Naïve Bayes for Text Classification

Roger Bukuru (BKRROG001)

University of Cape Town
Natural Language Processing (CSC5035Z)
Assignment Report

1 Introduction

In this assignment, we explored sentiment analysis using the Naïve Bayes classifier. We conducted this experiment to test the performance of this classifier under two different tokenization methods, namely Word-Based tokenization and Byte-Pair Encoding (BPE) on an African language dataset. We worked with the AfriSenti dataset, which provides sentiment-labelled text in 14 African languages composed from tweets (messages on X previously Twitter). For our task, we opted to use Kinyarwanda as our language of choice. We proceeded by pre-processing the data, implementing the classifier on both tokenization methods, and finally evaluating its performance.

2 Data Processing

Given our main goal is to implement a binary classifier, our first pre-processing step was to remove all tweets labelled as *neutral*, therefore only keeping tweets labelled as either *positive* or *negative* in tone. Furthermore, we standardized our tweet corpus by implementing the following straightforward techniques:

- Replace URLs with a [URL] token
- Replace numbers with a [NUM] token
- Replace *@user* words with a [USR] token and finally remove all trailing spaces.

One of the simplifying assumptions of our classifier is if we encounter unknown words that we simply ignore them. However, we opted to handle unknown words with the ⟨**UNK** ⟩ token for benefits such as retaining information and being able to calculate the likelihood estimates of unknown words. Finally, to make all sentences (tweets) in the training batch the same length, we added padding by adding a ⟨**PAD** ⟩ token.

3 Tokenization

We implemented two types of tokenization methods, the first being the standard word-based tokenizer and the second being the Byte-Pair Encoding (BPE) tokenizer. For the word-based tokenizer, we obtained a vocabulary size of 13674 with 30894 tokens, and for the BPE tokenizer, we obtained a vocabulary size of 1171 with 80362 tokens.

4 Model Training and Prediction

Training our classifier first involved performing text vectorization. We opted to use the simple binary text vectorization method. This was implemented after each respective tokenization method and then fed to our classifier, moreover we applied Laplace smoothing to handle unseen words. In table 1 we show the results of our classifier when used on both tokenization methods.

Tokenizer	Accuracy
Word-Based	80.0948
BPE	80.0948

Tab. 1: Naïve-Bayes Classifier Prediction Results

We note that we achieved the exact same prediction accuracy which potentially indicates that for this corpus either tokenization method would suffice, we interrogate this stance by evaluating the classifier on the development set.

5 Model Evaluation

We performed model evaluation by applying our classifier to a development set. In tables 2 and 3 we show the results.

	precision	**recall**	**f1-score**	**support**
0	0.78	0.81	0.80	287
1	0.75	0.72	0.73	225
accuracy			0.77	512
macro avg	0.76	0.76	0.76	512
weighted avg	0.77	0.77	0.77	512

Tab. 2: Word-Based Tokenizer Naïve Bayes Evaluation

	precision	**recall**	**f1-score**	**support**
0	0.55	0.73	0.63	287
1	0.41	0.24	0.30	225
accuracy			0.52	512
macro avg	0.48	0.49	0.47	512
weighted avg	0.49	0.52	0.49	512

Tab. 3: BPE Tokenizer Naïve Bayes Evaluation

We observe that despite similar performances with the test set, upon evaluation the BPE tokenizer performs worse then the word-based tokenizer across all metrics, which more or less maintained its test set performance. This shows that for the task on hand using the word-based tokenizer would be more beneficial. One reason could be that the granularity of tokenization offered by the BPE is lacking given the morphological nature of the language in question i.e the fragmented tokens might not provide as coherent and informative features as whole words.