SCHOOL OF INFORMATION STUDIES

SYRACUSE UNIVERSITY

PROJECT REPORT

IST 664 – Natural Language Processing| Guided by Professor Stephen Wallace

FAKE NEWS DETECTION

Submitted by:

Bhavish Kumar | Pranav Kottoli Radhakrishna | Tejas Dinesh Patil

TABLE OF CONTENTS

ABSTRACT

Fake news is defined as a made-up false story with the only intention to deceive or to mislead the people against either a political party or a group or an individual. Recently political competition, other socio-economic factors or sheer enmity/hatred have led to an increase in the popularity and spread of fake news just to polarize & divide the society or to tarnish the reputation of a group or an individual. At this present day, with the widespread effects of the large onset of fake news, it is very clear that humans are very poor detectors of fake news and end up blindly believing whatever they see or hear. Hence this is where we need the involvement of Artificial Intelligence/Machines to detect fake news. Using Language Processing and Machine Learning techniques, we will be able to flag every news as real/fake.

The objective of the project is to predict whether a news article is fake or not using NLP and machine learning. A machine learning model will be trained on historical data and evaluated on a test set. The goal is to ensure that the model generalizes well enough to make accurate predictions on data that it has never seen before. By accurately predicting whether a news article is fake or not we can effectively curb the effects of propaganda and misinformation that plagues the world today.

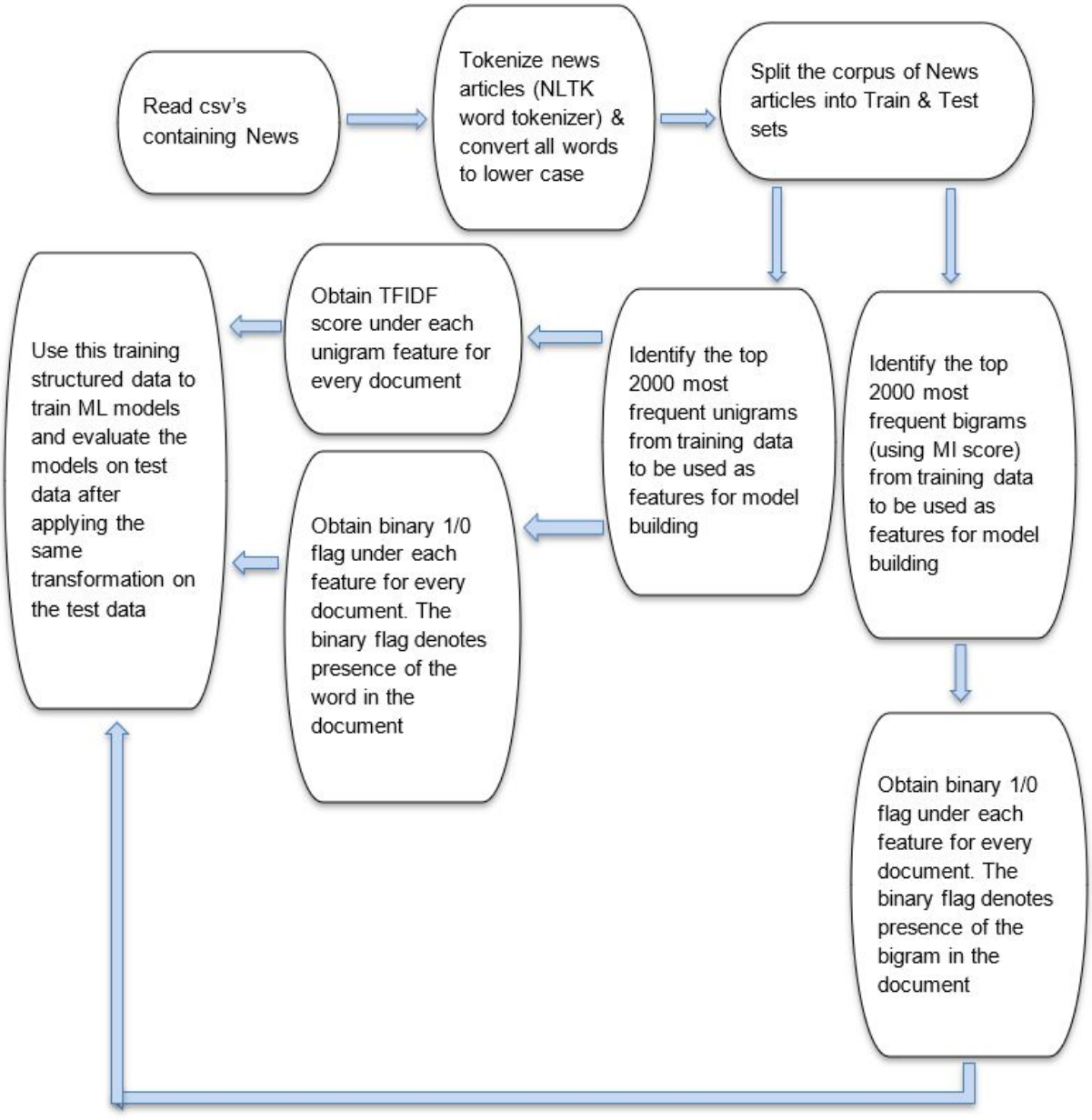
As per the problem statement, we have a separate collection of data with ‘True’ news and ‘Fake’ news. For 'Real' news and 'False' news, we have a different set of data according to the issue argument. There are four variables in each data set, namely 'Title,' 'Text,' 'Subject' and 'Date.' ‘Title’ variable contains the headline of the news article. The ‘Text’ variable contains the textual information presented in the article. The ‘Subject’ variable represents the domain to which the news article belongs, and the ‘Date’ variable shows the date on which the article was published. This dataset was obtained from the kaggle website. [Link to data set].

