the classifier's strengths and limitations across different sentiment categories. In particular we will focus those metrics on the classes -1 and +1, without considering too much the class 0 (neutral sentiment).

2 Techniques

2.1 Data Preprocessing

In this section we'll delve deeper into the data processing and cleaning pipeline that was necessary to perform in order to obtain clean and well formatted data. The initial dataset was available in the form of two excel sheets, with one page being dedicated to Obama's tweets and the other one to Romney's.

The work needed to perform data cleaning was considerable, given the fact that the initial data was very dirty; in particular specific aspects that had to be fixed involved the following issues:

- Rows with tweets but missing the labels.
- Rows with wrong labels, for instance strings instead of the desired numbers $\{0, \pm 1, 2\}$.
- Rows with empty values for the date and time fields.

After cleaning the dataset, to perform data preprocessing we integrated all the data in a *pandas dataframe*; this allowed us to seamlessly remove useless punctuations, links and hashtags, exploiting specific Regex expressions.

To perform model training we split the dataset in the following way: 70% of the data was training, 10% was used for validation and the remaining 20% was kept for testing. Regarding the model itself instead of training two different models, one for each political candidate, we trained a single one on all of the data; this choice was done because after experimenting with two separate models, no clear difference was found and we preferred one single model trained on more data, to have a lower variance on unseen test data.

2.1.1 Feedforward Neural Network with Word2Vec

Word2Vec is an embedding technique trained in an unsupervised manner. In our approach, we used a pre-trained Word2Vec model that was trained on a Twitter dataset, leveraging transfer learning. Each word in the tweets is encoded using Word2Vec embeddings. To process the entire tweet, we compute the mean of the word embeddings across all dimensions, creating a single vector representation for the tweet.

This average embedding is then passed as input to a neural network, which predicts the sentiment of the tweet by optimizing the categorical cross-entropy loss. While this approach provides a compact representation of each tweet, it has significant limitations. Word2Vec encodes words independently of their context, which makes the embeddings less expressive, especially for tasks where the sentiment depends on the nuanced context within a sentence.

2.1.2 Feedforward Neural Network with SentenceTransformer

To overcome the context insensitivity of Word2Vec, we used SentenceTransformer, a library that leverages a pre-trained transformer model to encode entire tweets into dense vectors (approximately 700 dimensions). Unlike Word2Vec, SentenceTransformer captures the context of each word using attention mechanisms, producing a more expressive and context-aware embedding for each tweet.

These embeddings are then fed into a feedforward neural network. The network consists of multiple layers with ReLU activations, and the final layer applies a softmax activation for sentiment classification. This approach reduces variance and aids in convergence, especially in the early training stages, because of the richer representation of the input.

2.1.3 LSTM Neural Networks

Long Short-Term Memory (LSTM) networks are well suited for natural language processing tasks because they can selectively retain or forget information over sequences, handling long-term dependencies in text (even if the tweets are often times quite short).

In our implementation, embeddings are learned directly within the Neural Network using an embedding layer at the input. This is followed by two LSTM layers that process the sequence of words in each tweet. The output of the LSTM layers is passed through fully connected layers to make the final sentiment prediction. Unlike previous approaches that used a single vector to represent the tweet, the LSTM processes the entire sequence of words, allowing it to understand the nuances of the text.

2.2 Traditional Machine Learning Models

In addition to neural networks, we experimented with several classical machine learning techniques, which are all trained on the embeddings from Sentence Transformers. The models include:

- **Naive Bayes:** A probabilistic model based on Bayes' theorem, assuming independence between features. It is simple and fast but struggles with feature interactions.
- **Logistic Regression:** A linear model for binary or multi-class classification, optimized using cross-entropy loss, aiming at maximizing the likelihood of the data.
- **K-Nearest Neighbors (KNN):** A non-parametric method that classifies samples based on the majority class of their nearest neighbors.
- **Decision Trees:** A tree-based model that splits the data recursively based on feature values. They capture non-linear relationships but are prone to overfitting.
- **Support Vector Machine (SVM):** SVM with a Radial Basis Function kernel, which can separate non-linear data by mapping it to a higher-dimensional space.

For these models, we also experimented with class weighting to address class imbalance in the dataset. However, this adjustment did not improve performance, so we reverted to uniform class weights.

After that we also tried Ensemble Methods, like Bagging and Boosting, especially with Decision Tree (Random Forest) but again they don't show any notable improvement on the results.

