Statistical NLP HW-1

Abstract

This document presents my solution and critical analysis of the Presidential Speech Classification task, which was assigned as part of the CS 272: Statistical NLP course at UCI. Throughout the competition on Kaggle, I used my username "Adithya" to submit all my solutions.

1 Introduction and Problem Statement

For this assignment, we were given a dataset comprising speech fragments from 19 US presidential candidates. Our objective is to train both a supervised and a self-supervised model that can accurately predict which candidate spoke a given speech fragment.

1.1 Dataset Analysis

The dataset consists of 4370 labelled samples in the training data, along with an additional 414 labelled samples that were used for development purposes. Furthermore, there are 43342 unlabelled samples that are intended for self-supervised training.

A closer examination of the dataset revealed that it is highly imbalanced, with the majority class "ClintonPrimary-2008" accounting for almost 30.08% of the total data, while the minority class "CainPrimary-2012" represents only 0.3% of the training data. This significant class imbalance can potentially cause the classifier to exhibit bias towards the majority class, which is an issue that needs to be addressed.

To gain insight into the high-dimensional structure of the training data, a t-SNE plot was generated using the count vectorizer representation. The resulting plot showed that the samples were tightly clustered together, with significant overlap between classes. To aid visualization, principal component analysis (PCA) was applied to reduce the dimensionality of the data. However, this did not have a significant impact on the overall structure of the t-SNE plot. The central cluster of samples remained prominent, but there was a notable increase in the number of outliers, indicating that some samples were more distinct from the rest of the data than previously thought.

1.2 Supervised Classifier

The starter code provided a standard count vectorizer to convert the text into count vectors, which were then fed into a linear logistic regression model for classification. However, to improve the performance of the model, some preprocessing steps were applied to the text data before training. This processed text were then used to generate the count vectors, which were fed into the logistic regression model for classification.

Some techniques attempted during the experiment to improve the validation accuracy include:

1. Text Preprocessing and filtering (lemmatization,stop word removal,etc.)
2. Increasing the number of N-grams from (1,1) to (1,2) to explore if the additional features could enhance performance.
3. Using PCA for dimensionality reduction on the larger (1,2) N-gram dataset.
4. Employing K-means clustering to classify all the speech samples into 19 distinct clusters.
5. Creating 19 reference documents by merging all speech fragments for each president to calculate a similarity measure later.

1.3 Text Processing

During the text processing phase, several techniques were attempted, including combinations of Porter stemmer, Snowball stemmer, and lemmatizer. Various maxDF and minDF values were also tested to remove custom stopwords from the dataset, but this did not yield any improvements. However, using the inbuilt stopword list for the

"English" language in NLTK led to a performance boost of 1%. Therefore, the proposed solution incorporated NLTK's stopword list.

This "cleaned" text was used as input for three different vectorization methods: count vectorizer, hashing vectorizer, and tf-idf vectorizer. Various parameters were manually adjusted and the performance of each vectorizer was compared. It was found that the count vectorizer produced the best results, making it the suitable choice for the proposed solution.

As a final step, the number of features was increased by modifying the N-gram range from (1,1) to (1,2). This adjustment resulted in an increase in the number of features in the count vector from 7778 to 46976. As a result, the validation accuracy improved from 41.54% to 43.47%.

1.4 Dimensionality Reduction

Increasing the Ngram range can potentially introduce redundant features. For example, the unigrams "United" and "States" are highly correlated with the bigram "United States." To optimize the feature vector, we need to remove this redundancy. Additionally, increasing the number of features can lead to longer execution times. To address this, a a standard dimensionality reduction technique (PCA) was used to reduce the number of features while retaining or improving the validation accuracy.

The relationship between the number of PCA components used and validation accuracy was compared, as shown in Figure 2. As the number of components increased, the validation accuracy improved. Although PCA achieved a reduction of almost 11 times in the number of parameters while still coming close to the proposed approach (as seen in Table 1.7), it was not incorporated into the final approach because the objective is to maximize validation accuracy.

1.5 Similarity

When a query is submitted to a search engine, the engine computes the similarity between the query vector and document vectors to retrieve the most relevant documents.

Similarly, we can use a feature extraction technique for our current problem by concatenating speech samples from each speaker to create a speech document. These 19 reference documents can be used as a basis for testing and self-supervised learning.

