# NLP Assignment 1

Tanishq Sharma

March 18, 2025

## 1 Code Mixed Text Classification

We have been given three datasets namely Hate, Humor and Sarcasm Text. We need to train the data on the training set and test it on the validation set. The evaluation would be on the F1-score

### 1.1 Data Preprocessing

The various methods I tried on every dataset were:

1. Removal of URLs
2. Removal of usernames (starting with @)
3. Converting to lowercase
4. Removal of words starting with numbers
5. Replacing dots with spaces
6. Removal of punctuations and special characters
7. Removal of extra spaces
8. Removal of stopwords
9. Lemmatization (for english words)

### 1.2 Hate Classification

The classification is done on the five models given in the table below. I used CountVectorizer from scikit-learn to extract the ngram features. By tuning, I got to know the best results came from **ngrams=(1, 3)** which means a combination of unigrams, bigrams and trigrams.

| Model | Accuracy | F1 Score |
|---|---|---|
| Logistic Regression | 0.68 | 0.78 |
| SVM (Linear) | 0.70 | 0.79 |
| SVM (RBF) | 0.70 | 0.82 |
| Random Forest | 0.70 | 0.82 |
| **Naïve Bayes** | **0.71** | **0.82** |

Table 1: Performances of Different Models on Raw Hate Text Data

We take the best model amongst from the above table which is **Naive Bayes**.

### 1.2.1 Data Preprocessing

Amongst all the above data preprocessing techniques, these techniques gave the best results:

1. Removal of URLs,
2. Removal of non-alphabetical strings (Its is not recommended but here increases the accuracy a bit).

**The final performance of Linear SVM become Accuracy: 0.7155 and F1 Score: 0.82**

## 1.3 Humor Classification

The classification is done on the five models given in the table below. I used CountVectorizer from scikit-learn to extract the ngram features. By tuning, I got to know the best results came from **ngrams=(1, 3)** which means a combination of unigrams, bigrams and trigrams.

| Model | Accuracy | F1 Score |
|---|---|---|
| Logistic Regression | 0.68 | 0.62 |
| Random Forest | 0.68 | 0.62 |
| SVM (RBF) | 0.49 | 0.60 |
| **SVM (Linear)** | **0.70** | **0.65** |
| Naive Bayes | 0.65 | 0.36 |

Table 2: Performances of Different Models on Raw Humor Text Data

We take the best model amongst from the above table which is **Linear SVM**.

### 1.3.1 Data Preprocessing

Amongst all the above data preprocessing techniques, these techniques gave the best results:

1. Removal of usernames,
2. Replacement of dot(.) with space.
3. Removal of punctuations and special characters

**The final performance of Linear SVM become Accuracy: 0.7186 and F1 Score: 0.67**

## 1.4 Sarcasm Classification

The classification is done on the five models given in the table below. I used CountVectorizer from scikit-learn to extract the ngram features. By tuning, I got to know the best results came from **ngrams=(1, 3)** which means a combination of unigrams, bigrams and trigrams.

| Model | Accuracy | F1 Score |
|---|---|---|
| Random Forest | 0.90 | 0.95 |
| Naive Bayes | 0.90 | 0.95 |
| SVM (RBF) | 0.95 | 0.98 |
| Logistic Regression | 0.96 | 0.98 |
| **SVM (Linear)** | **0.96** | **0.98** |

Table 3: Performances of Different Models on Raw Sarcasm Text Data

We take the best model amongst from the above table which is **Linear SVM**.

#### 1.4.1 Data Preprocessing

Amongst all the above data preprocessing techniques, these techniques gave the best results:

1. Removal of URLs,
2. Removal of usernames,
3. Removal of strings starting with numbers,
4. Replacement of dot(.) with space.
5. Removal of punctuations and special characters

**The final performance of Linear SVM become Accuracy: 0.96 and F1 Score: 0.98**

### 1.5 MAJOR CHALLENGES

**The main issue with all the dataset was the major class imbalance. We can never get good results unless we do some kind of data augmentation and make samples of both classes roughly equal in number. Also, As it is hinglish, there are not many reliable stopwords for hinglish present on the Internet. Converting all of them to english or hindi might help better. Preprocessing decreases word count but the data is already very small that is why it is an issue. Increasing the n in ngrams extraction, the recall of one class is low, imbalance increases. Using custom hinglish stopwords reduces performance. Removing usernames from text would lead to loss of info as hate is directed after**

## 2 Sequence Labeling Task

Part-of-Speech (POS) tagging is a fundamental task in Natural Language Processing (NLP). This report evaluates the performance of the Viterbi algorithm for POS tagging using a Hidden Markov Model (HMM). We compare a baseline implementation against a noise-handling version, highlighting improvements in accuracy and robustness.

### 2.1 Methodology

We train an HMM-based POS tagger on a labeled dataset and evaluate it on both clean and noisy test data. The transition and emission probabilities are estimated using add-one (Laplace) smoothing. Unknown words are handled using a backoff strategy that assigns low probabilities based on word suffixes.

The evaluation is conducted using:

- A baseline Viterbi decoder.
- An improved version with noise-handling techniques.

Accuracy is computed as:

$$\text{Accuracy} = \frac{\text{Correctly predicted tags}}{\text{Total tags in test set}} \tag{1}$$

### 2.2 Results and Analysis

| **Model** | **Test Accuracy** | **Noisy Test Accuracy** |
|---|---|---|
| Baseline Viterbi | 83.4% | 75.74% |
| Noise-handling Viterbi | 75.8% | 79.6% |

Table 4: Comparison of baseline and noise-handling Viterbi implementations.

The results indicate that incorporating noise-handling strategies improves accuracy on noisy data by approximately 7.3%, while maintaining a slight edge on clean data. Error analysis revealed that the baseline struggled with unseen words, whereas the improved version performed better due to backoff techniques.

### 2.3 Conclusion

Our evaluation shows that a noise-aware Viterbi algorithm significantly enhances POS tagging performance in real-world scenarios. Future work could explore deep learning approaches or hybrid models for further improvements.

## 3 Deliverable URLs

1. Code-Mixed Text Classification : Link
2. Sequence Labelling Task : Link