PROJECT REPORT

FAKE NEWS DETECTION USING MACHINE LEARNING

Abstract

Fake news detection is a crucial task in today's information-rich digital landscape. This project focuses on developing a machine learning model for accurately identifying fake news articles. By leveraging a dataset of genuine and fake news articles, the model aims to classify news articles and differentiate between legitimate and fabricated information. Through text preprocessing, TF-IDF vectorization, and the application of the PassiveAggressiveClassifier algorithm, the model achieves accurate classification of news articles. This project emphasizes the importance of text preprocessing and hardware optimization for efficient and effective fake news detection.

Introduction

The purpose of this project is to develop a machine learning model for the detection of fake news. With the rapid growth of information sharing on the internet, the spread of fake news has become a significant concern. Fake news can have severe consequences, including influencing public opinion, damaging reputations, and undermining trust in media and institutions. Therefore, the ability to automatically identify and classify fake news articles is of great importance.

Motivation behind the problem

The ease of information sharing and the rapid dissemination of news through social media platforms have contributed to the rapid spread of misinformation. The consequences of fake news are far-reaching, including the potential to sway public opinion, create social unrest, and undermine trust in media and institutions.

The motivation behind this project lies in the urgent need to combat the spread of fake news and promote accurate information dissemination. By developing a machine learning model for fake news detection, we aim to provide a proactive solution to identify and classify misleading or fabricated news articles. Such a model can assist individuals, media organizations, and fact-checking agencies in verifying the authenticity of news

and preventing the propagation of false information. By leveraging machine learning techniques, we aspire to foster a more informed and discerning society, where the spread of misinformation is mitigated, and accurate information prevails.

Prior Research

Prior research on fake news detection has explored manual fact-checking, rule-based systems, and machine learning approaches. Manual fact-checking is accurate but time-consuming. Rule-based systems struggle to adapt to evolving fake news tactics. Machine learning techniques have gained prominence, employing algorithms to learn patterns from labeled datasets. Feature engineering, including TF-IDF and metadata analysis, enhances detection. Various algorithms, ensembles, and hybrid methods have been utilized. Challenges include handling sophisticated fake news, adversarial attacks, and evolving strategies. Ongoing research focuses on resilience, scalability, and interpretability, utilizing deep learning and network analysis. The field continues to advance toward reliable and scalable solutions to combat the spread of fake news and promote information integrity.

Our Approach

1. **Dataset Preparation**

The first step in our project is to prepare the dataset for analysis. We import the necessary libraries, including pandas and matplotlib, and read the CSV files into pandas DataFrames. We assign labels to the data, where 1 represents genuine news and 0 represents fake news. Subsequently, we combine the two datasets using the **pd.concat()** function.

2. Exploratory Data Analysis (EDA)

To gain insights into the dataset, we conduct exploratory data analysis. We begin by visualizing the distribution of news articles based on their subjects using a count plot. This provides an overview of the dataset and helps identify any imbalances or patterns in the data.

```
import matplotlib.pyplot as plt
plt.figure(figsize=(6,4))

import seaborn as sns
sns.set(style = "whitegrid",font_scale = 1.0)
chart = sns.countplot(x = "subject", data = data)
chart.set_xticklabels(chart.get_xticklabels(),rotation=90)
```

3. Feature Extraction

Next, we preprocess the text data to ensure consistency and remove irrelevant information. We convert the text to lowercase, remove Twitter handles, URLs, hashtags, punctuation, and newlines using regular expressions. This step prepares the text for further analysis and feature extraction.

```
import re

def preprocess_text(text):
    text = text.lower()
    text = re.sub(r"@\S+","",text)
    text = re.sub("http[s]?\://\S+","",text)
    text = re.sub(r"#\S+","",text)
    text = re.sub(r'[^\w\s]','',text)
    text = re.sub(r"\n","",text)
    return text

data.text = data.text.apply(preprocess_text)
data.head()
```

4. Model Training

In this phase, we split the preprocessed text data into training and testing sets using the **train_test_split()** function from the sklearn.model_selection module. This allows us to evaluate the performance of our model on unseen data. We have used **Intel® extension for Sci-kit learn** for train_test_split().

