CS-579 Online Social Network Analysis

Final Project Report

On

Fake News Detection

Name of Student	CWID
1. Vivekanand Reddy Malipatel	A20524971
2. Shoaib Mohammed	A20512491

Under The Supervision
of
Prof. Kai Shu

College Of Computing
Illinois Institute of Technology
Chicago, Illinois.

April 2023

Table of Contents

1. Introduction

2.Literature Review

3.Project Design and Methodology

4.Results

5.Conclusion

6.References

Introduction

Social media has revolutionized the way people consume news and information. However, the ease and speed of dissemination have also made it easy for fake news or misinformation to be produced and spread. This has created a serious problem because fake news/misinformation can negatively impact individuals and society. The spread of misinformation can break the authenticity balance of the news ecosystem, intentionally persuade consumers to accept biased or false beliefs, and change the way people interpret and respond to real news and information. As a result, detecting fake news and misinformation in social media has become an important task.

Task:

The task is to classify a news article as agreed, disagreed, or unrelated to a given fake news article. This is done by providing the title of the fake news article and the title of a coming news article to the participants. Participants are then asked to classify the coming news article as either agreed (talking about the same fake news as the given article), disagreed (refuting the fake news), or unrelated to the given article.

Given the title of a fake news article A and the title of a coming news article B, participants are asked to classify B into one of the three categories:

- agreed: B talks about the same fake news as A.
- disagreed: B refutes the fake news in A.
- unrelated: B is unrelated to A.

Literature Review

Fake news and misinformation have become a significant concern for individuals and society. Machine learning and deep learning algorithms have been extensively used to classify fake news by analyzing the features of the news article, such as sentiment, polarity, and tone. Researchers have developed various approaches for detecting fake news, including content-based, stance-based, and propagation-based methods. Deep learning algorithms analyze the semantic and syntactic features of the news article, while NLP techniques analyze linguistic and stylistic characteristics. Each approach has its strengths and weaknesses, and the development of more effective detection techniques will become increasingly important as the problem of fake news and misinformation continues to grow.

Project Design and Methodology

The methodology of this project involves using supervised learning techniques to train a classifier that can distinguish between real news and fake news based on the titles of news articles. The approach involves providing the model with labeled training data, which includes pairs of news article titles and their corresponding labels (agreed, disagreed, or unrelated).

For the Fake News Classification project here are the step which have been Followed :-

1. Import train.csv files
2. Data cleaning using Stemming Process
3. Split the dataset into Training data and Test data
4. Tf-idf Vectorization to convert data into numerical data form
5. Training Machine Learning Models for Classification

1. Import train.csv and test.csv files

We have used the '**readdata.py**' file to import the original test and train datasets (**train_original.csv** and **test_original.csv**). We have used Pandas library functions to check for null values and duplicate values. We found that there were no null/duplicate values.
Next we have tried to get a brief description of the Train Dataset using the functions provided in the pandas library.

2. Data cleaning using Stemming Process

Our Next goal was to pre-process the data. This includes removing punctuations, stopwords, converting into lower case and stemming the words.

In the '**readdata.py**' file, we have used NLTK libraries in python to achieve this preprocessing. We have used nltk.corpus.stopwords library to import the stopwords and nltk.stem.SnowballStemmer library for the stemming process.
We have processed both the train_original and test_original dataset in the above mentioned processing. The process has generated few empty strings as the converted titles (**title1_clean** and **title2_clean**), which need to be omitted before saving the pre-processed data. We have removed the empty string rows and saved the processed dataframe with the original data in **train.csv** and **test.csv** files.

3. Split the Train dataset into Training data and Validation Data

Before Training the machine learning models, In each of the mode training process we have split the Train dataset into a 70-30 split. And we have used the 30% part of the split to validate the model's accuracy.

4. Tf-idf Vectorization to convert data into numerical data form

Before Training the machine learning models, In each of the mode training process we have used TF-IDF vectorizer to transform the text data into numerical features that can be used for training the models. We have used the TfidfVectorizer class from the scikit-learn library for this purpose.
We have set the vectorizer to `(max_features=5000, ngram_range=(1,2), stop_words='english')` to achieve a better result based on our study of the given datasets.

We have processed both train and test datasets in the vectorizer.

5. Training Machine Learning models for classification

Now with all the preprocessing we have we have started training different machine learning models that have significance in classifying the text based

datasets. With our study, we found that the below four algorithms are suitable for the classification.

a. Naive Bayesian Multinomial Classification
b. Decision Tree Classification
c. Random Forest Classification
d. Radial based Simple Vector Machine Classification

We have tried different combinations of parameters to feed the classifications in a loop and checked the accuracies they generated on the validation sets. Then we have used predict functions of the models to predict the labels of the Test Dataset.

We have created separate python files for each classification and stored the predicted Test Dataset values as csv files. We have named the predicted label column as 'predicted_label'.

The Below are the Classification reports and Accuracies of all the trained machine learning models.

a. Naive Bayesian Multinomial Classification

```
[Running] python -u "/Users/shoaib/Desktop/OnlineSocialNetworkAnalysisProject-2-main/NaiveBayesClassification.py"
              precision    recall  f1-score   support

      agreed       0.59      0.38      0.46     22449
   disagreed       0.49      0.19      0.28      1907
   unrelated       0.75      0.88      0.81     52566

    accuracy                           0.72     76922
   macro avg       0.61      0.48      0.52     76922
weighted avg       0.70      0.72      0.69     76922

Model Accuracy: 71.68%
71.68

[Done] exited with code=0 in 14.923 seconds
```

b. Decision Tree Classification

```
[Running] python -u "/Users/shoaib/Desktop/OnlineSocialNetworkAnalysisProject-2-main/DecisionTreeClassifier.py"
              precision    recall  f1-score   support

       agreed       0.68      0.40      0.50     22449
    disagreed       0.47      0.28      0.35      1907
    unrelated       0.76      0.91      0.83     52566

     accuracy                           0.75     76922
    macro avg       0.64      0.53      0.56     76922
 weighted avg       0.73      0.75      0.72     76922

Model Accuracy: 74.56%
74.56

[Done] exited with code=0 in 59.163 seconds
```

c. Random Forest Classification

```
[Running] python -u "/Users/shoaib/Desktop/OnlineSocialNetworkAnalysisProject-2-main/RandomForestClassifier.py"
              precision    recall  f1-score   support

       agreed       0.83      0.69      0.75     22449
    disagreed       0.76      0.26      0.39      1907
    unrelated       0.86      0.94      0.90     52566

     accuracy                           0.85     76922
    macro avg       0.82      0.63      0.68     76922
 weighted avg       0.85      0.85      0.84     76922

Model Accuracy: 84.9%
84.9

[Done] exited with code=0 in 298.137 seconds
```

d. SVM classification

```
[Running] python -u "/Users/shoaib/Desktop/OnlineSocialNetworkAnalysisProject-2-main/SVMClassifier.py"
Model Accuracy: 85.26%
85.26

[Done] exited with code=0 in 46243.012 seconds
```

In this process we have achieved the highest accuracy of **85.26%** with the

validation set with the **Radial based Simple Vector Machine Classification Model**. The Predicted labels output for the classification is stored in the '**SVMpredictedtest.csv**' file.

Now we feed the produced csv file to '**submission.py**' file to generate the final output.

Results

Working with the Train dataset, we have seen the dataset is very imbalanced and more leaned towards the "unrelated" label. We have not applied the undersampling techniques as we have very little data distribution towards the other labels, which can affect the accuracy of the model we train.
Considering the satisfactory Precision, Recall and Accuracy metrics of predictions of each classification, we have concluded that SVM has the best classification of the four models that we have used.

Hence, We are considering the SVM model that we have trained, to predict the labels of the test data.

In the Python file '**submission.py**', We have used the results generated from the Simple Vector classifier and converted it into a '**submission.csv**' with the suggested columns 'id' and 'label'.

Conclusion

With all the Imbalanced input data, data preprocessing and accurate enough classifications, we were able to predict the labels of the given test data and store it in the **submission.csv** file.

Our highest accuracy on the validation data was achieved by the Simple Vector Classification model with “Radial Based Function” Kernel parameter. For the classification results to be reproducible, we have set the random seed to all the classifications.

We are still working on tuning the hyperparameters and different data preprocessing techniques to achieve the best results from our chosen SVM classification.

References

- IIT CS-484 - Introduction to Machine Learning Course
- https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a
- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
- https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
- https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- https://datascienceplus.com/radial-kernel-support-vector-classifier/
- https://towardsdatascience.com/nlp-in-python-data-cleaning-6313a404a470
- https://medium.com/@utkarsh.kant/how-to-split-your-dataset-into-train-test-and-validation-sets-17f40e98dfd0