Fake News Detection

Group 23

[Our repository]

Lina Tarnueva
Maksim Popov
林奕澄
陳柏諺

What is Fake News?

01. ***Fake News Definition:***
Fake news consists of deliberately fabricated stories intended to mislead and spread false information.

02. ***Spread Mechanism:***
These stories often proliferate through social media platforms like Facebook, Twitter, Instagram, etc.

03. ***Impact:***
The widespread distribution of fake news can lead to significant negative consequences for society, influencing public opinion and causing harm to individuals and communities.

Purpose

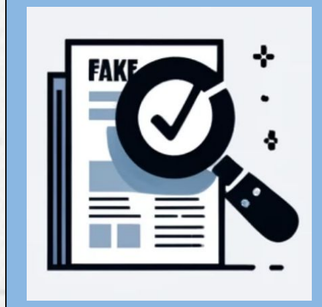
Problem:

- Fake news influences people's perception.
- Fake news is used as clickbait for the profit of content publishers; more clicks mean more money.
- The rise of fake news has become a global problem impacting people and culture, and it casts doubt on the authenticity of journalism.

Purpose:

- This project aims to develop a method for detecting fake news using machine learning.
- The main goal is to identify and predict whether a given news article is fake or not.
- We gathered data, preprocessed text, and translated articles into supervised model features.

Related Work



Early Methods:
Manual verification by fact-checkers and journalists, though effective, is not scalable for the vast amount of content generated daily.

Support Vector Machines (SVM):
Finds the optimal hyperplane to separate fake and real news, effective in high-dimensional spaces.

Natural Language Processing (NLP):
Word Embeddings: Techniques like Word2Vec and GloVe capture semantic meanings.
TF-IDF: Measures word importance in documents relative to a corpus.

Naive Bayes:
Uses word frequency for classification, a simple yet commonly used approach.

Decision Trees and Random Forests:
Apply decision rules based on text features, providing a robust classification method.

Neural Networks: Deep learning models like RNNs and transformers are promising.
LSTM Networks: Capture long-term dependencies in text.
BERT: Achieves state-of-the-art results in NLP tasks, including fake news detection.

Dataset & Platform

01

Source:
Labeled dataset of news articles categorized as true or false.

Composition:
A diverse collection of articles to ensure robust training and evaluation.

02

Preprocessing:

- **Tokenization:** Breaking down text into individual words or tokens.
- **Stop Words Removal:** Removing common words that do not help distinguish between fake and true news.
- **Vectorization:** Converting text data into numerical format using TF-IDF.

03

Language: Python
Libraries:

- **Pandas:** Data manipulation and analysis.
- **Torch:** Deep learning library for building and training neural networks.
- **Scikit-learn:** Machine learning algorithms and models.
- **NLTK:** Natural language processing tasks.

Environment: Jupyter Notebook for interactive development.

