Overview of the Code(Tinglish-to-English)

This code provides a solution for language identification and translation of Telugu-English code-mixed sentences. The project aims to identify which words in a sentence are in Telugu and which are in English, then translate the identified Telugu words into English. The implementation includes loading a dataset, training or using a pre-trained CRF model for language identification, translating the identified Telugu words, and saving the translated sentences into a file.

1. Importing Required Libraries

python

Copy code

```
import os

import pickle
```

- **os**: A Python library to interact with the operating system, but it's not used in this specific case.
- **pickle**: A Python library used for serializing and deserializing Python objects (in this case, the CRF model).

---

2. Defining File Paths

python

Copy code

```
# Define paths

dataset_path = r"C:\Users\lenovo\Downloads\tempp.txt"

model_path = r"D:\internship\AI and ML\AI and ML project\apurupa\Word-Level-Language-Identification-in-English-Telugu-Code-Mixed-Data\crf.model"
```

- **dataset_path**: The file path where the dataset (tempp.txt) is stored. This file contains the word-level labels indicating whether a word is in Telugu or English.
- **model_path**: The file path where the pre-trained CRF model (crf.model) is stored. This model will be used to predict the language label for each word in the input dataset.

---

3. Loading the Dataset

python

Copy code

```
# Function to read and process the dataset

def load_dataset(file_path):

    sentences = []
```

```
labels = []
current_sentence = []
current_labels = []

with open(file_path, 'r', encoding='utf-8') as file:
    for line in file:
        line = line.strip()
        if line == "":
            # End of a sentence
            if current_sentence:
                sentences.append(current_sentence)
                labels.append(current_labels)
                current_sentence = []
                current_labels = []
        else:
            word, label = line.split("\t")
            current_sentence.append(word)
            current_labels.append(label)

    # Append the last sentence if file doesn't end with a blank line
    if current_sentence:
        sentences.append(current_sentence)
        labels.append(current_labels)

return sentences, labels
```

- **load_dataset**: A function that reads the dataset file and processes it into sentences and their corresponding labels.
    - **Input**: The file at file_path.
    - **Output**: Two lists:
        - **sentences**: A list of sentences where each sentence is a list of words.
        - **labels**: A list of sentences where each sentence is a list of labels (e.g., "Telugu" or "English").

- **Processing**:
  - It reads each line in the file.
  - If the line is blank, it considers the end of the current sentence.
  - The words and labels in the non-blank lines are split by a tab (\t) and added to current_sentence and current_labels.
  - Once a sentence is processed (when a blank line is encountered), it's added to the sentences and labels lists.

---

4. Feature Extraction for CRF

python

Copy code

```
# Function to extract features for CRF
def word2features(sent, i):
    word = sent[i]
    features = {
        'word.lower()': word.lower(),
        'word.isupper()': word.isupper(),
        'word.istitle()': word.istitle(),
        'word.isdigit()': word.isdigit(),
    }
    if i > 0:
        word1 = sent[i - 1]
        features.update({
            '-1:word.lower()': word1.lower(),
            '-1:word.isupper()': word1.isupper(),
            '-1:word.istitle()': word1.istitle(),
        })
    else:
        features['BOS'] = True

    if i < len(sent) - 1:
        word1 = sent[i + 1]
```

```
    features.update({
        '+1:word.lower()': word1.lower(),
        '+1:word.isupper()': word1.isupper(),
        '+1:word.istitle()': word1.istitle(),
    })
else:
    features['EOS'] = True

return features
```

- **word2features**: A function that extracts linguistic features for a word in a sentence. These features are used by the CRF model to make predictions.
    - **Input**:
        - sent: The sentence (list of words).
        - i: The index of the current word in the sentence.
    - **Output**: A dictionary containing features for the word at position i.
        - **Basic Features**:
            - word.lower(): The lowercase version of the word.
            - word.isupper(): Whether the word is in uppercase.
            - word.istitle(): Whether the word is capitalized as in a title.
            - word.isdigit(): Whether the word consists only of digits.
        - **Context Features**:
            - Features of neighboring words (if available). For example, -1:word.lower() refers to the previous word in the sentence.
            - BOS (Beginning of Sentence) and EOS (End of Sentence) are added to handle the edges of the sentence.

python

Copy code

```
def sent2features(sent):
    return [word2features(sent, i) for i in range(len(sent))]
```

- **sent2features**: A function that applies word2features to every word in a sentence to create a list of feature dictionaries for the entire sentence.

---

5. Loading the CRF Model

python

Copy code

```
# Load the trained CRF model
with open(model_path, 'rb') as model_file:
    crf_model = pickle.load(model_file)
```

- **Loading the CRF Model**: The pickle.load function is used to load the pre-trained CRF model from the specified model_path.
    - **Input**: A path to the serialized CRF model (crf.model).
    - **Output**: A trained CRF model that can be used to predict language labels for words in sentences.

---

6. Predicting Language Labels

python

Copy code

```
# Predict labels for the dataset
X = [sent2features(sentence) for sentence in sentences]
predicted_labels = crf_model.predict(X)
```

- **sent2features** is applied to each sentence in the dataset to convert them into a list of feature sets (X).
- The CRF model is used to predict the language label for each word in the sentence (predicted_labels).

---

7. Translation Function (Placeholder)

python

Copy code

```
# Function to translate Telugu words to English
# Replace this with a proper translation function if needed
def translate_to_english(word):
    # Placeholder translation logic
    # In a real application, use a translation API or library
    translations = {
```

```
    "nenu": "I",
    "vasthanu": "will come",
    "idi": "this",
}
return translations.get(word, word)  # Return the word itself if no translation is found
```

- **translate_to_english**: A simple placeholder function that translates a Telugu word to English. In practice, you would replace this with a more sophisticated translation system (e.g., using an API like Google Translate).

---

8. Translating the Sentences

python

Copy code

```
# Translate sentences
translated_sentences = []
for sentence, labels in zip(sentences, predicted_labels):
    translated_sentence = []
    for word, label in zip(sentence, labels):
        if label == "Telugu":
            translated_sentence.append(translate_to_english(word))
        else:
            translated_sentence.append(word)
    translated_sentences.append(" ".join(translated_sentence))
```

- This part processes each sentence and translates the words that are labeled as "Telugu" (identified by the CRF model).
    - **For each word**: If the label is "Telugu", it calls the translate_to_english function; otherwise, it keeps the word as is.
    - The result is a translated sentence where Telugu words are replaced by their English equivalents.

---

9. Saving the Translated Sentences to a File

python

Copy code

```
# Save translated sentences to a file
```

```
output_path = r"C:\Users\lenovo\Downloads\translated_output.txt"
with open(output_path, "w", encoding="utf-8") as output_file:
    for translated_sentence in translated_sentences:
        output_file.write(translated_sentence + "\n")
print(f"Translated sentences saved to {output_path}")
```

- This part saves the translated sentences to a file (translated_output.txt).
    - It writes each translated sentence to a new line in the output file.
    - **Output**: A file containing all the translated sentences.

Summary of the Workflow

1. **Load Dataset**: Read the dataset and split sentences and labels.
2. **Feature Extraction**: Convert sentences into a format suitable for the CRF model using feature extraction.
3. **Load Pre-trained CRF Model**: Load a pre-trained CRF model for language identification.
4. **Predict Language Labels**: Use the model to identify the language of each word (Telugu or English).
5. **Translate Telugu Words**: Translate the identified Telugu words into English using a placeholder translation function.
6. **Save Translated Sentences**: Write the translated sentences to a file.

This should give you a complete understanding of each part of the code. Let me know if you need further clarification!