1. RESEARCH QUESTION

The aim of implementing this project was to answer the following questions:

a. Can we accurately classify every news article as Real/Fake using NLP & ML techniques?
b. Which NLP feature extraction technique will yield the best classification accuracy?
c. Which are the most important terms to determine if a news is real/fake

2. NLP SYSTEM



We have used the NLP system as shown above, to implement our project. The system consists of several steps as shown above. The first step is to read the data files into the required data structure such as dataframe, list or tuple. In this case we read and stored the initial dataset into a pandas dataframe, as the dataset consisted of multiple news articles and their respective attributes such as ‘title’, ‘text’, ‘subject’, ‘date’ and ‘is_fake flag’ (binary flag to denote whether the news is fake or not). Each row of the data frame represents one news article. The next step of the process was to tokenize the text using NLTK word tokenizer and convert all the words to lowercase in order to avoid duplicacy issues. Post this, the dataset was split into train and test sets to build supervised machine learning models using the train set and to evaluate the model performance using the test set. Three different NLP techniques were used to extract features and create three different training datasets to parallely perform supervised learning on each of the three datasets. The NLP techniques used for feature extraction are being discussed below. Each of the built models were evaluated on the test dataset using the Accuracy metric.

3. NLP TECHNIQUES USED

Three different preprocessing and feature engineering techniques were used to prepare the data for before building machine learning models:

1] Bag of Words: Unigram Approach

2] Bag of Words: Bigram Approach

3] Term Frequency - Inverse Document Frequency (TF-IDF) Approach

These techniques are discussed in detail below:

1] Bag of Words: Unigram Approach

We started with two different datasets containing fake news and true news respectively. Out of all the variables, 'title' (title of the news article) and 'text' (content of the article) were selected for this project as our main focus is to achieve the goal using language processing techniques. We added a target variable to both the datasets before merging them with respect to the columns. The combined dataset had a total of 44898 observations and an approximately balanced distribution of target variable.

In language processing techniques, First, we removed the '\n' (new line) characters from the title and text columns and then replaced the multiple spaces, tabs with single space. These steps were performed to achieve accuracy for word tokenizing, which is the next step for data pre-processing. Using the word tokenizer of nltk package, we tokenized each title and every sentence from each news article text. All the words were then converted to lowercase so that the same words in different cases should be treated as one word.