Flow Chart

01

Extracting Features

02

Logistic Regression

03

Fully Connected Neural Network

04

Result Analysis

Baseline

01

Extracting Features

02

Logistic Regression

03

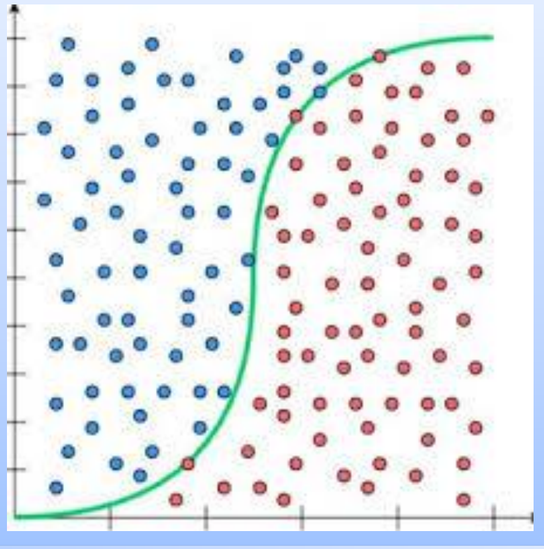
Fully Connected Neural Network

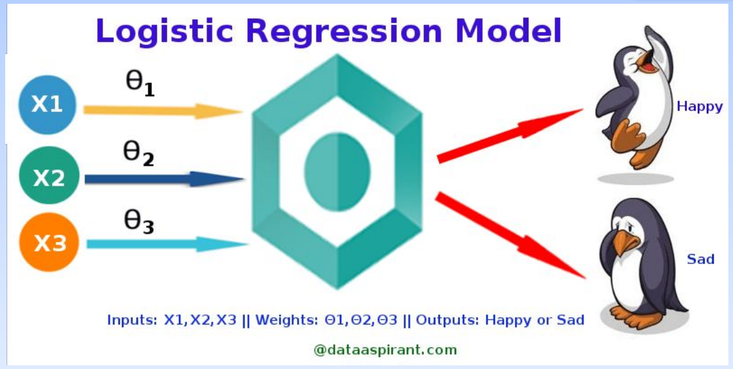
04

Result Analysis

Logistic Regression

$$\frac{1}{1 + e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_n x_n)}}$$






```python
def baseline_model(X):
    X_train, X_test, labl_train, labl_test = train_test_split(X, labl, test_size=0.3, random_state=42)
    model = LogisticRegression()
    model.fit(X_train, labl_train)
    labl_pred = model.predict(X_test)
    accuracy = accuracy_score(labl_test, labl_pred)
    loss = mean_squared_error(labl_test, labl_pred)
    return [loss,accuracy]
```

Main Approach

01

Extracting Features

02

Logistic Regression

03

Fully Connected Neural Network

04

Result Analysis

Extracting Features

Calculate frequency of emotional lexicon

Convert text data to vectors by LLM

Calculate frequency of emotional lexicon

Converting text data to numeric arrays with the frequency of lexicons

```python
def extract_feature(text):
    res = ()

    # Tokenize the text into words and sentences once
    words = word_tokenize(text)
    sentences = sent_tokenize(text)

    # Calculate word frequencies
    word_freq = FreqDist(words)

    # Calculate average sentence length
    avg_sentence_length = sum(len(sent.split()) for sent in sentences) / len(sentences) if sentences else 0
    res += (avg_sentence_length,)

    emotion_counts = count_emotions_in_text(words,nrc_lexicon)
    for _, val in emotion_counts.items():
      res += (val,)

    return np.array(res)
```

```python
def count_emotions_in_text(words, lexicon):
      emotion_counts = {emotion: 0 for emotion in set(emotion for emotions in lexicon.values() for emotion in emotions)}

      for word in words:
        if word in lexicon:
          for emotion in lexicon[word]:
            emotion_counts[emotion] += 1

      return emotion_counts
```

Outcome (Feature)

feature
[30.3, 6.0, 5.0, 11.0, 5.0, 6.0, 4.0, 16.0, 5....
[22.8, 22.0, 17.0, 30.0, 14.0, 6.0, 25.0, 27.0...
[26.11111111111111, 23.0, 24.0, 23.0, 9.0, 11....
[31.833333333333332, 0.0, 0.0, 0.0, 0.0, 0.0, ...
[30.142857142857142, 8.0, 24.0, 21.0, 16.0, 2....

Convert data by LLM

Converting text data to numeric vectors by a pre-trained model in LLM

```python
tokenizer = AutoTokenizer.from_pretrained('sentence-transformers/paraphrase-albert-small-v2')
model = AutoModel.from_pretrained('sentence-transformers/paraphrase-albert-small-v2')

# IMPORTANT: Change "mps" to "cuda" if using not Mac
device = torch.device("mps") if torch.backends.mps.is_available() else torch.device("cpu")
model.to(device)

def vectorisation(text):
    # Tokenize the text
    inputs = tokenizer(text, return_tensors='pt', truncation=True, padding=True, max_length=8).to(device)
    with torch.no_grad():
        outputs = model(**inputs)

    # Get the last hidden states
    last_hidden_states = outputs.last_hidden_state
    pooled_output = torch.mean(last_hidden_states, dim=1)

    return pooled_output.detach().cpu().numpy().flatten()
```

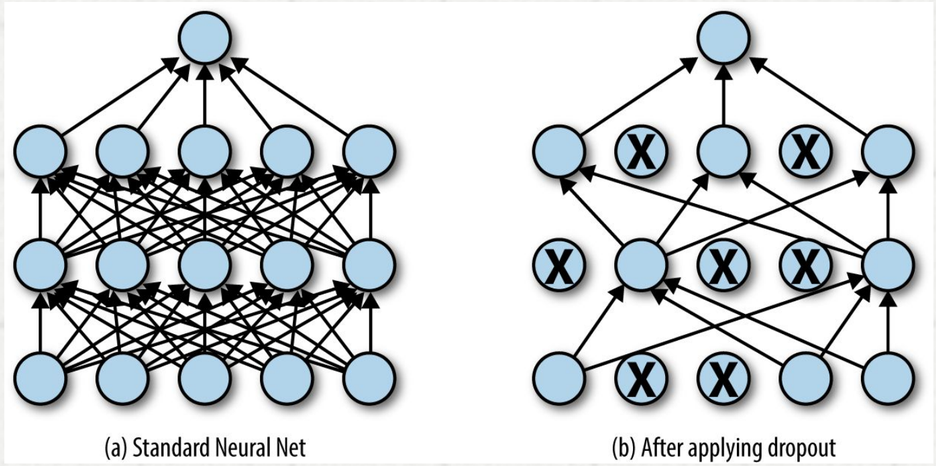
Outcome (Vector)

vector
[0.049537383, 0.18228671, -0.4111063, 0.045929...
[0.0073612956, 0.43443593, -0.13459995, -0.379...
[-0.13830999, 0.37805998, -0.33148205, -0.1998...
[0.3407455, 0.5023093, -0.08773428, 0.1662035,...
[-0.001923956, 0.1068653, -0.19831873, -0.2349...

Compute the mean value of the output layer in the model

Fully Connected Neural Network

- **FCNN with dropout**



Fully Connected Neural Network

- **Our neural**

```
model = Sequential([
    Input(shape=(X.shape[1],)),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

- **3 layers**
- **Drop out with probability = 0.5**
- **ReLu for inner layers**
- **Sigmoid for output**

Fully Connected Neural Network

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

- Adam
- Binary cross-entropy
- Epoch 40
- Batch size 32