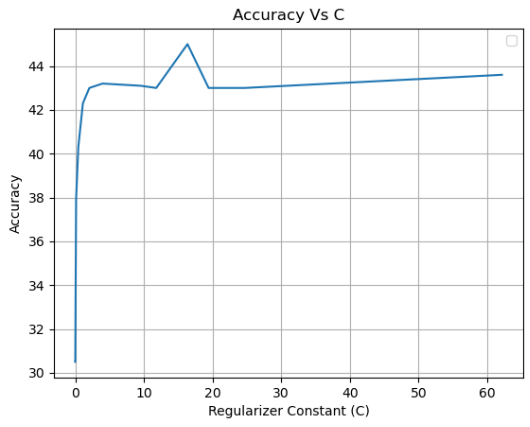

Figure 1: Effect of Regularization Constant "C" on Validation accuracy.

During testing, when a speech sample is encountered, we compute its cosine similarity with all 19 reference documents. This 19-dimensional vector can then be used as features for the logistic regression model. We concatenate this cosine similarity vector to the existing training data before passing it to the logistic regression classifier.

If we use only cosine distances to classify the data, the resulting validation accuracy is 41%. This approach involves computing the most relevant document out of the 19 reference documents for each speech fragment and assigning the same label as the most relevant reference document to the fragment. While this method is not as effective on its own, it can still serve as a useful baseline for comparison with other classification approaches.

1.6 Hyperparameter Tuning

For the logistic regression classifier, a Randomized search across different values for the solver, the regularization constant and the penalty was conducted to obtain the best classifier performance. Figure 1 depicts the variation of the validation accuracy as the value of the regularization constant "C" varies (fixed penalty and solver). Using the optimal values found by the randomized search the validation accuracy of the model rose from the base of 42 % to 42.5 % .

1.7 Results

Table 1.7 denotes the validation accuracies that were obtained using different feature extractions and algorithms. The proposed solution using cosine similarity outperforms all other approaches including the baseline.

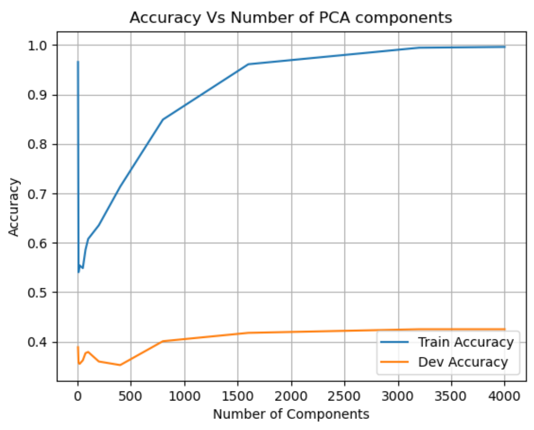

Figure 2: Effect of Number of Components in PCA on the Validation accuracy.

To group similar samples and aid in classification, K-means clustering was applied to the entire training data, resulting in 19 clusters. Each cluster was assigned a label based on the majority class within that cluster. This clustering approach was then used to predict the class of speech fragments. However, despite extensive hyperparameter tuning, the resulting validation accuracy remained low, peaking at around 18%. This suggests that the K-means clustering approach may not be well-suited for the task

Idx	Feature	Acc$_{dev}$ (%)
1	**Self Supervised**	**44.44**
2	**Proposed Approach**	**43.47**
3	Proposed + PCA	43.23
4	Default (Baseline)	41.30
5	Pure Cosine Similarity	41.01
6	Lemmatization	40.33
7	Porter stemming	37.92

2 Self Supervised

In this approach, the training dataset is augmented with samples from the unlabelled set and the labels are the predictions that the model makes for that corresponding sample. For this part the previous supervised classifier is used as a base upon which the unsupervised training is added. Some modifications that were tested for improving performance are-

1. Ensemble of classifiers for deciding whether a sample is added to the training dataset.

2. Effect of number and type of data samples that are presented to the model.

2.1 Algorithm

The way the self supervised algorithm works is as follows -

Algorithm 1 Self Supervised Learning Algorithm

```
$lst \leftarrow [train, val, unlabelled]$
$X_i \leftarrow Concat(X_i, CosSim(X_i)) \forall i \in lst$
$i \leftarrow 0$
$isFinished \leftarrow False$
$model \leftarrow LogisticRegression()$
while $i \leq MaxEpochs$ and NOT $isFinished$ do
    $model.fit(X_{train}$,y)
    $newSamples = ensemblePredict(X_{unlabelled})$
    $X_{new} \leftarrow GetNewSamples(X_{unlabelled})$
    if $newSamples$ != NULL then
        $X_{train} \leftarrow Concat(X_{train}, newSamples)$
    else
        $isFinished \leftarrow True$
    end if
end while
```

The algorithm begins by processing all unlabelled data and generating a cosine similarity matrix for it using the previously computed reference documents. This matrix has the shape of (number of samples, 19). Next, the unlabelled speech text data is transformed into a vector representation using the count vectorizer, which generates a bag of words representation of the text. This vector representation is then combined with the cosine similarity matrix to create the final feature vector, which is passed to the logistic regression model.