After tokenizing the words, it was seen that numerical figures, punctuation marks and other special characters were counted as individual words. Hence, to remove such tokens, which are wrongly considered as words, we defined a regex pattern that will help us filter out all the tokens with non alphabetic characters. These types of tokens usually tend to have a higher frequency and can affect the performance of machine learning models. Similarly, every language has some words which are basic building blocks for the sentences. They are repeated in the articles but cannot help in differentiating between true and fake news articles. Such words are called 'Stopwords'. There are some predefined stopwords that come with the nltk package for english language. We added a few of our own words to that list and filtered out such words from the tokenized words.

We need to split the data into train data and test data. This step is done to test the machine learning model on data that our model has not seen before. Also, we have done this splitting before extracting the features from the data. The features should be extracted only from the train data and not from the test data. This is done to provide leaking of information from test data to the training data. We split the data in the ratio of 0.70:0.30 for training and testing respectively.

Each of the title and text columns is a list of lists with major lists containing the number of lists equal to the number of observations. We flatten the lists and find the frequency distribution of words for both title and text column separately. Arranging the words in descending order with respect to the frequency distribution of the words. This gives us the most common words occurring in all the documents.

The next step is to transform the data with the most common words as features of the dataset and each cell containing a binary value showing if the corresponding document contains the given feature word. We extract 500 feature words for 'title' and 1000 feature words for 'text'.

Machine Learning

After completing the text pre-processing, we need to build the machine learning models that will help in classifying true and fake news articles. Since we have a target variable, we will be using only the supervised classification algorithms. For the unigram features, we use 3 different kinds of model:

1. *Logistic Regression (Regression based classification)*

   Being a regression algorithm, logistic regression with multiple input variables, tries to fit a linear hyperplane such that the sum of squared error is minimum for the fitted hyperplane. Then it uses the sigmoid function to map the output of a linear combination of all variables and weights between 0 and 1. A threshold is set between 0 and 1 (generally 0.5), to separate two classes.

   Regularization: Generally, we tune hyperparameters of any machine learning algorithm to fine tune the machine learning model, reduce the overfitting, etc. In logistic regression, we do regularization to avoid the overfitting and so we have regularization parameters.

   Regularization parameters:

   1] lambda: This parameter controls the intensity of regularization. It means that a penalty term is added to cost function in logistic regression. This added penalty term helps in reducing the coefficients of less important features, so that they will not affect the predictions by much.

   2] alpha: This parameter is for elastic net regularization where we use L1 and L2 regularization simultaneously. alpha acts as a slider between L1 and L2 regularization. L1 regularization can make the coefficients of variables completely 0, hence making them dead variables. L2 regularization also reduces the coefficients but cannot make it completely 0. For this project we haven't used elastic net regularization.

2. *Naive-Bayes Classifier (Probability based classification)*

This classification algorithm used Bayes theorem to predict the probability of an observation belonging to a class, where the class represents the target variable, in this case, either 1 or 0. It is based on a naive assumption that all the 'X' variables in data are independent of each other. It combines the prior knowledge [P(Y)] with evidence [P(X)] with the help of likelihood probability [p(Y|X)]. Note, here 'Y' represents the target variable class and 'X' represents the given observation.

3. *Random Forest (Tree based classification)*

Random forest is a tree based classification algorithm and it is made up of multiple decision trees. In random forest, all the decision trees work in parallel i.e. simultaneously the data provided to every decision tree. At the end, the final answer is chosen by majority voting between all the trees or by the weightage method. Unlike boosting methods, it has fixed probability distribution. First, we ran the baseline model and in the next step we fine tuned the hyperparameters for the random forest model.

1] *'criterion'*: Each tree in random forest progresses by creating a split. For that the model has to decide which feature to use for splitting and and decide the split value as well. This is done by calculating the impurity of the split. Impurity can be measured by either 'GINI Index' or 'Entropy'.

2] *'min_samples_leaf'*: This value tells the minimum number of samples needed to be there in the leaf node of any tree. If the number of samples in a node goes below this number, the algorithm stops the further splitting of that node.

3] *'min_samples_split'*: This value tells the minimum number of samples needed in any node to split it further.