```
history = model.fit(X_train, labl_train,
                    epochs=40,
                    batch_size=32,
                    validation_data=(X_test, labl_test))
```

Result Analysis

01
Extracting Features

02
Logistic Regression

03
Fully Connected Neural Network

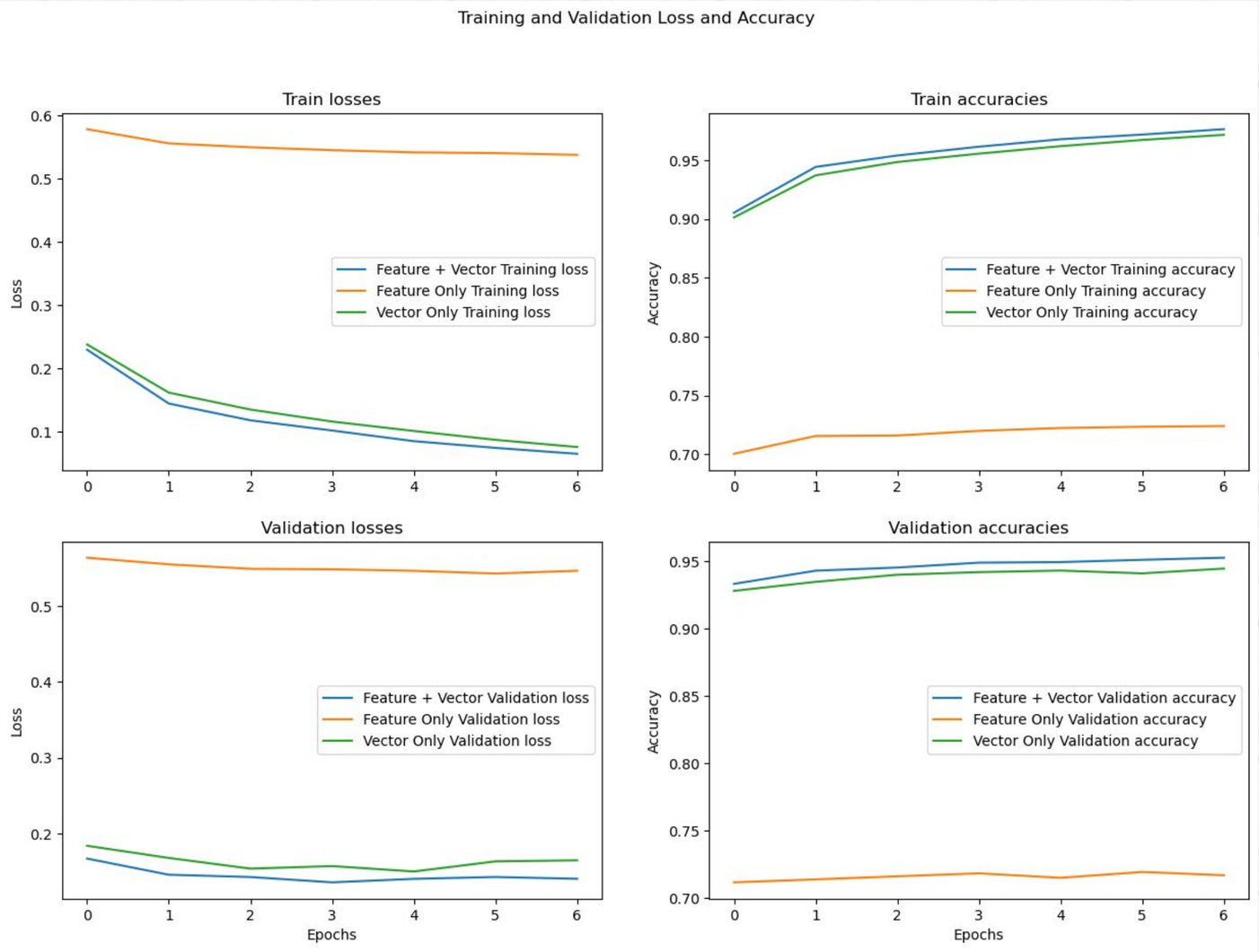
04
Result Analysis

Evaluation Metric

	Accuracy			Loss		
	Feature	Vector	Feature + Vector	Feature	Vector	Feature + Vector
Logistic Regression	60.9%	91.4%	86.9%	0.65	0.21	0.32
CFNN	72.4%	93.7%	94.5%	0.54	0.16	0.15

All the result of accuracy is rounded off to the 1st decimal place
The loss (**Binary cross-entropy**) is rounded off to the 2nd decimal place

Result Analysis

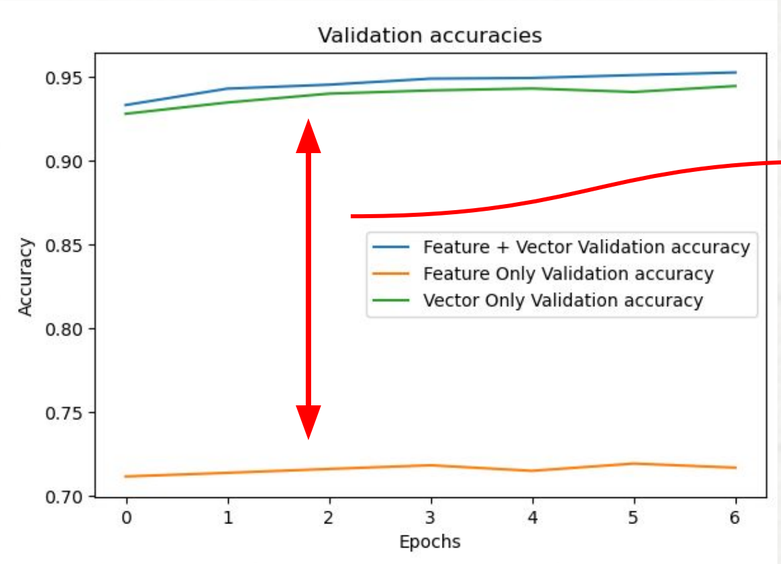


Result Analysis

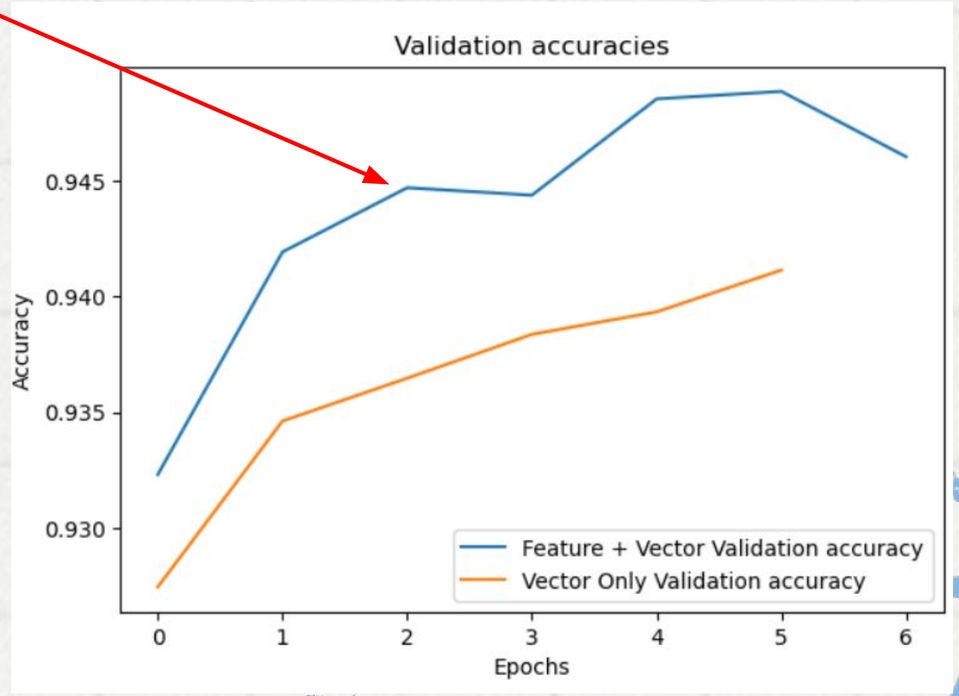
Observation 1

We can get higher accuracy with both feature and vector

Model with only feature performs poor



Huge Gap



Result Analysis

Observation 1

We can get higher accuracy with both feature and vector

Model with only feature performs much worse than others

Possible reason

The relationship between emotional lexicon and fake news might not be strong

→ Both true and fake news might have the same emotion

Result Analysis

Observation 2

CFNN performs better than Logistic Regression

	Accuracy			Loss		
	Feature	Vector	Feature + Vector	Feature	Vector	Feature + Vector
Logistic Regression	60.9%	91.4%	86.9%	0.65	0.21	0.32
CFNN	72.4%	93.7%	94.5%	0.54	0.16	0.15

Result Analysis

Observation 2

CFNN performs better than Logistic Regression

Possible Reason:

Basic structure of our NN

Our NN can be considered as improved logistic regression with several layers

References

Text Features Idea:

The idea for extracting text features was inspired by this paper. While the paper focuses on more general features, our approach specifically analyzes emotional features to detect fake news.
Source : textfeatures

Vectorization Method:

The method for vectorizing text data was adapted from this repository. However, instead of traditional vectorization methods, we employed large language models (LLM) for improved accuracy and efficiency.
Source : repository

Dataset

Contribution

We are confident that each group member has made significant contributions to this project. Here is a breakdown of the contributions by each member:

Lina Tarnueva (25%): Lina focused on creating the presentation and working with the dataset. She was responsible for gathering the dataset and contributing to the development of the presentation slides. Additionally, she was involved in the video recording for the project presentation.

Maksim Popov (25%): Maksim worked on feature extraction and model training. He implemented various feature extraction techniques and trained the machine learning models. Maksim also compiled and edited the video for the project presentation.

林奕澄 (25%): 林奕澄 contributed to the implementation of the neural network models. He set up the fully connected neural network, performed hyperparameter tuning, and optimized the model performance. Additionally, he was involved in the video recording for the project presentation.

陳柏諺 (25%): 陳柏諺 was responsible for creating the presentation and working with the dataset. He also contributed to the development of the presentation slides. Additionally, he was involved in the video recording for the project presentation.

Each member's effort and collaboration were essential in achieving the project's objectives and ensuring its success.

Thank you very much!