To convert the textual data into numerical features, we employ the TF-IDF vectorization technique. We use the **TfidfVectorizer** class from the sklearn.feature_extraction.text module to transform the text data into TF-IDF vectors. This vectorization process assigns weights to each word based on its importance in the corpus, enabling the model to learn meaningful patterns.

```
from sklearnex.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(data['text'],data.label,test_size=0.2,random_state=7)
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vectorizer=TfidfVectorizer(stop_words='english', max_df=0.7)

tfidf_train=tfidf_vectorizer.fit_transform(x_train)
tfidf_test=tfidf_vectorizer.transform(x_test)
```

We then train a machine learning model called **PassiveAggressiveClassifier**. This classifier is known for its effectiveness in handling text classification tasks. To enhance the performance of the model, we patch the scikit-learn library with the **Intel® extension**, which optimizes computations and speeds up the training process. We measure the training time to assess the efficiency of the **patched scikit-learn library**.

```
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearnex import patch_sklearn
from timeit import default_timer as timer

patch_sklearn()

params = {
    "max_iter": 50
}

start = timer()
model = PassiveAggressiveClassifier(**params).fit(tfidf_train, y_train)
train_patched = timer() - start
```

5. Model Evaluation

After training the model, we evaluate its performance using various metrics. Firstly, we make predictions on the test set using the trained model and calculate the accuracy score. The accuracy score indicates the percentage of correctly classified instances and serves as a general performance metric.

```
y_pred = model.predict(tfidf_test)

from sklearn.metrics import accuracy_score

score = accuracy_score(y_test, y_pred)
print(f"Patched Scikit-learn Accuracy: {round(score*100, 2)}%")
```

Additionally, we compute a confusion matrix to provide detailed insights into the model's predictions. The confusion matrix shows the number of true positives, true negatives, false positives, and false negatives. It helps us understand the classification results and identify any specific areas where the model may be struggling.

```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_test, y_pred, labels=[0, 1])
```

6. Unpatching scikit-learn

In this section, we unpatch the scikit-learn library and retrain the model using the original implementation. By comparing the performance of the patched and original scikit-learn libraries, we can evaluate the impact of the **Intel® extension** on training time and accuracy.

```
from sklearnex import unpatch_sklearn

unpatch_sklearn()

params = {
    "max_iter": 50
}

start = timer()
model = PassiveAggressiveClassifier(**params).fit(tfidf_train, y_train)
train_patched = timer() - start

f"Original Scikit-learn time: {train_patched:.2f} s"
```

By combining these steps, we develop a machine learning model that effectively detects fake news articles. This approach leverages the power of text preprocessing, TF-IDF vectorization, and the PassiveAggressiveClassifier algorithm to accurately classify news articles. The comparison between the patched and original scikit-learn libraries provides insights into the benefits of hardware optimization for training time and accuracy.

Results and Discussion

The results of our experiments indicate the effectiveness of the machine learning model in detecting fake news. The accuracy scores achieved demonstrate the model's ability to

distinguish between genuine and fake news articles. The confusion matrix provides a detailed breakdown of the model's predictions, revealing areas of success and potential weaknesses.

The comparison between the patched and original scikit-learn libraries allows us to assess the efficiency gains obtained from utilizing the Intel® extension. This analysis provides insights into the benefits of leveraging hardware acceleration for machine learning tasks.

```
from sklearn.metrics import confusion_matrix
conf = confusion_matrix(y_test,y_pred,labels=[0,1])
conf
```

```
array([[4716,   20],
       [  15, 4229]])
```

```
f"We have {conf[0][0]} True Positives, {conf[1][1]} True Negatives, {conf[1][0]} False Positives, {conf[0][1]} False Negati
```

```
'We have 4716 True Positives, 4229 True Negatives, 15 False Positives, 20 False Negatives'
```

```
'Intel® extension for Scikit-learn time: 0.27 s'
```

```
y_pred = model.predict(tfidf_test)

from sklearn.metrics import accuracy_score
score = accuracy_score(y_test,y_pred)
print(f"Patched Scikit-learn Accuracy : {round(score*100,2)}%")
```

```
Patched Scikit-learn Accuracy : 99.57000000000001%
```

References

1.https://onlineacademiccommunity.uvic.ca/isot/2022/11/27/fake-news-detection-datasets/

2. https://paperswithcode.com/task/fake-news-detection

3.https://www.sciencedirect.com/science/article/pii/S1877050918318210

4. https://arxiv.org/abs/1901.08232

5. https://arxiv.org/abs/1801.02421

Link to Solution:
https://github.com/Itssundarr/Triforce_KarunyaInstitute_Fake-news-detection

Conclusion

In conclusion, this project successfully addresses the challenge of fake news detection using machine learning techniques. By training a model on a dataset of genuine and fake news articles, we demonstrate the ability to classify news articles with a high degree of accuracy.

The project highlights the importance of preprocessing text data to standardize the input and improve.