4] *'max_depth'*: This parameter is the value of maximum depth (levels of node splitting) a tree can achieve.

5] *'max_features'*: It means number of columns selected randomly for each decision tree in the random forest. Ideally, to calculate *max_features* this formula is used:

max_features = [log_to_the_base(ncol) + 1]

6] *'n_estimators'*: This parameter is the count of the number of decision trees to include in our random forest model.

2] Bag of Bigrams approach :

In this approach, the nltk word tokenizer was followed by splitting the dataset into train and test. From the training dataset corpus, a bag of bigrams was obtained using the nltk.collocations package and the BigramCollocationFinder function. Out of all the bigrams from the corpus, the bigrams containing the stop words ('a', 'an', 'the' etc.) and special characters ('#', '!' etc.) were

removed. The stopwords present in the nltk stopwords corpus, along with other stop words were removed by applying a lambda word filter. The special characters were identified using a regular expression and the bigrams containing them were removed. Out of the remaining bigrams, the top 2000 most frequent bigrams were identified using a *Mutual Information score* (MI score). The MI score, computes the probability of 2 words occurring in a sequence. It is a log ratio of probability of 2 words occurring together as a joint event to the ratio of the product of the probability of them occurring individually. This score helps to identify if co-occurrences are a result of pure chance.

MI score = $\log_2(P(x,y)/(P(x).P(y)))$

These 2000 bigrams were used as features for the train and test dataset, to perform supervised machine learning. A bigram-document matrix was created (similar to Term Document Matrix), where the data frame was filled with binary 1/0 flags under each bigram feature corresponding to every document, where the flag denotes the presence of the bigram in the document. Below is a sample example of the bigram-document matrix dataset.

Eg: In the below dataset, only the bigram (‘roller’, ‘coaster’) is present in news article 1 and the only the bigram (‘cerebral’, ‘palsy’) is present in news article 2.

	(‘roller’, ‘coaster’)	(‘berkshire’, ‘hathway’)	(‘cerebral’, ‘palsy’)
Document 1 (News article 1)	1	0	0
Document 2	0	0	1

Using the training & test dataset of the above shown format containing 2000 bigrams as features, the following Machine Learning models were built and evaluated:

a. *Gaussian Naive Bayes Model:* Naïve Bayes is from the family of Probabilistic classifiers which is based upon Bayes theorem. A Naive Bayes model multiplies several different calculated probabilities together to identify the probability that something is true, or false. Naive Bayes is highly scalable and can handle both continuous and discrete values which makes it suitable for tasks involving such a high amount of data with so many different features. This model was trained on the training dataset with the ‘is_fake’ flag as the target variable, using the GaussianNB() function of the sklearn library and was evaluated on the test dataset (Hold Out Method of performance evaluation)

1. Training Gaussian Naive Bayes model and evaluating the model performance

```
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

b. *Logistic Regression Model:* Logistic regression is a machine learning model that fits a linear decision boundary between the positive and negative samples. This linear function is then passed through a sigmoid function to calculate the probability of the classes. Logistic regression is an excellent model to use when the features are linearly separable. One advantage of logistic regression is the model is interpretable — i.e. we know which features are important for predicting positive or negative. This model was trained on the training dataset using the LogisticRegression() function of sklearn and was evaluated on the test dataset.

2. Training Logistic Regression model

```
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
```

```
logr_pipe = make_pipeline(StandardScaler(), LogisticRegression(solver='lbfgs'))
logr_pipe.fit(X_train, y_train)
```

c. *Random Forest Model:* Random Forest is an ensemble tree-based learning algorithm. The Random Forest Classifier is a set of decision trees from a randomly selected subset of training set. It aggregates the votes from different decision trees to decide the final class of the test object. This model was trained on the training dataset using the RandomForestClassifier() function of sklearn and was evaluated on the test dataset.

3. Training Random Forest Classifier Model

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=1000, max_features=5, random_state=16)
results = cross_val_score(model, X_train, y_train)
print(f"Accuracy: {round(results.mean()*100, 2)}%")
```

d. *Gradient Boosted Trees Model:* Gradient Boosted Tree is an ensemble learning method which follows the wisdom of the crowd principle to make classification decisions by considering the output of several decision trees that are built sequentially. The final model is produced by taking the weighted average of predictions made by each base classifier. The GBM also uses the splitting criteria as Gini Index or Entropy similar to a decision tree, as GBM is nothing but multiple decision trees built sequentially. This model was trained on the training dataset using the gbm() function of sklearn and was evaluated on the test dataset.

