Text Classification Report

Syiti Liviani M. (1301174515) – IF-41-INT

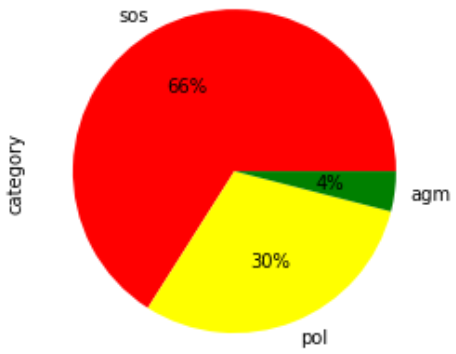
The task is to do Text Classification of a collection of Bahasa Indonesia document into several classes. In this assignment, I am trying to classify whether a tweet is talking about social, political, or religious issues.

Collecting Dataset

The dataset is collected by using Twint (Twitter API). I choose to analyze the category from tweets of @tirtoid account, because it's an online media that actively tweets about social, political, and religious issues, and mostly they use standard Indonesia language in the tweets. The corpus contains 150 latest tweets from @tirtoid that will be labeled with 3 categories: social, political, and religious.

- Social category (sos) : talking about social, lifestyle, civilian, and health
- Political category (pol) : talking about politics, government, and policy
- Religious category (agm) : talking about religious figures, religious traditions, and religious places

The distribution of categories can be seen in figure below:



Majority of tweet is talking about social issues (66%), then politic issues (30%), and last is religious issues (4%).

Preprocessing

First, case folding is done to transform the text into lowercase, removing numbers, removing special characters, and removing whitespace at the beginning and end of the text. Then, the text is tokenized using TweetTokenizer from nltk library, to produce tokens of words. After that,

I removed tokens that contains ‘http’ or ‘pictwitter’ in it. I also remove stopwords from the tokens, using stopwords from nltk corpus, so that we can focus on important words only. Last, I do stemming, using StemmerFactory from Sastrawi, to transform each word into its basic form. It’s done to match all word variations to bring up the most relevant documents. To prepare for the modelling, I split the data into 70% data train and 30% data test.

Modelling and Evaluation

For modelling, I build a pipeline model that consists of a vectorizer and a classifier. I’m using Tfidfvectorizer because I want to compute the tf-idf scores within the training dataset. While for the classifier, I’m using Multinomial Naïve Bayes classifier, because it is suitable for text classification with word counts as features and works well on fractional count like tf-idf.

For the performance evaluation, I will use F1-Score since target class is imbalanced. The hyperparameter for vectorizer and classifier is tuned with the validation data using RandomizedSearchCV to find the most optimal parameters with 50 iterations.

Optimal parameters: `{'vect__use_idf': True, 'vect__sublinear_tf': False, 'vect__smooth_idf': True, 'vect__norm': 'l1', 'vect__ngram_range': (1, 4), 'vect__min_df': 5, 'vect__max_df': 0.25, 'vect__analyzer': 'word', 'clf__fit_prior': True, 'clf__alpha': 0.5}`

Then, I do cross validation with 5 folds and get the mean and standard deviation of F1-Score and the result can be seen in picture below.

```
print(f'CLF Score: {clf_score.mean()} ± {clf_score.std()}')

CLF Score: 0.7217940001024847 ± 0.06419009275710899
```

Then, I will do the classification by using the optimal parameter and got the result as below.

	precision	recall	f1-score	support
agm	0.0000	0.0000	0.0000	1
pol	0.8889	0.5333	0.6667	15
sos	0.7778	0.9655	0.8615	29
accuracy			0.8000	45
macro avg	0.5556	0.4996	0.5094	45
weighted avg	0.7975	0.8000	0.7774	45

From the result, we can see that the classifier has done a good job on ‘sos’ and ‘pol’ class, while for ‘agm’ class, the classifier got it wrong. I think this is because ‘agm’ class distribution in the dataset is very low (6%). To tackle the problem, maybe we can add more ‘agm’ class to the corpus and do oversampling technique.