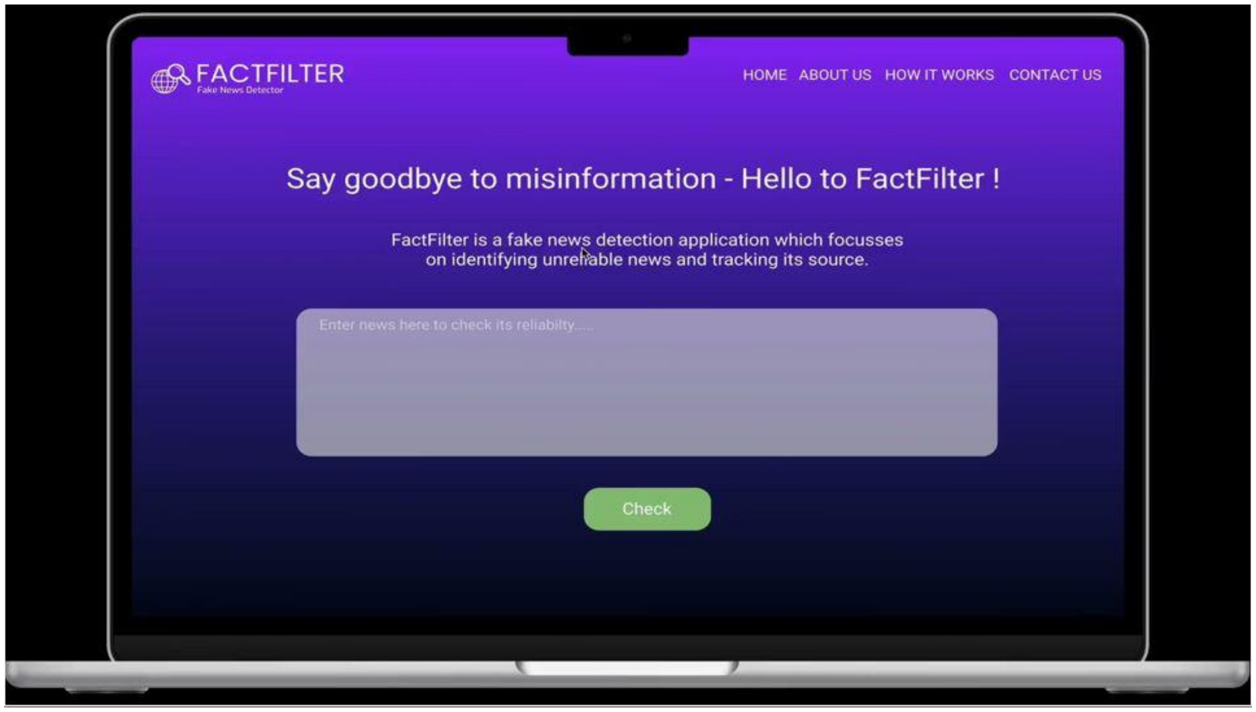
FactFilter

Fake News Prediction Application



Say goodbye to misinformation - Hello to FactFilter !

FactFilter is a fake news detection application which focusses on identifying unreliable news and tracking its source.

Team Members:

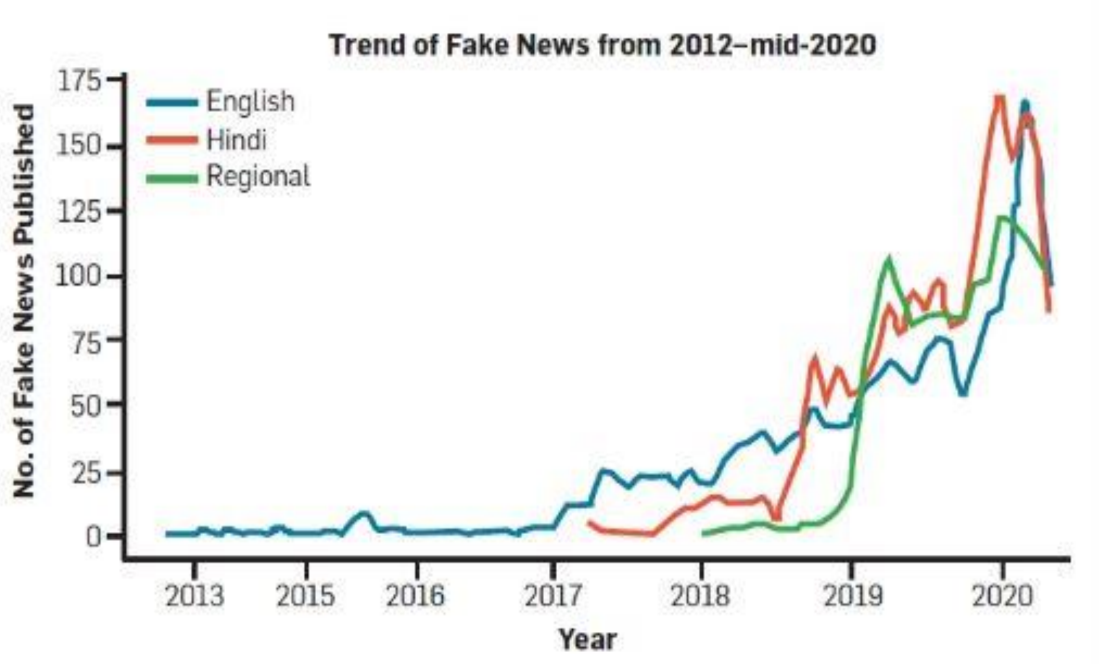
Mridul Jain
Hardik Sharma
Vansh sonawane
Shyamashree Ghorai