4. Training Gradient Boosted Trees model

```
from sklearn.ensemble import GradientBoostingClassifier as gbm
model_gbm = gbm(n_estimators=1000, random_state=16)
results = cross_val_score(model_gbm, X_train, y_train, cv=3)
print(f"Accuracy for GBM: {round(results.mean()*100, 2)}%")
```

The performance scores of each of the models will be discussed in the Results section below.

3] tf-idf :

This technique involves using term frequency-inverse document frequency. Where term frequency refers to the frequency of a phrase in a document and inverse document frequency refers to the logarithmically scaled inverse fraction of the documents that contain the word (total number of documents divided by the number of documents containing the term). Finally, we take the product of the two to obtain the tf-idf score of a given word, for a given document.

The data contains two text based columns namely text and title, both of these columns were used with this technique. Before calculating the tf-idf, we preprocess the data by removing stopwords and converting words to lowercase. We split the data into training and testing sets. We then calculate the tf-idf for the top 2000 words by term frequency in the training set. This is done due to memory constraints. Finally, the Random Forest, Gradient Boosting and Naive Bayes models are trained using the training data. The tf-idf is calculated for the testing data and this data is used to test the models.

4. RESULTs INTERPRETATION and INSIGHTs

Firstly, the bigram method produced the poorest results with accuracies ranging from 58 to 67 percent (Accuracy of 58% for Gaussian Naive Bayes model; 66.6% for Logistic Regression model; 67% for Random Forest model & 65.4% for Gradient Boosted Trees model).

This relatively poor performance with the bigrams method can be attributed to the relatively low frequency of occurrence of each bigram which resulted in a highly sparse train and test dataset matrices.

The tf-idf and unigram methods produced accuracy scores ranging from 92 to 100 percent. It produced similarly high scores for precision, recall and f1 implying that there is no imbalance in true positive versus false positive predictions. (Unigram Method: Accuracy of 93% for Naive Bayes Classifier; 99% for Random Forest Model & 99% for Logistic Regression Model).

The above results imply the existence of certain keywords which can be used to detect whether a news article is fake or not. We believe the model must be retrained periodically as these keywords may evolve over time.

By looking at the feature importances provided by the Random Forest and Gradient Boosting models, we can see that the actual text of the article is more important when it comes to determining whether news is fake or not. However, the title also plays an important role. Additionally, terms related to politics such as trump, washington and gop have a big impact and are among the most important.

5. POTENTIAL APPLICATIONS

The rise of the internet over the past few years has made way for many new news applications and websites. With that being said, many applications and websites (news related), which have not been officially established, publish news articles which are not necessarily true. This has contributed to a spike in fake news, with more news available than you need. Not everybody can be an expert at spotting fake news. In today's world it has become very difficult to trust any news article that you read.

An application can be developed that extracts filters out the fake news articles and only displays true news articles to the reader. We have used three feature extraction techniques in this project. In the application, a prediction can be made using all the possible combinations of feature extraction techniques and machine learning algorithms. The final verdict will be considered by majority voting technique. Since, all the accuracies we obtained were more than 50%, this technique will increase the overall prediction accuracy of the application.

6. COMPLEXITY OF PROBLEM

The most time consuming part of the project was to identify the required number of features sufficient for training a model to yield high accuracy. A large number of features was resulting in very high time & space complexity, and with the limited number of resources available on our systems, we were not able to create Training & Test Datasets (term-document matrices) with a very high number of features. Moreover, keeping a large number of features was resulting in extremely sparse datasets. On the other hand, having a very small number of features was resulting in poor model performance as it was giving low accuracies. Hence using an optimum number of features without having to compromise on accuracy was the key and was challenging.