AI_Phase5

October 28, 2023

1 Build an NLP model to differentiate real news from fake news

1.1 Problem Definition:

In today's information age, the proliferation of fake news and misinformation poses a significant challenge to society. The widespread dissemination of false or misleading information can have detrimental effects on public perception, decision-making, and the democratic process. As a responsible response to this issue, our project aims to develop an NLP (Natural Language Processing) model that effectively differentiates real news from fake news.

1.2 Design Thinking Process:

Empowering Readers: Our primary goal is to empower readers with a tool that equips them to navigate through the complex landscape of news sources and distinguish reliable information from misinformation. This aligns with our commitment to fostering a more informed and discerning society.

Leveraging NLP and Machine Learning: To achieve this, we employ state-of-the-art NLP analysis and classification algorithms. These algorithms will harness the power of language understanding to determine the credibility of news articles.

Python with NLP Libraries: Our technology stack is built on Python, utilizing NLP libraries and classification models. Python's versatility and the rich ecosystem of NLP libraries will enable us to implement an efficient and scalable solution.

Architecting a Truth-Seeking AI Detective: At the heart of our project is the creation of a "truth-seeking AI detective." This sophisticated AI system will act as a lie detector for news, promoting accurate information and debunking falsehoods.

Elevating News Consumption: Our work aims to elevate news consumption by providing users with data-backed insights into the credibility of news sources. This will, in turn, encourage more informed and critical decision-making.

Real-World Analogy: The Lie Detector for News: An apt analogy for our project is a "lie detector for news." Just as a lie detector assists in uncovering truth from deception, our system will assist readers in distinguishing credible news from fake news, ultimately promoting truth and accuracy in the information landscape. By following this design thinking approach, we intend to develop a robust and user-friendly NLP-based news credibility detection system that will serve as a valuable resource for news consumers, journalists, and researchers alike.

By following this design thinking approach, we intend to develop a robust and user-friendly NLP-based news credibility detection system that will serve as a valuable resource for news consumers,

journalists, and researchers alike.

1.3 Phases of Development:

1. Model Training in Google Colab
   - Preprocess and clean the news dataset.
   - Implement and test various machine learning algorithms (e.g., Random Forest, Naive Bayes).
   - Analyze the performance metrics of each algorithm (e.g., accuracy, precision, recall).
   - Choose the algorithm that demonstrates the highest accuracy for news credibility assessment.
   - Save the trained model in Google Colab.
   - Export the selected model from Google Colab to your local machine.
2. Web Application Development
   - Set up a Django project for the web application.
   - Develop the RESTful API to handle incoming news data and predictions.
   - Begin developing the user interface using React.js.
   - Establish communication with the Django REST API to fetch predictions.
   - Integrate the web application with The Guardian's news API to retrieve real-time news articles.
   - Implement the logic for running background threads every 10 seconds to fetch new news.
   - Develop functionality to check whether the news is already in the database.
   - Set up a database system to store news articles and their predicted results.
   - Establish database connectivity within the Django project.
3. Testing, Deployment, and Monitoring
   - Conduct extensive testing, including unit testing and integration testing, to ensure the system functions correctly.
   - Verify the accuracy of news predictions.
   - Address and resolve any issues or bugs that arise during testing.
   - Plan for regular maintenance, updates, and scalability considerations.
4. Project Evaluation
   - Continuously monitor the accuracy of news predictions.
   - Stay updated with developments in machine learning and NLP to enhance the model's accuracy.

1.4 Description of dataset used:

W've used the dataset provided by you from the source https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset

This dataset contains the attributes such as *title, text, subject, date*. This source has two csv files. One with collection of fake news data and another one with collection of real news data

1.5 Data Preprocessing Steps and Feature Extraction Techniques.:

In this code, we are working with a dataset that consists of real and fake news articles, and our goal is to preprocess the data for text classification. Here's a step-by-step explanation of the code:

1. We start by loading two CSV files, 'True.csv' and 'Fake.csv', containing real and fake news articles respectively into two pandas DataFrames named 'real_data' and 'fake_data'.

```
real_data = pd.read_csv('dataset/True.csv')
fake_data = pd.read_csv('dataset/Fake.csv')
```

2. We add a new column 'target' to both DataFrames. This column is used to label real articles with '1' and fake articles with '0'.

```
real_data['target'] = 1
fake_data['target'] = 0
```

