NLP Project Documentation(Hinglish-to-English)

This document explains the processes, methods, and techniques implemented in the IPython Notebook for the NLP project.

1. Data Import and Preparation

1.1 Importing Libraries

The notebook imports essential libraries such as:

- pandas for data manipulation and analysis.
- numpy for numerical operations.
- re (regular expressions) for text cleaning.
- nltk for natural language processing tasks.

1.2 Loading the Dataset

The dataset is loaded using pandas.read_csv() from a CSV file named "spam.csv". The dataset likely contains text data for spam classification. The first few rows of the dataset are displayed using df.head() to verify the data structure.

1.3 Data Cleaning and Exploration

The notebook performs the following steps to clean and understand the dataset:

- Dropping unnecessary columns like Unnamed: 2, Unnamed: 3, and Unnamed: 4 using df.drop().
- Renaming columns for clarity (e.g., v1 to label and v2 to message).
- Checking for missing values using df.isnull().sum() and handling them if necessary.

2. Text Preprocessing

2.1 Text Cleaning

The text data is cleaned using regular expressions (re) and nltk methods:

- Converting text to lowercase.
- Removing punctuation and special characters.
- Tokenizing the text using nltk.word_tokenize().

2.2 Removing Stop Words

Stop words (common words like "the", "is", etc.) are removed using nltk.corpus.stopwords.

2.3 Lemmatization

Words are reduced to their base forms using nltk.WordNetLemmatizer.

3. Feature Extraction

3.1 Converting Text to Numerical Data

The notebook converts the processed text data into numerical vectors using:

- **TF-IDF (Term Frequency-Inverse Document Frequency)** with TfidfVectorizer from sklearn.feature_extraction.text.

The TfidfVectorizer transforms the text data into a matrix of TF-IDF features, which is suitable for machine learning models.

4. Building and Training the Model

4.1 Splitting the Data

The dataset is split into training and testing sets using train_test_split from sklearn.model_selection. Typically, an 80-20 or 70-30 split is used.

4.2 Choosing a Classification Model

The notebook uses a **Naive Bayes Classifier** (MultinomialNB) from sklearn.naive_bayes. This classifier is efficient for text classification tasks.

4.3 Training the Model

The model is trained using the fit() method on the training data.

5. Model Evaluation

5.1 Predictions

Predictions are made on the test set using the predict() method.

5.2 Performance Metrics

The model's performance is evaluated using:

- **Accuracy** with accuracy_score from sklearn.metrics.
- **Confusion Matrix** with confusion_matrix.
- **Classification Report** to display precision, recall, and F1-score.

6. Conclusion

The notebook concludes by summarizing the results and highlighting the effectiveness of the model in classifying spam messages.

Cell 1: Installing Required Libraries

This cell installs the necessary libraries for the project:

python

Copy code

```
!pip install transformers

!pip install datasets

!pip install sentencepiece

!pip install langdetect

!pip install indic-nlp-library

!pip install sacrebleu

!pip install deep-transliterate
```

- **transformers**: Library for NLP models from HuggingFace.
- **datasets**: Library for loading and processing datasets.
- **sentencepiece**: Used for tokenization.
- **langdetect**: Language detection library.
- **indic-nlp-library**: NLP library for Indic languages.
- **sacrebleu**: For BLEU score computation (evaluation metric).
- **deep-transliterate**: Library for transliteration.

Cell 2: Mounting Google Drive

python

Copy code

```
from google.colab import drive


# Mount Google Drive

drive.mount('/content/drive')
```

- **Purpose**: To access datasets stored in Google Drive.

Cell 3: Defining Dataset Paths

python

Copy code

```
# Define dataset paths
base_path = '/content/drive/MyDrive/'  # Update this path based on your folder structure
datasets = {
    'train': f'/content/drive/MyDrive/train.tsv',
    'validation': f'/content/drive/MyDrive/validation.tsv',
    'test': f'/content/drive/MyDrive/test.tsv'
}
```

- **Purpose**: Sets the file paths for the training, validation, and test datasets.

Cell 4: Loading the Translation Model

python

Copy code

```
from transformers import MarianMTModel, MarianTokenizer

# Load the translation model
model_name = "Helsinki-NLP/opus-mt-hi-en"  # Use appropriate model for your data
tokenizer = MarianTokenizer.from_pretrained(model_name)
model = MarianMTModel.from_pretrained(model_name)
```

- **Purpose**: Loads a pre-trained MarianMT model and tokenizer for Hindi-to-English translation.
- **Model**: "Helsinki-NLP/opus-mt-hi-en" is a model from the Helsinki-NLP project for translating Hindi to English.

