Write a python program where you have to check that an article is AI generated or Human generated which gives us at least 70 % accuracy. Compare results for TF-IDF, Word2Vec, and GloVe embeddings for feature extraction over classical machine learning models (e.g., Logistic Regression, Naïve Bayes, SVM, or Random Forest) for classification. Use dataset https://www.kaggle.com/datasets/sunilthite/llm-detect-ai-generated-text-dataset

```
import nltk
import shutil
import os

# Define the path where you want to store NLTK data
nltk_data_path = os.path.expanduser('~/nltk_data')

# Remove the directory if it exists (optional, for a clean installa
if os.path.exists(nltk_data_path):
    shutil.rmtree(nltk_data_path)

# Create the directory if it doesn't exist
os.makedirs(nltk_data_path, exist_ok=True)

# Set the NLTK data path
nltk.data.path.append(nltk_data_path)

# Download the 'punkt_tab' resource to the specified path
nltk.download('punkt_tab', download_dir=nltk_data_path)

# Now you can use word_tokenize
from nltk.tokenize import word_tokenize
print(word_tokenize("Hello world!"))
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data..
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
['Hello', 'world', '!']
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import nltk
from nltk.tokenize import word_tokenize
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Load dataset
url = '/content/drive/MyDrive/Colab Notebooks/NLP/Exp-11/Training_
data = pd.read_csv(url)

# Preprocess data
```

Gemini LookupError

Please explain the error:

```
LookupError:
**********************************
  Resource punkt_tab not found.
  Please use the NLTK Downloader to ob

  >>> import nltk
  >>> nltk.download('punkt_tab')

  For more information see: https://ww

  Attempted to load tokenizers/punkt_t

  Searched in:
    - '/root/nltk_data'
    - '/usr/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/local/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/local/lib/nltk_data'
**********************************
```

Hypothesis

The error message `LookupError: Resource punkt_tab not found` indicates that the NLTK data package `punkt_tab` is missing. This package is required by `nltk.tokenize.word_tokenize` to properly tokenize the text into words based on punctuation. Although it seems you have downloaded `punkt`, you need to download an additional data package called `punkt_tab`. The `punkt` package only provides the tokenizer that handles basic tokenization. The `punkt_tab` package is needed to refine the sentences that the tokenizer provides and this is needed to identify where sentence boundaries are more accurately. Without it, NLTK is unable to perform accurate sentence splitting using `nltk.tokenize.word_tokenize`

Suggested Changes

```
import nltk

# Download the necessary NLTK data pa
```

```
data['text'] = data['text'].apply(lambda x: x.lower())
X = data['text']
y = data['generated']

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_siz

# TF-IDF Feature Extraction
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Train and evaluate models using TF-IDF
models = {
    'Naive Bayes': MultinomialNB(),
    'Logistic Regression': LogisticRegression(),
    'SVM': SVC(),
    'Random Forest': RandomForestClassifier()
}

for name, model in models.items():
    model.fit(X_train_tfidf, y_train)
    y_pred = model.predict(X_test_tfidf)
    print(f'{name} Accuracy (TF-IDF): {accuracy_score(y_test, y_pr

# Word2Vec Feature Extraction
tokenized_train = [word_tokenize(text) for text in X_train]
tokenized_test = [word_tokenize(text) for text in X_test]

word2vec_model = Word2Vec(sentences=tokenized_train, vector_size=1
X_train_w2v = np.array([np.mean([word2vec_model.wv[word] for word
X_test_w2v = np.array([np.mean([word2vec_model.wv[word] for word i

# Train and evaluate models using Word2Vec
for name, model in models.items():
    model.fit(X_train_w2v, y_train)
    y_pred = model.predict(X_test_w2v)
    print(f'{name} Accuracy (Word2Vec): {accuracy_score(y_test, y_

# GloVe Feature Extraction
glove_vectors = KeyedVectors.load_word2vec_format('glove.6B.100d.w
X_train_glove = np.array([np.mean([glove_vectors[word] for word ir
X_test_glove = np.array([np.mean([glove_vectors[word] for word in

# Train and evaluate models using GloVe
for name, model in models.items():
    model.fit(X_train_glove, y_train)
    y_pred = model.predict(X_test_glove)
    print(f'{name} Accuracy (GloVe): {accuracy_score(y_test, y_pre
```

```
Naive Bayes Accuracy (TF-IDF): 0.97
Logistic Regression Accuracy (TF-IDF): 0.99
SVM Accuracy (TF-IDF): 1.00
Random Forest Accuracy (TF-IDF): 0.99
```

```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('punkt_tab') # Downloac

# ... (rest of your code) ...
```

Use code with caution

Rate this answer

Enter a prompt here

0 / 2000

Responses may display inaccurate or offensive information that doesn't represent Google's views. Learn more