INTRODUCTION

In an era characterized by the rapid dissemination of information through digital platforms, the rise of fake news has posed a profound challenge to the accuracy and reliability of news sources. In response to this critical issue, innovative solutions have emerged to empower individuals with the ability to discern between credible information and fabricated content. One such solution is "FactFilter," a pioneering fake news detection app. FactFilter stands as a beacon of information integrity, offering users a tool to navigate the intricate landscape of news articles by employing cutting-edge technologies like Natural Language Processing (NLP) and source credibility analysis. This app aims to empower users with the means to make informed decisions, fostering media literacy and critical thinking in an age where misinformation can sway opinions and skew public discourse. Through its multifaceted approach, FactFilter seeks to counteract the proliferation of fake news, promoting a more informed and discerning society.

PROBLEM STATEMENT

India recorded a massive 214 percent rise in cases related to misinformation and rumours in 2020, a three-fold rise over 2019, according to the latest National Crime Records Bureau (NCRB) data. A total of 1,527 incidents of fake news were reported when the coronavirus pandemic hit the country, compared to 486 cases recorded in 2019. In 2018, the number was way less, with 280 cases.



APPROACH TO THE APPLICATION

The implementation of fake news detection comprises two phases:

1. News collection and training
2. Machine learning prediction

- In the first phase, web crawlers in parallel collect data from www and social media and preprocessed them to train machine learning as a fake news detection model. In the data collection and training phase, the IR module crawls the web to retrieve the news data from news websites and use them as domain corpus used in the NLP module.

- The NLP module receives the news content and performs text segmentation, cleansing, and feature extraction. NLP will process the retrieved news list and return featured data
- Splitting the pre- processed data into training and test data
- The logistic regression model is trained, the accuracy is checked for the classification. After that check the accuracy with test data.
- Now real time data or news is fed into the trained model to predict whether the news is real or fake.

BUSINESS MODEL

Freemium Subscription Model:

Offer a basic version of FactFilter for free, allowing users to access essential fake news detection features. To access more advanced features like real-time analysis, comprehensive fact-checking, and integration with social media platforms, users can opt for a premium subscription. This tiered approach attracts a wide user base while generating revenue from those seeking enhanced functionality.

Licensing to Media Outlets and Platforms:

License the FactFilter technology to news organizations, media outlets, and content platforms. These entities can integrate FactFilter's API or SDK into their platforms to ensure the accuracy and reliability of the news articles they publish. The licensing model provides a steady stream of income through agreements with various media partners.

Tech Stack , Dependencies & Future Add Ons

TECH STACK:

- Python
- Python Libraries (Pandas, Seaborn, Matplotlib,Sklearn)
- ReactJs
- Django
- Deep Learning
- Selenium

DEPENDENCIES:

- Model Training
- Model Selection

- Data Quality
- Machine Learning Concepts

FUTURE SCOPE:

- Multi Language Support
- Image Recognition

APPLICATIONS

Here the user's goal is to check whether the news is real or fake-

- Media and Journalism: Detecting and filtering fake news articles from real ones in order to ensure accurate and reliable information is disseminated to the public.
- Social Media Platforms: Detecting and flagging fake news articles shared on social media platforms to prevent the spread of false information.
- Government and Politics: Detecting and combating fake news in government and political organizations to ensure accurate information is available to citizens.
- Education: Helping students learn how to identify fake news and improve media literacy skills.

