INTEL-UNNATI INDUSTRIAL TRAINING PROGRAMME
Number 2 (2023), pp. 1–11

Team:Whitecap

MACHINE LEARNING

Fake News Detection Using Python and Machine Learning

Ashly Biju, Jiya Mary Joby , and Aiswarya Lakshmi

Saintgits Group of Institutions, Kottayam, Kerala

July 14, 2023

Abstract: With the advancement of social media and online platforms,the spread of fake news has become a major concern. Fake news simply means 'wrong information' that can have real world adverse effects and can be difficult to distinguish from real news.Therefore, it is a research challenge to validate the source, content and publisher of a news article for classifying it as genuine or fake. The existing systems and techniques are not efficient enough to accurately classify a given news based on its statistical rating.In this adverse situation machine learning techniques can be used to detect fake news.Machine learning plays an imperative part in categorizing news data and information, despite some limitations.In this paper,we propose a system for fake news detection that uses machine learning techniques implemented in python.We propose a dataset of fake and true news to train the proposed system.

Keywords: fake news,true news,tokenization, stemming, lemmetaization, vectorisation TFD-IDF, classification, Large Language Models, transformers, BERT

1 Introduction

In today's society,social media is the most easy and convinient way of sharing news to each other.So most of the news consumption by people is through different social media platforms. The term 'fake news' became more relevant after the advancement of social media platforms [2]. This fake news not just adversely affect an individual but it also affects the society as a whole. The right to information is basic right and it is more important that we get the true information. Thus, it is very important to stop the chain of fake news from the root itself. This is where our project will be beneficial. This project uses machine learning principles and algorithms to detect fake news which can be of great usage in the identification of correct information. In this paper it is seeked to produce a model that can

accurately predict the likelihood that a given article is fake news harnessing the power of machine language and python and thus prevent dissemination of fake news to an extent.

2 Libraries Used

In the project for various tasks, following packages are used.

```
NumPy
Pandas
Seaborn
NLTK
Matplotlib
BERT
Scikit-learn
```

3 Methodology

In this work two types of models are used. For the first part, various classical Machine Learning models are used. Among them the decision tree classifier is found to be better in terms of accuracy and other performance measures. Various stages in the implementation process are:

Data Loading: Loading the data for the Machine Learning task from reliabile sources.

Pre-processing & Data cleaning In this stage the loaded data will be cleaned and make ready for using Machine Learning algorithm. In text analysis, the text should be properly pre-processed.For pre-processing we use the nltk library.

Feature extraction: Term Frequency-Inverse Document Frequency (TF-IDF). Through this step calculate a score for each word based on its frequency in the document and its rarity across all documents.

Dataset Preparation: Split the dataset into training and testing sets.

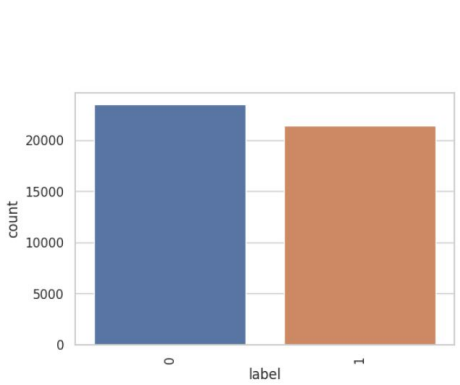
Model Training: Choose a classification algorithm, such as Naive Bayes, Logistic Regression, or decision tree and train the model using the training set & the extracted features.

Model Evaluation: Test the trained model on the testing set to evaluate its performance.Use the metrics such as accuracy, precision, recall, and F1-score to assess the model's effectiveness.

