Text Classifier (Real versus Fake)

Name: Anshita Saxena, Matricule Number: 20240044

1 Problem Setup

The primary goal of this study is to develop a text classification model that can distinguish between "real/facts" and "fake" data about animals. This problem is treated as a binary text classification problem. The practical implications of this task are to ensure the accurate information published in textbooks, social media, and websites. This problem is limited to using text data instead of having several types of data including video or image. The research questions in hand are the possibility of using linear classifiers to distinguish between real and fake data and whether the choice of linear classifier matters or not.

2 Dataset Generation and Experimental Procedure

ChatGPT 3.5 is used to generate the dataset. I have generated the samples for an animal "Duck". The other animal samples were taken by the students who shared their datasets (bumblebees- Vicknesh Suresh, elephant- Felis Sedano Luo, capybaras- Jonathan Colaço Carr, dolphins- Mohamed Elahmedi). I have used the prompts- for facts: "Please give me 100 facts about duck and please mention "duck" word in every fact.", and for fakes: "Please give me 100 fake about duck and please mention "duck" word in every fake.". As TA suggested to not find the animal name based on the animal description as this will be another machine learning project using embeddings or pre-trained models. Hence, putting the animal name in every fact and fake and then extracting the animal name from animal_description and creating a new column called animal_type is a limitation of the setup, therefore, this dataset has three columns namely animal_type, animal_description, and labels. To avoid the data class imbalance problem, 100 samples for five animals as facts and fakes, represented as 1 and 0 respectively were put in the dataset.

We split the complete dataset into training, and testing in the ratio of 80% and 20%. Then, training set was split into training and validation in the ratio of 80% of the 80%, and 20% of 80% respectively after shuffling the dataset with the "stratify" argument on animal_type and labels which ensures the correct distribution. After splitting the dataset, preprocess the train, validation, and test sets separately using a tokenizer. Experiments were performed in two phases. In the first phase, get the most favorable pre-processing parameters on train and validation datasets using different combinations and running lots of experiments(around 863 experiments). In the second phase, after finding the best accuracy from this combination, perform a grid search(split train and validation datasets in many folds) for model hyperparameter selection for each algorithm using train and validation and find the accuracy on unseen data (test set).

3 Parameter Setting Range

The experiments were performed for the four different classifiers namely Bernoulli Naive Bayes(BNB), Multinomial Naive Bayes(MNB), Logistic Regression(LR), and Support Vector Machine(SVM). Preprocessing techniques are selected among the choices from punctuation, stop words, lowercase, stem Porter Stemmer, Snowball Stemmer, lemmatization, Bag_of_words, Bag_of_words with TfIdfTransformer (equivalent to TfIdfvectorizer=Bag_of_words_tfidftransformer) with n-grams.

Lower	Punctuation	Stop words	Stem/Lemmatize	vectorizer	ngram	Classifier	Accuracy
True	True	True	Lemmatizer	Bag_of_words	(1, 1)	BNB	91.875
True	True	True	Porter Stemmer	Bag_of_words	(1, 2)	LR	92.500
True	True	True	Snowball Stemmer	Bag_of_words_tfidftransformer	(1, 2)	MNB	95.625
False	True	True	Snowball Stemmer	Bag_of_words_tfidftransformer	(1, 2)	SVM	94.375

Table 1: Best preprocessing techniques with default parameter of models in the first phase of experiments.

4 Results

The accuracy was 50% with no punctuation and any sets of other choices among lowercase, punctuation, stop words, vectorizer, ngram, and classifiers. Accuracy was around 75% for BNB, around 80% for MNB, and around 85% for LR and SVM for only bigrams were present with stemmers which was increased to 82% for BNB, 88% for MNB and 87% for LR and SVM when only using lowercase, punctuation, and no stop words with lemmatize. Accuracies varied from 88% to 91% for all the classifiers using only unigrams. Accuracy varied from 91% to 94% for unigram_and_bigram range. A combination of all 863 experiments is stored inside accuracies_pre_processing.txt file which is created after running the code-script.

We used train and validation sets to find the best hyperparameters for all the classifiers using GridSearchCV as I checked with TA after finding the best pre-processing techniques as shown in table 1. The best grid search model hyperparameters with their best scores and accuracy gained on the test set are given in Table 2 and are stored in best_grid_parameters_accuracies.txt and final_results.txt files after running the code-script(a1.py).

As per TA's suggestion, I kept the results in the table below to not exceed the written pages limit of 1.5 pages and with a table limit of 2 pages. Multinomial Naive Bayes with the parameters given in Table 2 was the best model and highlighted in bold.

5 Limitation

The conclusions drawn from the experiments are based on the specific dataset which has some patterns and features that can be distinguished from other datasets with different content or domains. Therefore, generalizing the findings to all possible types of real vs. fake facts without considering dataset-specific nuances would be unreasonable. Secondly, the feature engineering process consists of test processing techniques, feature extraction, and feature representation tailored to the dataset. Hence, it is not reasonable to apply the same set of features and preprocessing techniques on all types of text data regardless of domain or language. Thirdly, the conclusion based on the performance of the different models on a given dataset is informative and generalizes well on similar tasks and datasets, however, further adaptation must be made to all other datasets. Finally, different languages because of the distinct linguistic features and nuances, and domains require further fine-tuning of the trained model. Hence, the trained model can not be applied directly across all domains and contexts.

6 REFERENCES

1. Machine Learning, NLP: Text Classification using scikit-learn, python and NLTK.

Classifiers	Preprocessing Parameters	Model Parameters	GridSearchCV score	Test set Accuracy
BNB	{Lowercase:True, Punctuation:True, Stop words:True, Stem/Lemmatize:Lemmatizer, vectorizer:Bag_of_words, ngram_range:(1, 1)}	alpha=0.01	92.875	92.0
LR	{Lowercase:True, Punctuation:True, Stop words:True, Stem/Lemmatize:Porter Stemmer, vectorizer:Bag_of_words, ngram_range:(1, 2)}	{'C': 5, 'max_iter': 100, 'penalty': 'l2'}	92.125	92.5
MNB	{Lowercase:True, Punctuation:True, Stop words:True, Stem/Lemmatize:Snowball Stemmer, vectorizer:Bag_of_words_tfidftransformer, ngram_range:(1, 2)}	alpha=0.01	**93.875**	**93.25**
SVM	{Lowercase:False, Punctuation:True, Stop words:True, Stem/Lemmatize:Snowball Stemmer, vectorizer:Bag_of_words_tfidftransformer, ngram_range:(1, 2)}	{'C': 1, 'loss': 'hinge'}	93.375	92.75

Table 2: Comprehensive table with Classifiers and their hyperparameters with accuracy on train, validation, and test sets.