CODE IMPLEMENTATION

```
#importing the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn import feature_extraction, linear_model, model_selection,
preprocessing
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
import re
import string

fake=pd.read_csv("fake_Data.csv")
fake

true=pd.read_csv("True_Data.csv")
true.head()

fake.shape

true.shape



"""## Data Cleaning And Preperation"""

# Add flag to track fake and real
fake['target'] = 'fake'
true['target'] = 'true'

# Concatenate dataframes
data = pd.concat([fake, true]).reset_index(drop = True)
data.shape

# Shuffle the data
from sklearn.utils import shuffle
data = shuffle(data)
data = data.reset_index(drop=True)

# Check the data
```

```python
data.head()

# Removing the date (we won't use it for the analysis)
data.drop(["date"],axis=1,inplace=True)
data.head()

# Removing the title (we will only use the text)
data.drop(["title"],axis=1,inplace=True)
data.head()

# Converting to lowercase

data['text'] = data['text'].apply(lambda x: x.lower())
data.head()

data.info()

data.isnull().sum()

# Removing stopwords
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop = stopwords.words('english')

data['text'] = data['text'].apply(lambda x: ' '.join([word for word in x.split()
if word not in (stop)]))

data.head()

"""## Basic data Exploration"""

# How many articles per subject?
print(data.groupby(['subject'])['text'].count())
data.groupby(['subject'])['text'].count().plot(kind="bar")
plt.show()

# How many fake and real articles?
print(data.groupby(['target'])['text'].count())
data.groupby(['target'])['text'].count().plot(kind="bar")
plt.show()
```

```
# Most frequent words counter (Code adapted from
https://www.kaggle.com/rodolfoluna/fake-news-detector)
from nltk import tokenize

token_space = tokenize.WhitespaceTokenizer()

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()

# Most frequent words in fake news
counter(data[data["target"] == "fake"], "text", 20)

# Most frequent words in real news
counter(data[data["target"] == "true"], "text", 20)

"""## Modeling

"""

# Function to plot the confusion matrix
from sklearn import metrics
import itertools

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    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)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

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

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

"""## Preparing the data

"""

# Split the data
X_train,X_test,y_train,y_test = train_test_split(data['text'], data.target,
test_size=0.2, random_state=42)

"""## Logistic regression

"""

# Vectorizing and applying TF-IDF
from sklearn.linear_model import LogisticRegression

pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', LogisticRegression())])

# Fitting the model
model = pipe.fit(X_train, y_train)

# Accuracy
prediction = model.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))

cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```

```
"""## Decision Tree Classifier

"""

from sklearn.tree import DecisionTreeClassifier

# Vectorizing and applying TF-IDF
pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', DecisionTreeClassifier(criterion= 'entropy',
                                           max_depth = 20,
                                           splitter='best',
                                           random_state=42))])
# Fitting the model
model = pipe.fit(X_train, y_train)

# Accuracy
prediction = model.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))

cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])

"""## Random Forest Classifier

"""

from sklearn.ensemble import RandomForestClassifier

pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', RandomForestClassifier(n_estimators=50,
criterion="entropy"))])

model = pipe.fit(X_train, y_train)
prediction = model.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))

cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])

"""## TfidfVectorizer

Convert a collection of raw documents to a matrix of TF-IDF features
```

```
TF-IDF where TF means term frequency, and IDF means Inverse Document frequency.
"""

from sklearn.feature_extraction.text import TfidfVectorizer
text = ['Hello Hardik Sharma this side , I love machine learning','Welcome to the
Machine learning hub' ]

vect = TfidfVectorizer()

vect.fit(text)

## TF will count the frequency of word in each document. and IDF
print(vect.idf_)

print(vect.vocabulary_)

""" A word  which is present in all the data, it will have low IDF value. With
this unique words will be highlighted using the Max IDF values."""

example = text[0]
example

example = vect.transform([example])
print(example.toarray())

"""Here, 0 is present in the which indexed word, which is not available in given
sentence.

PassiveAggressiveClassifier
Passive: if correct classification, keep the model; Aggressive: if incorrect
classification, update to adjust to this misclassified example.
"""

tfvect = TfidfVectorizer(stop_words='english',max_df=0.7)
tfid_x_train = tfvect.fit_transform(X_train)
tfid_x_test = tfvect.transform(X_test)

classifier = PassiveAggressiveClassifier(max_iter=50)
classifier.fit(tfid_x_train,y_train)

y_pred = classifier.predict(tfid_x_test)
score = accuracy_score(y_test,y_pred)
print(f'Accuracy: {round(score*100,2)}%')
```

```
def fake_news_det(news):
    input_data = [news]
    vectorized_input_data = tfvect.transform(input_data)
    prediction = classifier.predict(vectorized_input_data)
    print(prediction)

fake_news_det('U.S. Secretary of State John F. Kerry said Monday that he will
stop in Paris later this week, amid criticism')

fake_news_det('On Friday, it was revealed that former Milwauk')

import pickle
pickle.dump(classifier,open('model.pkl', 'wb'))

# load the model from disk
loaded_model = pickle.load(open('model.pkl', 'rb'))

def fake_news_det1(news):
    input_data = [news]
    vectorized_input_data = tfvect.transform(input_data)
    prediction = loaded_model.predict(vectorized_input_data)
    print(prediction)

fake_news_det1('''Go to Article
President Barack Obama has been campaigning hard for the woman who is supposedly
going to extend his legacy four more years.''')

fake_news_det('''U.S. Secretary of State John F. Kerry said Monday that he will
stop in Paris later this week, amid criticis''')
```