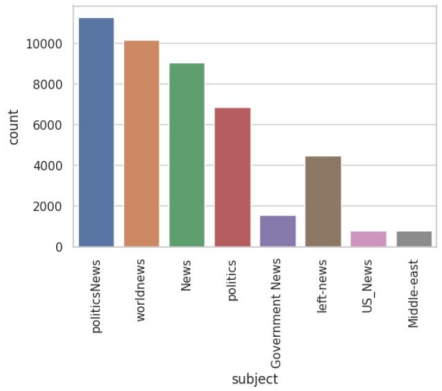
Model selection and reporting In this final stage, based on various performance measures the better model will be selected and report the model performance matrices.

4 Implementation

As the first step in the task, the two files- `fake.csv` and `true.csv` are loaded into the Intel's `DevCloud -OpenAPI`for the classification task using the url https://onlineacademiccommunity.uvic.ca/isot/wp-content/uploads/sites/7295/2023/03/News-_dataset.zip. The `.csv` files are turned into a `pands` DataFrame. Both the datasets are properly labelled and is concatenated to a single dataframe. The EDA on subjects and real and fake news status is shown in Figure 1. From Figure 1, it is clear that

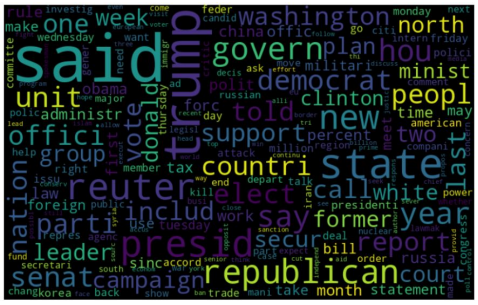


(a) Real Vs fake news count

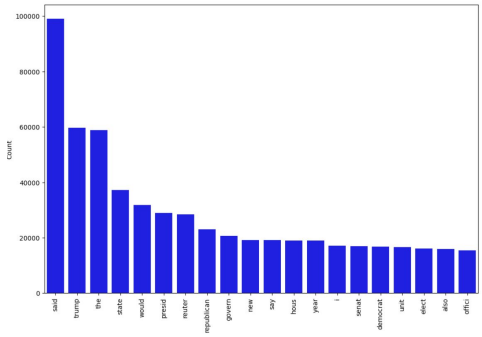


(b) Subject-wise news items in the database

Figure 1: Type and subject-wise Distribution of news items



(a) Word cloud showing cleaned news data



(b) Word count plot for cleaned data

Figure 2: Visualization of pre-processed data

the dataset contains almost equal number of real and fake news feeds and so sufficiently balensed in terms of the class. Further more the most dominating subjects are *political news*, *world news*, *news and politics*.
The pre-processing and cleaning stage is completed in four distinct and consecutive steps:

- *Tokenization:* Break the text into individual words or tokens.
- *Lowercasing:* Convert all tokens to lowercase to ensure case insensitivity.
- *Stopword removal:*Remove common words that do not carry much meaning.
- *Lemmatization:* Reduce words to their base or dictionary form to normalize the text.

The pre-processing stage is completed in $365.365\ sec$ in *Google Colaboratoy* and in $243.56\ sec$ in *Intel DevCloud*. The word cloud and word frequency plot of the cleaned data is shown in Figure 2.

In the feature extraction stage `CountVectorizer` and `TfidfTransformer` from `sklearn.feature_extraction.text` is used. The dimension of the extracted feature set is 44898×75749. For the model training, 33673 samples are used We have used intel© optimised algorithms for the project. Logistic regression, Naive Bayes classifier, Decision

Tree classifier and Passive-aggressive algorithms are used for classification of the fake news data. A second model is developed using the Bi-Directional Encoder Representations from Transformers (BERT). It uses Transformers to understand the contextual relation between words in a sentence/text. BERT Transformer generally has two mechanisms: An encoder that reads the text input and a decoder that predicts for a given task. The BERT model will learn the contextual language learning. BERT (Base) model with 12 layers of encoder stack is used in this study. The model crashed in `Google Colaboratory` (free version) even with one epoach. So the BERT model is trained in a local mini-workstation on `Anaconda` distribution.
Results of these implementations are discussed in the next section.

