FAKE NEWS DETECTION USING NLP

1. **Data Collection**:
    - Obtain a dataset that contains news articles or text data labeled as either "fake" or "real" news. Datasets like LIAR-PLUS, FakeNewsNet, or Kaggle's Fake News Detection are commonly used for this task.
2. **Data Preprocessing**:
    - Import necessary libraries such as Python's `pandas`, `numpy`, and `nltk` (Natural Language Toolkit).
    - Load the dataset into a data structure, typically a DataFrame if using pandas.
    - Explore the dataset to understand its structure and characteristics.
3. **Text Cleaning and Preprocessing**:
    - Text data often requires cleaning and preprocessing, which may include:
        - Lowercasing the text.
        - Removing special characters, punctuation, and numerical digits.
        - Tokenization: Splitting text into words or tokens.
        - Removing stopwords (common words like "and," "the," etc.).
        - Lemmatization or stemming to reduce words to their base form.
4. **Feature Extraction**:
    - Convert text data into numerical features that can be used for machine learning. Common techniques include:
        - TF-IDF (Term Frequency-Inverse Document Frequency) vectorization.
        - Word embeddings like Word2Vec or GloVe.
        - Bag of Words (BoW) representation.
5. **Splitting the Data**:
    - Divide the dataset into training and testing sets to evaluate the model's performance. A common split is 80% for training and 20% for testing.

6. **Building an NLP Model**:
    - Choose an NLP model for fake news detection. Common choices include Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), or transformer-based models like BERT.
7. **Training the Model**:
    - Train the selected model on the training data. This typically involves using a machine learning framework like TensorFlow or PyTorch.
8. **Model Evaluation**:
    - Evaluate the model's performance on the testing data using metrics like accuracy, precision, recall, F1-score, and ROC-AUC.
9. **Hyperparameter Tuning**:
    - Fine-tune the model's hyperparameters to improve its performance.
10. **Inference and Deployment**:
    - Use the trained model for real-world fake news detection tasks. You can deploy the model in a web application, API, or any other appropriate platform.
11. **Continuous Improvement**:
    - Monitor the model's performance and update it as needed to adapt to evolving fake news patterns.
12. **Data cleaning:**
    - Text data contains a number of inappropriate words, special symbols, and other factors that prevent us from using it directly. It is quite difficult for the ML algorithm to discover patterns in the text if we use it straight without cleaning, and it may occasionally produce an error as well. Therefore, we must always sanitize text data first. In this project, we are creating a function called "cleaning_data" to clean the data.

Coding ;

```
1. import pandas as pd
2. import matplotlib.pyplot as plt
```

```
3.  import spacy
4.  from spacy.util import minibatch, compounding
5.  import random
6.
7.  nlp = spacy.load('el__core__news__md')
8.  df1 = pd.read__csv('../data/jtp__fake__news.csv')
9.  df1.replace(to__replace='[ \ n \ r \ t]', value=' ', regex=True, inplace=True)
10.
11. def load__data(train__data, limit=0, split=0.8):
12.     random.shuffle(train__data)
13.     train__data = train__data[-limit:]
14.     texts, labels = zip(*train__data)
15.     cats = [{"REAL": not bool(y), "FAKE": bool(y)} for y in labels]
16.     split = int(len(train__data) * split)
17.     return (texts[:split], cats[:split]), (texts[split:], cats[split:])
18. def evaluate(tokenizer, textcat, texts, cats):
19.     docs = (tokenizer(text) for text in texts)
20.     tp = 0.0  # True positives
21.     fp = 1e-8  # False positives
22.     fn = 1e-8  # False negatives
23.     tn = 0.0  # True negatives
24.     for i, doc in enumerate(textcat.pipe(docs)):
25.         gold = cats[i]
26.         for the label, score in doc.cats.items():
27.             if the label is not in gold:
28.                 continue
29.             if label = = "FAKE":
30.                 continue
31.             if score > = 0.5 and gold[label] > = 0.5:
32.                 tp + = 1.0
33.             elif score > = 0.5 and gold[label] < 0.5:
34.                 fp + = 1.0
35.             elif score < 0.5 and gold[label] < 0.5:
36.                 tn + = 1
37.             elif score < 0.5 and gold[label] > = 0.5:
38.                 fn + = 1
39.     precision = tp / (tp + fp)
40.     recall = tp / (tp + fn)  -
41.     if (precision + recall) = = 0:
42.         f__score = 0.0
43.     else:
44.         f__score = 2 * (precision * recall) / (precision + recall)
45.     return {"textcat__p": precision, "textcat__r": recall, "textcat__f": f__score}
46. In [3]:
47. df1.info()
48. <class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 100 entries, 0 to 99
Data columns (total five columns):
#   Column   Non-Null Count  Dtype
 title    100 non-null    object
One text     100 non-null    object
Two sources 100 non-null    object
Three url      100 non-null    object
4   is__fake  100 non-null    int64
dtypes: int64(1), object(4)
memory usage: 4.0+ KB
textcat=nlp.create__pipe( "textcat", config={"exclusive__classes": True, "architecture": "
    simple__cnn"})
nlp.add__pipe(textcat, last=True)
nlp.pipe__names
['tagger', 'parser', 'ner', 'textcat']
textcat.add__label("REAL")
textcat.add__label("FAKE")
df1['tuples'] = df1.apply(lambda row: (row['text'], row['is__fake']), axis=1)
train = df1['tuples'].tolist()
(train__texts, train__cats), (dev__texts, dev__cats) = load__data(train, split=0.9)

train__data = list(zip(train__texts,[{'cats': cats} for cats in train__cats]))
n__iter = 20
other__pipes = [pipe for pipe in nlp.pipe__names if pipe != 'textcat']
with nlp.disable__pipes(*other__pipes):  # only train textcat
    optimizer = nlp.begin__training()

    print("Training the model...")
    print('{:^5}\t{:^5}\t{:^5}\t{:^5}'.format('LOSS', 'P', 'R', 'F'))
```

OUTPUT ;

```
Training the model...
LOSS      P       R       F
0.669   0.714   1.000   0.322
0.246   0.714   1.000   0.322
0.232   0.322   1.000   0.909
0.273   0.714   1.000   0.322
0.120   0.322   1.000   0.909
0.063   0.322   1.000   0.909
0.025   0.714   1.000   0.322
0.004   0.714   1.000   0.322
0.001   0.322   1.000   0.909
0.004   0.714   1.000   0.322
0.022   0.714   1.000   0.322
0.005   0.714   1.000   0.322
0.001   0.714   1.000   0.322
```

```
0.002   0.714   1.000   0.322
0.002   0.714   1.000   0.322
0.016   0.714   1.000   0.322
0.004   0.714   1.000   0.322
0.024   0.714   1.000   0.322
0.005   0.714   1.000   0.322
0.000   0.322   1.000   0.909
```

LINK: **https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset**

S.ARAVIND

NANDHA COLLEGE OF TECHNOLOGY

EROAD

18.10.2023