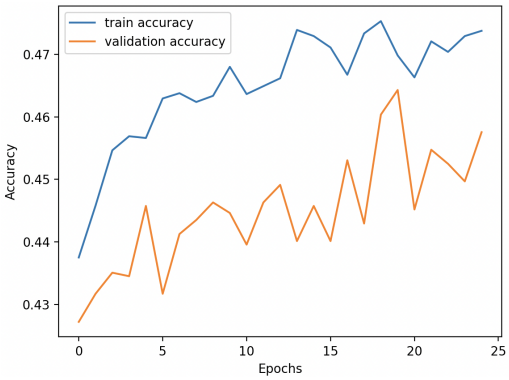
Finally, we also experimented Dimensionality Reduction using the PCA method, hoping to capture a better representation of the data. In this case we observe that the performance of the various models reach the maximum on a dimensionality of 250, with a notable reduction from around 750 dimensions of the standard embeddings of SentenceTransformer. Anyway, since our objective was to have better performance, not the same performance but with lower dimensions, we discarded this possibility even if we recognize that it could be an useful approach for analyzing the efficiency.

In the end, our final best model is the SVM with RBF kernel and SentenceTransformer embeddings at full dimensionality. This also indicate some kind of Gaussian distribution inherently present in the dataset.

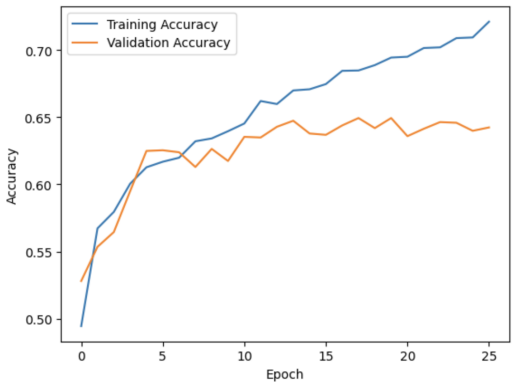
3 Results

Model	Accuracy	Precision	Recall	F1 Score
Naive Bayes	0.51	0.64 / 0.54	0.62 / 0.65	0.63 / 0.59
Decision Tree	0.61	0.60 / 0.69	0.85 / 0.48	0.70 / 0.56
Logistic Regression	0.66	0.67 / 0.68	0.79 / 0.60	0.72 / 0.63
KNN	0.60	0.62 / 0.59	0.77 / 0.60	0.69 / 0.60
SVM	**0.68**	**0.68 / 0.72**	**0.83 / 0.62**	**0.75 / 0.67**

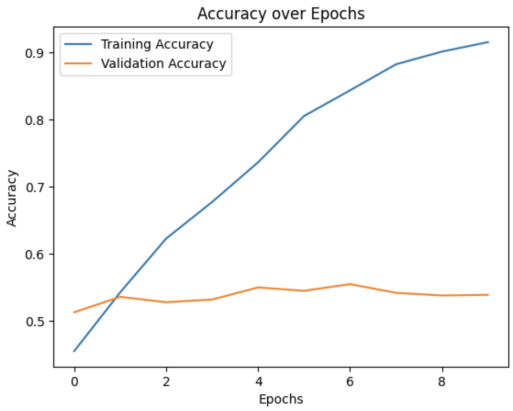
Table 1: Performance Metrics of Machine Learning models (Class -1 and +1)



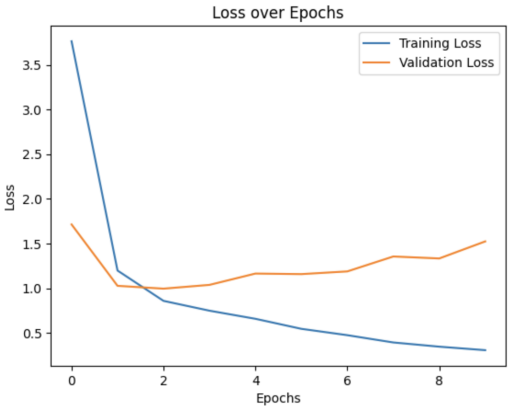
(a) FFNN with W2V: The model shows evident underfitting.



(b) FFNN with Sentence Transformers: The model shows slight overfitting.



(c) LSTM: The model shows clear overfitting, immediatly.



(d) LSTM: The crossentropy loss only decreases on training data.

Figure 1: Visualization of different Neural Networks and their behaviors.

4 Conclusion

In conclusion, we can say that the best method for embedding tweets is Sentence Transformers, given their ability to capture context and produce more meaningful vectors. Furthermore, the best model for recognizing sentiment in these tweets is the SVM with an RBF kernel. Neural networks are not bad, but they fail to generalize as well, probably due to the lack of sufficient training data, leading to high variance that negatively impacts their performance on test data.

This project allowed us to tackle several new challenges, which allowed us to learn many different aspects. For instance, this was the first time that we had to deal with extensive data preprocessing, since we were starting from a dataset of raw strings, rather than simple and cleaned numerical values. Another great learning point was seeing how an older and simpler model like SVM could outperform complex neural network architectures. This reminded us of the importance of understanding the underlying architecture of the models we are using, as not necessarily new and fancy stuff is all-around better than classical and well established methods.