Once the model is trained and evaluated on the labelled dataset, it is used to make predictions on the unlabelled dataset. The predictions are then used to determine whether a sample should be added to the training dataset. This is accomplished by examining the confidence of the model's prediction. An ensemble of classifiers, including K-Nearest Neighbours, Random Forest Classifier, Multinomial Naive Bayes, and Logistic Regression, are all simultaneously trained on the training data. If at least three out of the four classifiers predict the same label, and the predicted probability exceeds a predefined threshold of 95%, then the sample is considered valid and will be added to the training data. The value of 95% for the predefined threshold was obtained by manually changing the values and evaluating their impact on the algorithm's performance.

2.2 Results

From figure 3 we see that the validation accuracy (orange line) is decreasing while the training accuracy increases (blue line). This is indicative of

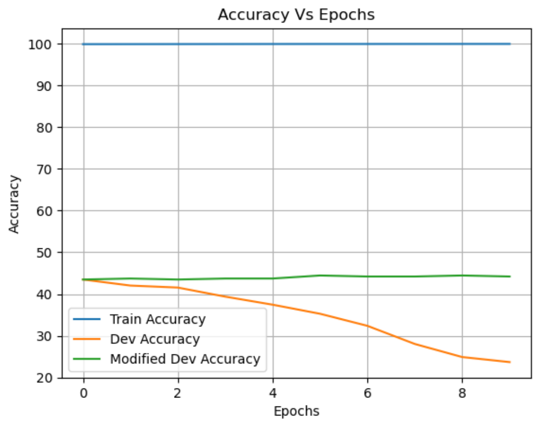

Figure 3: Change in Validation Accuracy over epochs

overfitting. To alleviate the issue, the value of "C" regularization parameter for the logistic regression model was varied but this did not seem to improve the model's performance as the accuracy continued to decrease as the value of the regularizer was increased from 0 to 100 as evident from figure 1

2.3 Critical Analysis

It was observed that, in every epoch, the number of samples transferred to the training data for a fixed threshold increased, despite a drop in validation accuracy. This observation suggests that the model's confidence in its predictions increased over time. This pattern was consistently demonstrated across all classifiers in the ensemble.

A more in depth examination of the results uncovered that most of the samples ($\approx 96\%$ of the unlabelled samples that are added to the training data belong to one of the two dominant classes) are being assigned the labels "CLINTON-PRIMARY2008" or "OBAMA-PRIMARY2008". Both these classes are the 2 majority classes that together constitute 48% of the overall training data.

This observation sheds light on the cause of the drop in validation accuracy. By predicting only the two dominant classes, the model adds more samples from these classes to the training data, further skewing the distribution and exacerbating the imbalance. Consequently, the model improves in identifying the majority classes, but struggles with the remaining ones during training. This results in lower validation accuracy, highlighting the importance of addressing class imbalance in the training data.

To address this issue, a possible solution is to limit the number of samples added to the training data per epoch, while also balancing the distribution of classes. Specifically, we can exclude the two dominant classes and only allow samples from the remaining classes to be included in the training dataset, thereby providing the model with more opportunities to train on minority classes. Furthermore, by restricting the selection to the top 1% of samples, we can prevent large changes to the underlying distribution and potentially improve model performance.

The same self-supervised learning algorithm describer earlier with an additional restriction on the classes added and the number of samples added gives us an improved validation accuracy of 44.44% as can be seen from table 1.7. From figure 3, we see that the accuracy for the updated approach (Green Line) remains around 43% across all epochs which is an improvement compared to earlier.

Finally, the percentage of samples allowed per epoch was increased from 1% to 5%. However, this change did not result in an improvement in accuracy as initially expected. Instead, the accuracy began to decline like earlier, potentially due to the sudden increase in the number of samples being added to the training set at once.

3 Conclusion and Future Work

In conclusion, incorporating the cosine similarity matrix as an additional feature during training led to improved performance compared to the baseline score. However, when the same approach was applied to the self-supervised setup, the model's performance did not improve, and the development accuracy consistently decreased after every epoch. This could be due to the severe class imbalance in the training dataset, which caused the model to be biased towards predicting the majority class. Additionally, the sudden increase in the number of samples during training may have disrupted the underlying data distribution and caused the performance to deteriorate. By controlling the addition of samples to the dataset and ensuring that the underlying distribution remains intact, we were able to improve the validation accuracy of the model. In future work, one possible direction is to explore dynamic adjustments to the training parameters, rather than using static values, to enable the model to adapt to the changing data distribution and achieve better performance.