CSCI 4342 Natural Language Processing

Semester 2 2023/2024

Section: 1

Fake News Detector

Lecturer: Prof. Dr. SURIANI BT. SULAIMAN

Group members:

2020355	2016152	2020600	2024334
Rashad Mohamed Amr	Miad binti Wan Mahri	Binshahbal Danyah Ahmed Salem	Fadwa Ramadan Ali Hassan

Introduction

Developing a Fake News Detector is motivated by the pervasive issue of fake news in the digital age. Fake news spreads rapidly through social media and online platforms, influencing public opinion and decision-making. By distinguishing false information from real news, this project aims to mitigate the harmful effects of misinformation and protect the public from deceitful narratives.

Problem statement

The importance of this problem lies in its broad societal impact. Fake news can lead to misinformation and societal discord and can even influence political outcomes and public health decisions. As such, an effective fake news detection system is crucial for maintaining the integrity of information consumed by the public. It also helps foster a more informed and discerning society where individuals can trust the news they read.

Review of Previous Works

01

Ruchansky et al. (2017)

hybrid deep models for fake news detection provided a foundation for using deep learning in this field

02

Devlin et al. (2019)

with BERT,

03

Brown et al. (2020)

with GPT-3,

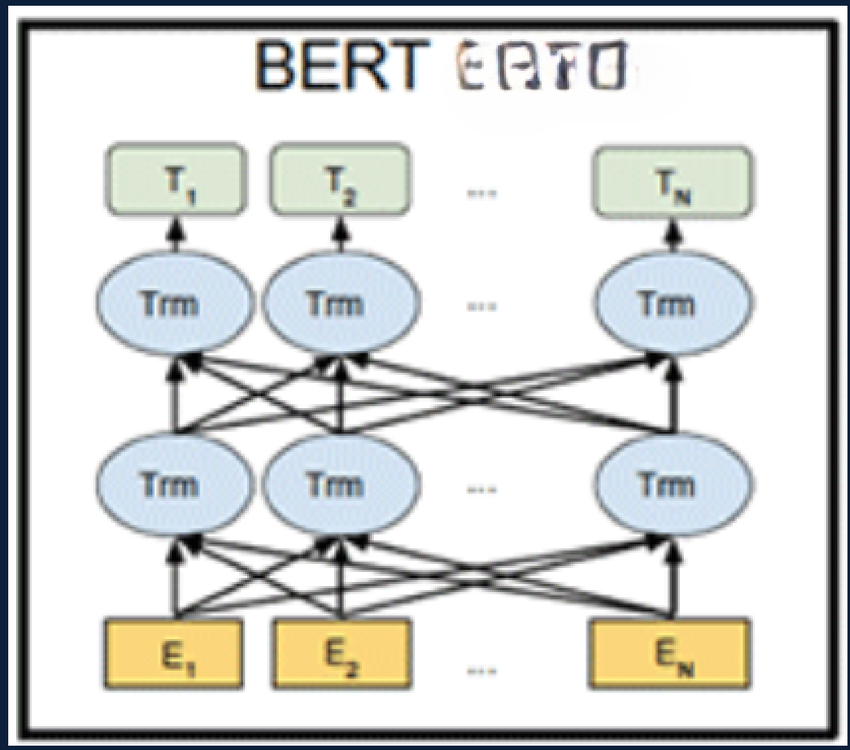
Approach and methodology

Our approach leverages state-of-the-art transformer models like BERT and RoBERTa, which have demonstrated exceptional performance in various NLP tasks. By fine-tuning these pre-trained models on specific fake news datasets, we can adapt them to the nuances of fake news detection. This method allows us to utilise the deep learning capabilities of transformers to achieve a high level of accuracy and reliability in classification tasks. Additionally, combining empirical evaluations with human qualitative assessments ensures that our model's outputs are both statistically valid and contextually accurate.

The overall approach and methodology used:

- Leveraging state-of-the-art transformer models like BERT and RoBERTa for fake news detection

BERT, or Bidirectional Encoder Representations from Transformers. Its design allows the model to consider the context from both the left and the right sides of each word.



- Fine-tuning these pre-trained models on specific fake news datasets to adapt them to the nuances of fake news detection
- Combining empirical evaluations with human qualitative assessments for comprehensive model evaluation

Experimental Setup

1. Dataset Selection
2. Data Preprocessing
3. Data Augmentation
4. Model Developmen
5. Evaluation

1. DATASET SELECTION

Dataset/ Corpus

- LIAR: CONTAINS 12,836 HUMAN-LABELED SHORT STATEMENTS FROM VARIOUS CONTEXTS.
- FAKENEWSNET: INCLUDES NEWS CONTENT, SOCIAL CONTEXT, AND SPATIOTEMPORAL INFORMATION.

```
[ ]  # Load CSV files
     fake_news = pd.read_csv('politifact_fake.csv')
     real_news = pd.read_csv('politifact_real.csv')
     fake_news_gc = pd.read_csv('gossipcop_fake.csv')
     real_news_gc = pd.read_csv('gossipcop_real.csv')
```

2. DATA PREPROCESSING

```
[ ]  # Text cleaning function
     def clean_text(text):
         text = text.lower()
         text = re.sub(r'\b\w{1,2}\b', '', text)  # Remove short words
         text = re.sub(r'[^\w\s]', '', text)  # Remove punctuation
         text = re.sub(r'\s+', ' ', text)  # Remove extra spaces
         stop_words = set(stopwords.words('english'))
         text = ' '.join(word for word in text.split() if word not in stop_words)
         return text
```

2. DATA PREPROCESSING

```
# Apply text cleaning
data['cleaned_title'] = data['title'].apply(clean_text)

X = data['cleaned_title'].values
y = data['label'].values

tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(X)
X = tokenizer.texts_to_sequences(X)
X = pad_sequences(X, maxlen=100)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

3. DATA AUGMENTATION

DATA AUGMENTATION IS PARTICULARLY VALUABLE FOR TRAINING FAKE NEWS DETECTION MODELS BECAUSE:

- **REAL-WORLD NEWS CAN BE LIMITED AND BIASED.**
- **FAKE NEWS CREATORS OFTEN EMPLOY VARIOUS WRITING STYLES AND TECHNIQUES TO DECEIVE.**

By incorporating data augmentation into NLTK-based fake news detection model, we can increase the model's robustness and effectiveness in combating the ever-evolving challenge of fake news.

4. MODEL DEVELOPMEN

MODEL 1:

```
[ ]  # Build model
     model = Sequential()
     model.add(Embedding(input_dim=5000, output_dim=128, input_length=100))
     model.add(Bidirectional(LSTM(128, dropout=0.2, recurrent_dropout=0.2)))
     model.add(Dense(1, activation='sigmoid'))
     model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

     # Train model
     history = model.fit(X_train, y_train, batch_size=32, epochs=5, validation_data=(X_test, y_test))
```

4. MODEL DEVELOPMEN

MODEL 2:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Bidirectional

# Build BiLSTM model
bilstm_model = Sequential()
bilstm_model.add(Embedding(input_dim=5000, output_dim=128, input_length=100))
bilstm_model.add(Bidirectional(LSTM(128, dropout=0.2, recurrent_dropout=0.2)))
bilstm_model.add(Dense(1, activation='sigmoid'))

bilstm_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train BiLSTM model
bilstm_history = bilstm_model.fit(X_train, y_train, batch_size=32, epochs=5, validation_data=(X_test, y_test))

# Evaluate BiLSTM model
bilstm_loss, bilstm_accuracy = bilstm_model.evaluate(X_test, y_test)
print(f'BiLSTM Accuracy: {bilstm_accuracy}')
```

4. MODEL DEVELOPMEN

MODEL 3:

```
[ ]  from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten

     # Build CNN model
     cnn_model = Sequential()
     cnn_model.add(Embedding(input_dim=5000, output_dim=128, input_length=100))
     cnn_model.add(Conv1D(filters=128, kernel_size=5, activation='relu'))
     cnn_model.add(MaxPooling1D(pool_size=2))
     cnn_model.add(Flatten())
     cnn_model.add(Dense(128, activation='relu'))
     cnn_model.add(Dense(1, activation='sigmoid'))

     cnn_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

     # Train CNN model
     cnn_history = cnn_model.fit(X_train, y_train, batch_size=32, epochs=5, validation_data=(X_test, y_test))

     # Evaluate CNN model
     cnn_loss, cnn_accuracy = cnn_model.evaluate(X_test, y_test)
     print(f'CNN Accuracy: {cnn_accuracy}')
```

4. MODEL DEVELOPMEN

MODEL 4:

```
[ ]  # Build CNN + LSTM model
     cnn_lstm_model = Sequential()
     cnn_lstm_model.add(Embedding(input_dim=5000, output_dim=128, input_length=100))
     cnn_lstm_model.add(Conv1D(filters=128, kernel_size=5, activation='relu'))
     cnn_lstm_model.add(MaxPooling1D(pool_size=2))
     cnn_lstm_model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
     cnn_lstm_model.add(Dense(1, activation='sigmoid'))

     cnn_lstm_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

     # Train CNN + LSTM model
     cnn_lstm_history = cnn_lstm_model.fit(X_train, y_train, batch_size=32, epochs=5, validation_data=(X_test, y_test))

     # Evaluate CNN + LSTM model
     cnn_lstm_loss, cnn_lstm_accuracy = cnn_lstm_model.evaluate(X_test, y_test)
     print(f'CNN + LSTM Accuracy: {cnn_lstm_accuracy}')
```

5. EVALUATION

```
[ ]  import matplotlib.pyplot as plt

     # Evaluate model
     loss, accuracy = model.evaluate(X_test, y_test)
     print(f'Accuracy: {accuracy}')

     # Plot training & validation accuracy values
     plt.plot(history.history['accuracy'])
     plt.plot(history.history['val_accuracy'])
     plt.title('Model accuracy')
     plt.ylabel('Accuracy')
     plt.xlabel('Epoch')
     plt.legend(['Train', 'Test'], loc='upper left')
     plt.show()

     # Plot training & validation loss values
     plt.plot(history.history['loss'])
     plt.plot(history.history['val_loss'])
     plt.title('Model loss')
     plt.ylabel('Loss')
     plt.xlabel('Epoch')
     plt.legend(['Train', 'Test'], loc='upper left')
     plt.show()
```

5. EVALUATION

```
[ ]  import matplotlib.pyplot as plt

     # Evaluate model
     loss, accuracy = model.evaluate(X_test, y_test)
     print(f'Accuracy: {accuracy}')

     # Plot training & validation accuracy values
     plt.plot(history.history['accuracy'])
     plt.plot(history.history['val_accuracy'])
     plt.title('Model accuracy')
     plt.ylabel('Accuracy')
     plt.xlabel('Epoch')
     plt.legend(['Train', 'Test'], loc='upper left')
     plt.show()

     # Plot training & validation loss values
     plt.plot(history.history['loss'])
     plt.plot(history.history['val_loss'])
     plt.title('Model loss')
     plt.ylabel('Loss')
     plt.xlabel('Epoch')
     plt.legend(['Train', 'Test'], loc='upper left')
     plt.show()
```

Multidisciplinary Aspects

The development of a Fake News Detector is multidisciplinary, leveraging insights and techniques from various fields.

Computer Science and Engineering

- Natural Language Processing (NLP): Advanced NLP techniques, including tokenization, lemmatization, and sentiment analysis.
- Machine Learning and Artificial Intelligence: The core of our project relies on machine learning models.
- Data Science: Data preprocessing, augmentation, and analysis.

Linguistics

- Semantics and Pragmatics: Understanding the meaning and context of language helps in distinguishing between truthful and deceptive content.

Multidisciplinary Aspects

Results

```
0s
results = {
    "Model": ["BiLSTM", "CNN", "CNN + LSTM"],
    "Accuracy": [bilstm_accuracy, cnn_accuracy, cnn_lstm_accuracy],
    "Loss": [bilstm_loss, cnn_loss, cnn_lstm_loss]
}

results_df = pd.DataFrame(results)
print(results_df)
```

```
         Model  Accuracy      Loss
0      BiLSTM  0.818750  0.540794
1         CNN  0.808405  0.820783
2  CNN + LSTM  0.809483  0.840110
```

Results

+ Code + Text

[20] 0s

```
def clean_text(text):
    text = text.lower()
    text = re.sub(r'\b\w{1,2}\b', '', text)  # Remove short words
    text = re.sub(r'[^a-z\s]', '', text)  # Remove non-alphabet characters
    text = re.sub(r'\s+', ' ', text)  # Remove extra spaces
    return text

def tokenize_text(text, tokenizer, max_length=100):
    text = clean_text(text)
    sequence = tokenizer.texts_to_sequences([text])
    padded_sequence = pad_sequences(sequence, maxlen=max_length, padding='post')
    return padded_sequence
```

[21] 1s

```
# Assuming tokenizer is the same tokenizer used during training
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(data['cleaned_title'])
```

```
def predict_news(model, news_text, tokenizer):
    processed_text = tokenize_text(news_text, tokenizer)
    prediction = model.predict(processed_text)
    if prediction > 0.5:
        return "Real News"
    else:
        return "Fake News"
```

Conclusion

the Fake News Detector project represents a crucial step towards combating misinformation in the digital age. By leveraging state-of-the-art NLP techniques and sophisticated machine learning models like BERT and RoBERTa, we aim to create a highly accurate and reliable system for distinguishing fake news from real news. This multidisciplinary effort integrates insights from computer science, linguistics, psychology, sociology, and ethics to ensure a comprehensive and contextually aware approach. Our work not only advances the field of AI and NLP but also contributes to fostering a more informed and discerning society. As we move forward, continuous improvement and adaptation of our models will be essential to keep pace with the evolving landscape of fake news, ultimately protecting the integrity of information and supporting public trust.

THANK YOU ALL