

Ateneo De Davao University
School of Arts and Sciences
Computer Studies Department



**A Study on Fake News Detection Model and Implementation with Python and
Machine Learning**

Sungbin Lee

March 2022

1. Introduction

Recently, “fake news” has been attracting attention from all over the world, regardless of field. According to BuzzFeed, 17 of the top 20 fake news related to the US presidential election were falsified, and Facebook's response rate to fake news outperformed mainstream media organizations' Facebook response to real news. These fake news are being mass-produced for political and economic purposes, and as a result, individuals, companies, and countries are also paying a huge social cost. While the existence of fake news is not new, it has recently received a lot of attention due to the sheer volume of misinformation surrounding COVID-19. Through this study I will acquire basic knowledge in this field and at the end create a machine learning model using Python and NLP to successfully detect fake news.

2. Review of Related Work

2.1 NLP

Natural Language Processing (NLP) refers to the work of analyzing the meaning of natural language so that the computer can process it. In recent years, technology for natural language processing documents using machine learning or deep learning covers all fields that traditionally belong to text mining, so its scope has been expanded. (Brownlee, 2019).

2.2.1 Classification

Classification is a branch of supervised learning that is used to predict one of several predefined classes for a given input data. As one of the most used fields in machine learning, there are various algorithms and application cases. In implementation, I will be using passive-aggressive classification because it is one of the available incremental learning algorithms and is very easy to implement. Passive-aggressive algorithms are a suite of online learning algorithms for both

classification and regression proposed by Crammer et al (2006). The idea is very simple and the performance has proven to be superior to many other alternative methods such as Online Perceptron and MIRA by the authors.

2.3 Fake News Detection

Fake news detection research is largely divided into content-based detection research and context-based detection research (Bondielli & Marcelloni, 2019). Content-based detection research has been mainly proposed in a form combined with a machine learning-based classification model, and is a method that uses the linguistic characteristics of the content and the features appearing in the content format. In context-based detection research, there is a method that uses information about the distributor or source of fake news, and a method that uses network characteristics. Fake news detection studies are mostly content-based studies, and specifically, studies using linguistic features. TF-IDF was used the most as a method to extract the linguistic features of fake news, Jung (2019) showed that sentiment analysis and text analysis using syntactic features were combined for the first time to detect fake news. In the case of Wang (2017) a successful fake news detection model was created by applying word2vec to a short fake news dataset.

2.4 Topic Modeling

Topic Modeling is one of the application fields of text mining, and it is a representative application technology that is most actively used with tangible results in various fields (Lee & Kim, 2018). Topic modeling is performed by grouping similar sentences based on the frequency of terms included in each document, and then extracting key terms representing each group to suggest a set of topic keywords for the group (Document used here means a broad concept including document, title, summary, body, comment, etc.). The main theoretical backgrounds are the Vector Space Model (Albright, 2016; Salton et al, 1975) and TF-IDF (Weiss et al, 2010). TF-IDF is short for term frequency-inverse document frequency, and is done in such a

way that a low value is assigned to a general word that appears frequently in multiple documents, and a high value is assigned to a special word that appears in a specific document. Each document has as many dimensions as the number of terms and is expressed as a vector with TF-IDF as a value. The TF-IDF value is calculated in the following way and is used in various fields (Lee & Kim, 2009).

$$\begin{aligned}
 TF-IDF(d, t) &= TF(d, t) \times IDF(t) \\
 TF(d, t) &= \begin{cases} 0 & \text{if } freq(d, t) = 0 \\ 1 + \log(1 + \log(freq(d, t))) & \text{otherwise} \end{cases} \\
 IDF(t) &= \log \frac{|d|}{|d_t|}
 \end{aligned}$$

2.5 Word Embedding

Word embedding is an expression that collectively refers to a method of representing a word as a vector, and is largely divided into a sparse expression and a dense expression. A sparse representation is a traditional vector representation method, such as Bag of word or TF-IDF. Bag of word is a method of evaluating and expressing words only based on appearance frequency without considering the order of words. TF-IDF, which is used in various studies including fake news detection research, basically evaluates words based on their frequency. TF-IDF has the possibility of overfitting because it generates a sparse matrix that expands the dimension by the number of individual terms appearing in the entire data. In addition, there is a limitation in not reflecting the contextual information of words and the direction of meaning between words. On the other hand, word2vec is a representative method corresponding to dense expression, which is a deep learning-based word embedding technique that reconstructs the linguistic context of a word with a dense vector expression (Mikolov et al., 2013). The word prediction method is divided into cbow and skipgram, and there are a plurality of hyperparameters such as the number of windows, the minimum frequency of words, and the number of processes.

3. Implementation

3.1 Libraries

The following libraries should be installed with pip:

```
pip3 install pandas
pip3 install sklearn
pip3 install numpy
pip3 install matplotlib
```

Importing all necessary libraries:

```
import itertools
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
```

3.2 Dataset

Having the right data set is one of the most important components of any data science. For this implementation, the dataset used includes many articles. These articles are classified as "REAL" or "FAKE" through preprocessing. This data will be used as training data so that the model can determine whether an article is providing factual information or fake news. In this case the following [Kaggle dataset](#) will be used as training data.

```
# Import dataset
df=pd.read_csv('./dataset/train.csv')

# Get the shape
df.shape

# Get the head
```

```
df.head()
```

Screenshot:

```
>>> df.shape
(20800, 5)
>>> df.head()
```

	id	title	author	\
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	

	text	label
0	House Dem Aide: We Didn't Even See Comey's Let...	1
1	Ever get the feeling your life circles the rou...	0
2	Why the Truth Might Get You Fired October 29, ...	1
3	Videos 15 Civilians Killed In Single US Aistr...	1
4	Print \nAn Iranian woman has been sentenced to...	1

I can see that the csv file contains a data set of 20800 rows and 5 unique features (columns) and I can see that the data set is divided into columns like id, title, author, text and label. What I'm interested in are the labels and text columns. The text column contains the content of the article, while the label column indicates whether the article is true or not. This is premade in binary format using '1's and '0'.

In general, this is a perfect way to represent these values, but for convenience I convert the '1' and '0' to 'REAL' and 'FAKE' booleans.

```
# Change the labels
df.loc[(df['label'] == 1) , ['label']] = 'FAKE'
df.loc[(df['label'] == 0) , ['label']] = 'REAL'

# Isolate the labels
labels = df.label
```

```
labels.head()
```

Screenshot :

```
>>> df.loc[(df['label'] == 1) , ['label']] = 'FAKE'
... df.loc[(df['label'] == 0) , ['label']] = 'REAL'
...
>>> labels = df.label
... labels.head()
>>> labels.head()
...
0    FAKE
1    REAL
2    FAKE
3    FAKE
4    FAKE
Name: label, dtype: object
```

Now I need to split the data set into two separate sets. 80% of the data will be used to train the model and the remaining 20% will be used as test data.

```
#Split the dataset
x_train,x_test,y_train,y_test=train_test_split(df['text'].values.astype('str'), labels,
test_size=0.2, random_state=7)
```

3.4 Building Vectorizer Classifiers

Now declare a TfidfVectorizer with stop words from English and allow document frequency up to 0.7. After the vectorizer, transform it on the training set and also transform it on the testing set.

```
#Initialize a TfidfVectorizer
tfidf_vectorizer=TfidfVectorizer(stop_words='english', max_df=0.7)
```

```
# Fit & transform train set, transform test set
tfidf_train=tfidf_vectorizer.fit_transform(x_train)
tfidf_test=tfidf_vectorizer.transform(x_test)
```

The PassiveAggressiveClassifier is to now be initialized. In order to incorporate it into the model, I will use the “tfidf_train” and “y_train”.

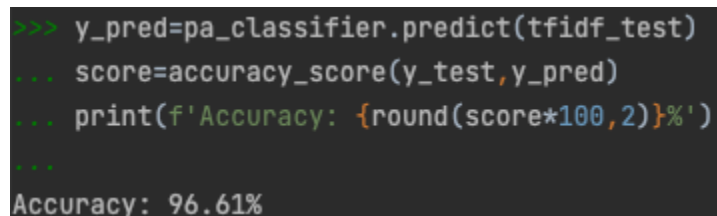
```
# Initialize the PassiveAggressiveClassifier and fit training sets
pa_classifier=PassiveAggressiveClassifier(max_iter=50)
pa_classifier.fit(tfidf_train,y_train)
```

3.5 Result

Finally, I am going to use a vectorizer to predict whether the article is reliable and calculate the accuracy of the model.

```
# Predict and calculate accuracy
y_pred=pa_classifier.predict(tfidf_test)
score=accuracy_score(y_test,y_pred)
print(f'Accuracy: {round(score*100,2)}%')
```

Screenshot:

A screenshot of a terminal window showing the execution of Python code. The code defines y_pred, calculates the accuracy score, and prints the result. The output shows an accuracy of 96.61%.

```
>>> y_pred=pa_classifier.predict(tfidf_test)
... score=accuracy_score(y_test,y_pred)
... print(f'Accuracy: {round(score*100,2)}%')
...
Accuracy: 96.61%
```

I was able to verify the accuracy of the model while performing the tests. I can see the accuracy, but I don't know how many times the prediction succeeded and failed. To access this information, I used a confusion matrix and also visualized it to understand intuitively. This can be easily done like this:


```

# Build confusion matrix
confusion_matrix(y_test,y_pred, labels=['FAKE','REAL'])

# Visualize the matrix
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        #print("Normalized confusion matrix")
    #else:
        # print('Confusion matrix, without normalization')

    #print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

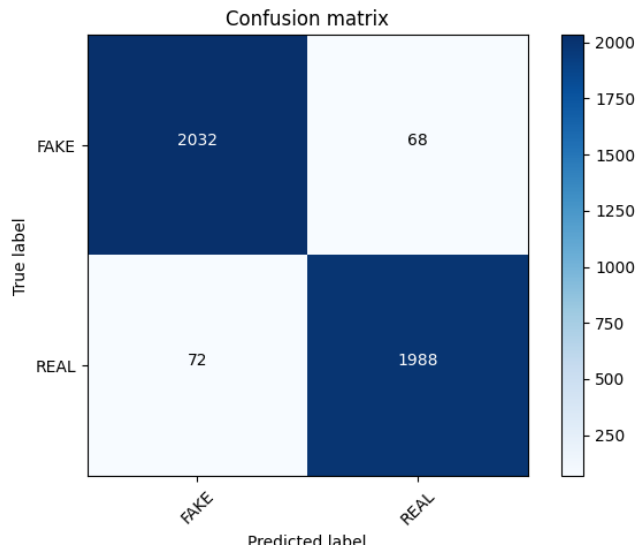
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

cm = confusion_matrix(y_test,y_pred, labels=['FAKE','REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
plt.show()

```

Screenshot:

```
Accuracy: 96.61%
>>> confusion_matrix(y_test,y_pred, labels=['FAKE','REAL'])
...
array([[2032,  68],
       [ 73, 1987]])
```



From the confusion matrix I can draw the following conclusions:

The model successfully predicted 2032 positives.

The model successfully predicted 1988 negatives.

The model predicted 68 false positives.

The model predicted 72 false negatives.

4. Conclusion

I implemented a model that detects fake news and showed an accuracy of 96.61%, but there is a limitation in that it is based only on trained data and well-organized data. On the other hand, our society is facing difficult situations such as Covid-19, invasion of Ukraine, and inflation. In these difficult times, it is important to further advance models such as fake news detection to effectively combat misinformation.

5. References

Brownlee, J. (2019, August 7). What is natural language processing? Machine Learning Mastery. Retrieved March 20, 2022, from <https://machinelearningmastery.com/natural-language-processing/>

Bondielli, A., & Marcelloni, F. (2019). Survey on fake news and rumour detection techniques. *information Sciences*, Vol. 497, pp. 38-55

Crammer K., Dekel O., Keshet J., Shalev-Shwartz S., Singer Y. (2006). Online Passive-Aggressive Algorithms, *Journal of Machine Learning Research* 7, pp. 551–585

Jung, H. (2019). Fake News Detection Using Content-based Feature Extraction Method. Master Thesis.

Wang, W. Y. (2017). liar, liar pants on fire: A new benchmark dataset for fake news detection. arXiv preprint arXiv:1705.00648.

Lee, D., Kim, Y., & Kim, K. (2018). Topic Based Hierarchical Network Analysis for Entrepreneur Using Text Mining. *The Journal of Society for e-Business Studies*, Vol. 23, No. 3, pp. 33-49.

Albright, R. (2006). Taming Text with the SVD, SAS Institute Inc.

Salton, G., Wong, A., and Yang, C. S. (1975). A Vector Space Model for Automatic Indexing. *Communications of the ACM*, Vol. 18, No. 11, pp. 613-620.

Weiss, S. M., Indurkha, N., & Zhang, T. (2010). *Fundamentals of Predictive Text Mining*, Springer.

Lee, S. and Kim, H. J. (2009) Keyword Extraction from News Corpus using Modified TFIDF. The Journal of Society for e-Business Studies, Vol. 14, No. 4, pp. 59-73.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, pp. 3111-3119.