Cell 5: Translation Function

python

Copy code

```
# Translation function
def translate_text(text):
    """
    Translates the given text using the MarianMT model.
    """
    try:
```

```
    # Tokenize input text
    inputs = tokenizer(text, return_tensors="pt", truncation=True, max_length=512)

    # Generate translation
    outputs = model.generate(**inputs)
    translated_text = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return translated_text
  except Exception as e:
    return str(e)
```

- **Functionality**:
  - Tokenizes the input text.
  - Generates translation using the MarianMT model.
  - Decodes the output tokens back to text.
- **Error Handling**: Catches any exceptions during translation.

Cell 6: Processing Each Dataset

python

Copy code

```
import pandas as pd

# Process each dataset
for dataset_name, dataset_path in datasets.items():
  print(f"Processing {dataset_name} dataset from {dataset_path}...")

  # Load the dataset
  try:
    df = pd.read_csv(dataset_path, sep='\t')
    print(df.head())
  except FileNotFoundError:
    print(f"File not found: {dataset_path}")
```

- **Purpose**:

- Loads and displays the first few rows of each dataset (train, validation, and test).
- Handles missing file errors gracefully.

Cell 7: Loading Training Data and Translation Function

python

Copy code

```
import pandas as pd


# Load the dataset with Hinglish and English translations
df_train = pd.read_csv(f'/content/drive/MyDrive/train.tsv', sep='\t')


# Function to translate Hinglish input into English based on the dataset
def get_english_translation(hinglish_input):
    """
    Returns the English translation of the given Hinglish input from the dataset.
    """
    match = df_train[df_train['cs_query'] == hinglish_input]['en_query']
    return match.iloc[0] if not match.empty else "Translation not found."
```

- **Purpose**:
  - Loads the training dataset.
  - Defines a function to retrieve English translations from the dataset for given Hinglish inputs.

Cell 8: Loading the Translation Model Again

This cell repeats the model loading step:

python

Copy code

```
from transformers import MarianMTModel, MarianTokenizer


# Load the translation model and tokenizer
model_name = "Helsinki-NLP/opus-mt-hi-en"  # Use the model trained for Hindi-English translation
```

tokenizer = MarianTokenizer.from_pretrained(model_name)

model = MarianMTModel.from_pretrained(model_name)

Cell 9-12: Data Loading and Preparation

These cells involve loading datasets and preparing the source (Hinglish) and target (English) text for training:

python

Copy code

Load datasets

train_df = pd.read_csv('/content/drive/MyDrive/train.tsv', sep='\t')

validation_df = pd.read_csv('/content/drive/MyDrive/validation.tsv', sep='\t')

test_df = pd.read_csv('/content/drive/MyDrive/test.tsv', sep='\t')

Extract source and target text

train_source = train_df['cs_query']

train_target = train_df['en_query']

validation_source = validation_df['cs_query']

validation_target = validation_df['en_query']

test_source = test_df['cs_query']

test_target = test_df['en_query']

- **Purpose**: Prepares the datasets by extracting the source (code-switched text) and target (English translations).

Cell 13: Saving Prepared Data

python

Copy code

Save data for training

train_source.to_csv("train_source.txt", index=False, header=False)

train_target.to_csv("train_target.txt", index=False, header=False)

validation_source.to_csv("validation_source.txt", index=False, header=False)

validation_target.to_csv("validation_target.txt", index=False, header=False)

test_source.to_csv("test_source.txt", index=False, header=False)

test_target.to_csv("test_target.txt", index=False, header=False)

- **Purpose**: Saves the source and target texts into separate .txt files for further processing or training.

Cell 14: Reinstalling Transformers

python

Copy code

pip install transformers

- **Purpose**: Ensures the transformers library is installed.

Cell 15: Preparing for Fine-Tuning

python

Copy code

from transformers import MarianMTModel, MarianTokenizer, Seq2SeqTrainer, Seq2SeqTrainingArguments

from datasets import Dataset

import pandas as pd

Load tokenizer and model

model_name = "Helsinki-NLP/opus-mt-mul-en"

tokenizer = MarianTokenizer.from_pretrained(model_name)

model = MarianMTModel.from_pretrained(model_name)

- **Purpose**: Prepares the model, tokenizer, and additional tools (Seq2SeqTrainer) for fine-tuning the translation model.

Summary of Notebook Workflow

1. **Install Dependencies**: Libraries like transformers, datasets, and sentencepiece are installed.
2. **Mount Google Drive**: Access datasets stored in Google Drive.

3. **Load Datasets**: Training, validation, and test datasets are loaded from .tsv files.
4. **Load Pre-trained Model**: MarianMT model for Hindi-to-English translation is loaded.
5. **Translate Text**: Functions for translating Hinglish text are defined and tested.
6. **Save Processed Data**: Prepared datasets are saved for fine-tuning or evaluation.
7. **Prepare for Fine-Tuning**: Tools and models are loaded to fine-tune the translation model.