3. We concatenate the two DataFrames into a single DataFrame named 'data'. The 'ignore_index' parameter ensures that the index is reset for the merged DataFrame.

```
data = pd.concat([real_data, fake_data], ignore_index=True, sort=False)
```

4. We combine the 'subject', 'title', and 'text' columns into a single 'text' column, and remove the 'title', 'subject', and 'date' columns.

```
data['text'] = data['subject'] + " " + data['title'] + " " + data['text']
del data['title']
del data['subject']
del data['date']
```

5. We perform text preprocessing on the 'text' column. This includes removing HTML tags, punctuation marks, non-alphabet characters, converting text to lowercase, and lemmatization using NLTK. We also download and use NLTK's English stopwords.

```
from bs4 import BeautifulSoup
import re
import nltk

# Code to remove HTML tags and perform other text preprocessing steps
# ...

#Apply function on text column
data['text'] = data['text'].apply(cleaning)
```

6. We set some parameters for text tokenization and padding. `max_features` specifies the maximum number of words in the vocabulary, and `maxlen` defines the maximum length of the sequences. We use the Keras tokenizer to tokenize and pad the text data.

```
max_features = 10000
maxlen = 300
tokenizer = text.Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(X_train)
tokenized_train = tokenizer.texts_to_sequences(X_train)
X_train = sequence.pad_sequences(tokenized_train, maxlen=maxlen)
tokenized_test = tokenizer.texts_to_sequences(X_test)
X_test = sequence.pad_sequences(tokenized_test, maxlen=maxlen)
```

This code prepares the text data for a classification task, with the 'text' column being the input features, and 'target' being the labels (1 for real and 0 for fake). The text is preprocessed and tokenized to be used with a neural network model for classification.

1.6 Choice of Machine Learning Algorithm:

- We've chosen a sequential neural network model for text classification.
- The model consists of an embedding layer, followed by two LSTM (Long Short-Term Memory) layers, and then two fully connected (dense) layers.
- We've used LSTM layers to capture sequential information in the text data effectively.

```
batch_size = 256
epochs = 10
embed_size = 100
model = Sequential()
model.add(Embedding(max_features, output_dim=embed_size, input_length=maxlen, trainable=False))
model.add(LSTM(units=128, return_sequences=True, recurrent_dropout=0.25, dropout=0.25))
model.add(LSTM(units=64, recurrent_dropout=0.1, dropout=0.1))
model.add(Dense(units=32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

1.7 Model Training

- We've compiled the model with specific settings.
- We use the Adam optimizer with a learning rate of 0.01 and binary cross-entropy loss for binary classification.
- We are interested in tracking accuracy as our evaluation metric.

```
model.compile(optimizer=keras.optimizers.Adam(lr=0.01), loss='binary_crossentropy', metrics=['a
```

- The model is trained on the training data with a batch size of 256 for 10 epochs.
- We use a validation split of 30% to monitor the model's performance on the validation set.
- The 'shuffle' parameter is set to 'True' to shuffle the training data for each epoch.

```
history = model.fit(X_train, y_train, validation_split=0.3, epochs=10, batch_size=batch_size, s
```

1.8 Evaluation Metrics

- We evaluate the model's accuracy on both the training and testing data to assess its performance.