5 Results & Discussion

Popular classical Machine learning algorithms from the `Python` library `sklearn` is used for model training and testing. Summary of the results is shown in Table 1. Intel exten-

Table 1: Summary of Classification models tested on the fake news data

Model No.	Model Name	RMSE value	Precision	Accuracy	Recall	f1-score	Processing Time
1.	Logistic Regression	0.084	0.99	0.987	0.99	0.99	5.060 sec
2.	Naive Bayes	0.254	0.93	0.937	0.95	0.94	0.159 sec
3.	Decision Tree	0.070	0.99	0.995	1.00	1.00	33.66 sec
4.	Passive Aggressive	0.068	0.99	0.993	0.99	0.99	0.437 sec
5.	BERT Classifier	-	0.93	0.94	0.94	0.93	3.5 hrs
6.	Random forest	0.003	0.92	0.95	0.91	0.95	1009.75 sec

sion of sklearn libary (Source:https://intel.github.io/scikit-learn-intelex/) is used as the AI accelerator and the result is shown in Table 2. From Table 2, it is clear that there is sig-

Table 2: Summary of Classification models tested on the fake news data using intel© extension of sklearn

Model No.	Model Name	RMSE value	Precision	Accuracy	Recall	f1-score	Processing Time
1.	Logistic Regression	0.114	0.99	0.99	0.99	0.99	3.82 sec
2.	Naive Bayes	0.252	0.93	0.94	0.95	0.94	0.098 sec
3.	Decision Tree	0.065	1.0	1.0	1.00	1.00	26.98 sec
4.	Passive Aggressive	0.073	0.99	0.99	0.99	0.99	0.381 sec
5.	Random forest	0.003	0.92	0.95	0.91	0.95	989.25 sec

nificant acceleration when the intel© accelerated versions of `sklearn` library is used. The percentage improvements are 24.5%, 38.36%, 19.84% and 12.81% respectively in Logistic regression, Navie Bayes classifier, Decision tree classifier and Passive aggressive classifier algorithms. Using the `chi-square` feature extraction technique, a small set of 1000 samples based on feature importance is extracted. All the classification models discussed in the previous large model is trained and tested on the new small dataset. Summary of the model performance is shown in Table 3.

Table 3: Summary of Classification models tested on the feature extracted data

Model No.	Model Name	RMSE value	Precision	Accuracy	Recall	f1-score	Processing Time
1.	Logistic Regression	0.130	0.99	0.987	0.98	0.98	3.89 sec
2.	Naive Bayes	0.243	0.94	0.94	0.95	0.94	0.213 sec
3.	Decision Tree	0.079	0.99	0.995	0.99	0.99	7.457 sec
4.	Passive Aggressive	0.083	1.00	0.993	0.99	0.99	0.591 sec
6.	Random forest	0.003	0.92	0.95	0.91	0.95	1009.75 sec

6 Conclusions

The fake news detection task showed promising results using classical machine learning algorithms and the BERT model. Classical models, including Logistic regression, Naive Bayes classifier, Decision tree classifier, and Passive-Aggressive classifier, demonstrated comparable accuracy to BERT despite being simpler and faster to learn.Use of the Intel© AI accelerator substantially improve the inference time in classification. These algorithms leverage established mathematical principles and statistical techniques for efficient computation, making them suitable for real-time applications. While BERT excels in capturing semantic relationships, its computational intensity and training time can be limitations in resource-constrained scenarios. Overall, classical machine learning algorithms offer a practical alternative with comparable performance, making them preferable for applications prioritizing computational efficiency and learning time. BERT is suitable when abundant resources and high accuracy are paramount.

Acknowledgments

We would like to express our heartfelt gratitude and appreciation to Intel© Corporation for providing an opportunity to this project.First and foremost, we would like to extend our sincere thanks to our team mentor Siju Swamy for his invaluable guidance and constant support throughout the project.We are deeply indebted to our college Saintgits College of Engineering and Technology for providing us with the necessary resources,and sessions on machine learning. We extend our gratitude to all the researchers, scholars, and experts in the field of machine learning and natural language processing and artificial intelligence, whose seminal work has paved the way for our project. We acknowledge the mentors, institutional heads, and industrial mentors for their invaluable guidance and support in completing this industrial training under Intel© -Unnati Programme whose expertise and encouragement have been instrumental in shaping our work. []

References

[1] ALDWAIRI, M., AND ALWAHEDI, A. Detecting fake news in social media networks. *Procedia Computer Science 141* (2018), 215–222. https://doi.org/10.1016/j.procs.2018.10.171. The 9th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2018) / The 8th Inter-

national Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2018) / Affiliated Workshops.

[2] BAARIR, N. F., AND DJEFFAL, A. Fake news detection using machine learning. In *2020 2nd International Workshop on Human-Centric Smart Environments for Health and Well-being (IHSH)* (2021), IEEE, pp. 125–130.

[3] BIRD, S., KLEIN, E., AND LOPER, E. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[4] CHOWDHARY, K., AND CHOWDHARY, K. Natural language processing. *Fundamentals of artificial intelligence* (2020), 603–649.

[5] DATAFLAIR. Detecting Fake News with Python and Machine Learning. https://data-flair.training/blogs/advanced-python-project-detecting-fake-news/, 2021. Last Accessed June 10, 2023.

[6] GÉRON, A. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

[7] REIS, J. C., CORREIA, A., MURAI, F., VELOSO, A., AND BENEVENUTO, F. Supervised learning for fake news detection. *IEEE Intelligent Systems 34*, 2 (2019), 76–81.

[8] SHU, K., SLIVA, A., WANG, S., TANG, J., AND LIU, H. Fake news detection on social media: A data mining perspective. *ACM SIGKDD explorations newsletter 19*, 1 (2017), 22–36.

[9] TOWARDSAI. Fake News Detection using BERT Model Python. https://towardsai.net/p/l/fake-news-detection-using-bert-model-python, 2022. Last Accessed June 10, 2023.

A Main code sections for the solution

A.1 Code for using intel© AI accelerator in DevCloud

Global patching used to enable patching for `scikit-learn` installation for all further runs:

```
from sklearnex import patch_sklearn
patch_sklearn()
```

A.2 Loading data from the source

Data for this project is taken from the source: https://onlineacademiccommunity.uvic.ca/isot/wp-content/uploads/sites/7295/2023/03/News-_dataset.zip. The python code section for this stage is shown bellow:

```
data_directory = 'data/'
if not os.path.exists(data_directory):
    !mkdir data/
    !wget https://onlineacademiccommunity.uvic.ca/isot/wp-content/uploads/sites/
                                            7295/2023/03/News-_dataset.zip --
                                            directory-prefix=data/
```

```
    !unzip data/News-_dataset.zip -d data/
    fake_data = pd.read_csv('data/Fake.csv')
#fake_data.head()
true_data = pd.read_csv('data/True.csv')
#true_data.head()
```

A.3 Python function for visualization

The general function for basic EDA of the data is created in python. The code for this task is given bellow:

```
def visualize(dataFile,feature):
       plt.figure(figsize = (6,4))
       sns.set(style = "whitegrid",font_scale = 1.0)
       chart = sns.countplot(x = feature, data = data)
       chart.set_xticklabels(chart.get_xticklabels(),rotation=90)
       plt.show()
```

A.4 Data cleaning & pre-processing

Python code for checking for missing values, creation of meaning ful news content from the data and other preprocessing task are shown bellow:

```
def Check_forNAN(data):
    print("Wait...Checking for NANs in the Dataset is progressing...")
    print("Total NANs:",data.isnull().sum())
    print("Checking is completed successfully...\n")
    print(10*"--","\n")
    print("Summary of the dataframe:.....\n")
    data.info()

        print("check finished.")
```

```
    # tokenization
    def tokenize(column):
        tokens = nltk.word_tokenize(column)
        return [w for w in tokens if w.isalpha()]
    # stopwords removal
    def remove_stopwords(tokenized_column):
        stops = set(stopwords.words("english"))
        return [word for word in tokenized_column if not word in stops]
    # stemming
    def apply_stemming(tokenized_column):
        stemmer = PorterStemmer()
        return [stemmer.stem(word) for word in tokenized_column]
    # creating words bag
    def rejoin_words(tokenized_column):
        return ( " ".join(tokenized_column))
```

A function -`PreProcess()` is defined to do all these steps sequentially:

```
    ## Creating Data cleaning and pre-processing function
def PreProcess(data):
    data['tokenized'] = data.apply(lambda x: tokenize(x['text']), axis=1)
```

```
    data['stopwords_removed'] = data.apply(lambda x: remove_stopwords(x['tokenized
                                                    ']), axis=1)
    data['stemmed'] = data.apply(lambda x: apply_stemming(x['stopwords_removed']),
                                                      axis=1)
    data['rejoined'] = data.apply(lambda x: rejoin_words(x['stemmed']), axis=1)
```

A.5 Feature Extraction

Here we use the TF-IDF technique for feature extraction

```
#splitting the data frame into data and label
    data.label = data.label.astype(str)
#data.label = data.label.str.strip()
dict = { 'REAL' : 1 , 'FAKE' : '0'}
#data['label'] = data['label'].map(dict)
data['label'].head()
X = data['rejoined']
y = data['label']
#vectorisation
count_vectorizer = CountVectorizer()
count_vectorizer.fit_transform(X)
freq_term_matrix = count_vectorizer.transform(X)
tfidf = TfidfTransformer(norm = "l2")
tfidf.fit(freq_term_matrix)
tf_idf_matrix = tfidf.fit_transform(freq_term_matrix)
print(tf_idf_matrix)
```

Here we use chi-square feature extraction tecnique

```
chi2_features = SelectKBest(chi2, k = 1000)
X_small = chi2_features.fit_transform(tf_idf_matrix, y)
```

A.6 Visulisation of bag of words and word count

```
fake_data = data[data["label"] == "1"]
all_words = ' '.join([text for text in fake_data.rejoined])
wordcloud = WordCloud(width= 800, height= 500,
                          max_font_size = 110,
                          collocations = False).generate(all_words)
plt.figure(figsize=(10,7))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
def counter(text, column_text, quantity):
    all_words = ' '.join([text for text in text[column_text]])
    token_phrase = token_space.tokenize(all_words)
    frequency = nltk.FreqDist(token_phrase)
    df_frequency = pd.DataFrame({"Word": list(frequency.keys()),
                                 "Frequency": list(frequency.values())})
    df_frequency = df_frequency.nlargest(columns = "Frequency", n = quantity)
    plt.figure(figsize=(12,8))
    ax = sns.barplot(data = df_frequency, x = "Word", y = "Frequency", color = '
                                                  blue')
    ax.set(ylabel = "Count")
    plt.xticks(rotation='vertical')
```

```
    plt.show()
    counter(data[data['label'] == '1'], 'rejoined', 20)
```

A.7 Dataset preperation

Splitting the dataset into training and testing sets.It helps to avoid overfitting and to accurately evaluate your model.

```
#splitting data for training and test
x_train, x_test, y_train, y_test = train_test_split(tf_idf_matrix,y, random_state=
                                        21)
```

A.8 Model Training& Evaluation

Model training is done on the basis of machine learning algorithms such as Naive Bayes,Logistic Regression or Decision tree using the training set and extracted features.The model is then evaluated using metrics such as accuracy, precision, recall, and F1-score.

```
#logistic regrssion
accuracy_values=[]
logitmodel = LogisticRegression()
logitmodel.fit(x_train, y_train)
y_pred=logitmodel.predict(x_test)
Accuracy = logitmodel.score(x_test, y_test)
accuracy_values.append((Accuracy*100))
print(Accuracy*100)

#Naive bayes classification
NB = MultinomialNB()
NB.fit(x_train, y_train)
y_pred=NB.predict(x_test)
Accuracy = NB.score(x_test, y_test)
accuracy_values.append((Accuracy*100))
print(Accuracy*100)

#Decision tree
clf = DecisionTreeClassifier()
clf.fit(x_train, y_train)
y_pred=clf.predict(x_test)
Accuracy = clf.score(x_test, y_test)
accuracy_values.append((Accuracy*100))
print(Accuracy*100)

#Passive aggresive clasiifier
pac=PassiveAggressiveClassifier(max_iter=50)
pac.fit(x_train,y_train)
y_pred=pac.predict(x_test)
score=accuracy_score(y_test,y_pred)
accuracy_values.append((score*100))
print(f'Accuracy: {round(score*100,2)}%')

#Random Forest regression
rf = RandomForestRegressor(**params).fit(x_train, y_train)
train_patched = timer() - start
y_pred = rf.predict(x_test)
```

```
mse_opt = metrics.mean_squared_error(y_test, y_pred)
rf = RandomForestRegressor(**params).fit(x_train, y_train)
```

B Main code for BERT model

B.1 Downloading the dataset and extracting it to the appropriate data directory

```
data_directory = 'data/'
if not os.path.exists(data_directory):
!mkdir data/
!wget https://onlineacademiccommunity.uvic.ca/isot/wp-content/uploads/sites/7295/
                                        2023/03/News-_dataset.zip --directory-
                                        prefix=data/
!unzip data/News-_dataset.zip -d data/
df_fake = pd.read_csv("./data/Fake.csv")
df_true = pd.read_csv("./data/True.csv")
#creating the label
df_fake["Label"] = "Fake"
df_true["Label"] = "True"
df = pd.concat([df_fake,df_true])
df = df.sample(frac=1).reset_index(drop=True)
df = pd.concat([df_fake,df_true])
df = df.sample(frac=1).reset_index(drop=True)
```

B.2 Getting the pretrained model

```
#function get_model() is defined
def get_model():
   dropout_rate = 0.2
   input_ids = Input(shape = (Length,), dtype = tf.int32, name = 'input_ids')
   input_mask = Input(shape = (Length,), dtype = tf.int32, name = 'input_mask')
   embeddings = bert([input_ids, input_mask])[1] #pooler output
   print(embeddings)
   out = Dropout(0.2)(embeddings)
   out = Dense(20,activation = 'relu')(out)
   out = Dropout(0.2)(out)
   y = Dense(1,activation = 'sigmoid')(out)
   model = Model(inputs=[input_ids, input_mask], outputs=y)
   model.layers[2].trainable = True
   optimizer = Adam(learning_rate=1e-05, epsilon=1e-08, clipnorm=1.0)
   model.compile(optimizer = optimizer, loss = 'binary_crossentropy', metrics = '
                                          accuracy')
   return model
```

```
# calling the model

bert = TFBertModel.from_pretrained('bert-base-uncased')
model = get_model()
tf.keras.utils.plot_model(model)
```

B.3 Training and prediction

```
#training model using fit method
history = model.fit(x = {'input_ids':X_train_tokens['input_ids'],'input_mask':
                                        X_train_tokens['attention_mask']}, y =
                                        y_train, epochs=3, validation_split = 0.2
                                        , batch_size = 64, callbacks=[
                                        EarlyStopping( monitor='val_accuracy' ,
                                        mode='max', patience=3,verbose=False,
                                        restore_best_weights=True)])
#predict method of the trained model generates predictions on the test data
yhat = np.where(model.predict({ 'input_ids' : X_test_seq['input_ids'] , '
                                        input_mask' : X_test_seq['attention_mask'
                                        ]}) >=0.5,1,0)
print(classification_report(y_test,yhat))
```