```
print("Accuracy of the model on Training Data is -", model.evaluate(X_train, y_train)[1] * 100
print("Accuracy of the model on Testing Data is -", model.evaluate(X_test, y_test)[1] * 100, "%
```

- We create two plots to visualize the training and validation accuracy as well as the training and validation loss over the epochs. These plots provide insights into how well the model is learning from the data.

The choice of the LSTM-based neural network is suitable for text data due to its ability to capture sequential patterns, and we monitor the accuracy as our evaluation metric to assess the model's performance.

1.9 Innovative approach:

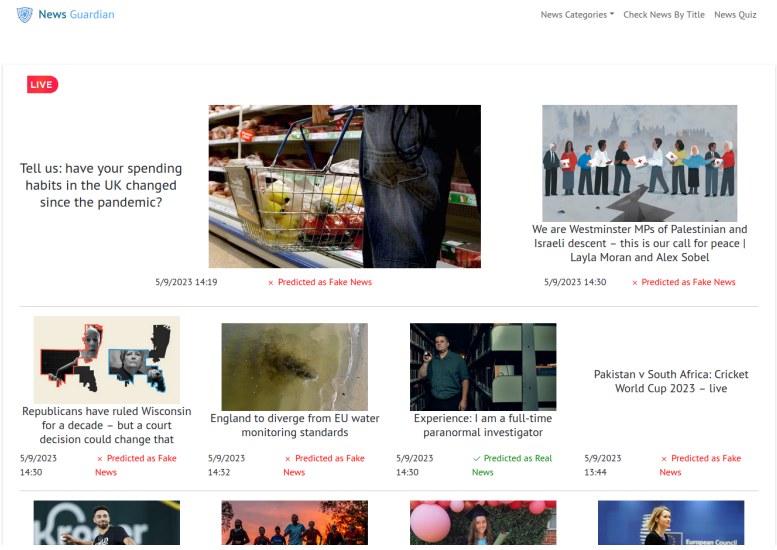
During the development of this project, several innovative techniques and approaches have been implemented to enhance the functionality and user experience. Here are the key innovations:

1. **Real-Time News Classification**:
    - The system integrates with The Guardian's news API to fetch the latest news articles in real time.
    - It uses the AI model to predict and classify whether each news article is real or fake as it is retrieved, providing users with immediate insights into the credibility of the news.
2. **Dynamic Web Application**:
    - A web application has been created using Django and React.js to deliver a seamless user experience.
    - The application functions like a conventional news website, with real-time updates on news articles and their credibility.
3. **Manual News Checking**:
    - Users are given the option to manually check the credibility of news articles.
    - They can copy and paste the news content into an input field and have the AI model analyze it for authenticity.
4. **User Engagement with Quiz**:
    - A quiz-like feature has been introduced, where users are presented with news articles and asked to determine whether they are real or fake.
    - This interactive approach educates users about the challenges of identifying fake news and provides instant feedback on their accuracy.

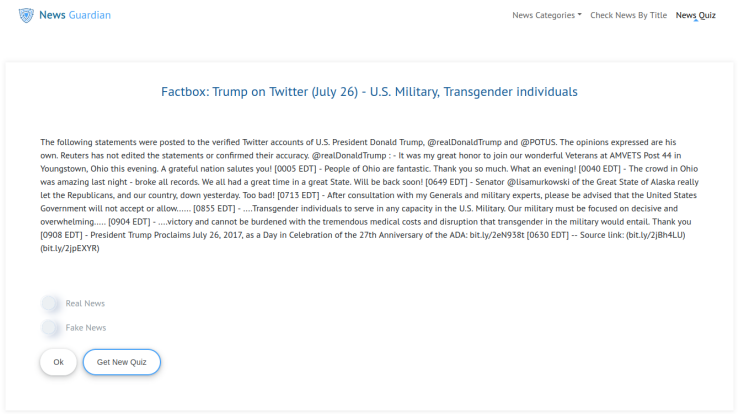
These innovative techniques not only enhance the user's interaction with the system but also contribute to raising awareness about the importance of fake news detection and critical thinking in today's information age. The combination of real-time updates, manual checks, and user engagement through quizzes makes this project a valuable resource for promoting media literacy and responsible news consumption.

Trailer » http://artificialbrains.s3.amazonaws.com/news_guardian.mp4

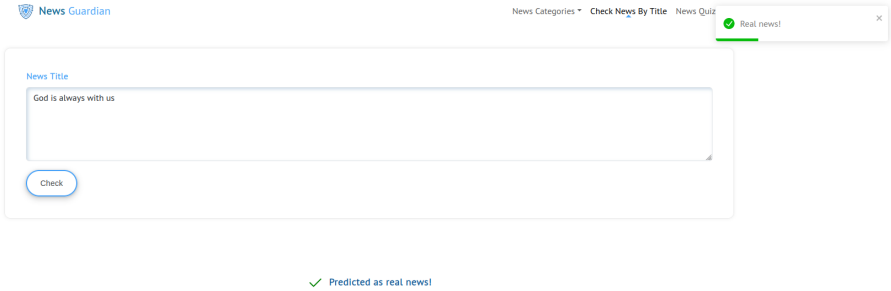
- **Live News Monitoring:** View real-time predictions for news articles.



- **News Quiz:** Test your fake news detection skills by taking our news quiz.



- **Check News by Title:** Enter a news title to see if it's predicted as real or fake.



Full project source code and description : https://github.com/DJDarkCyber/Fake-News-Detector

[ ]: