Project -4

Prediction of the Next Word Using Python Language

Predicting the next word in a sequence is a common problem in natural language processing (NLP) and can be done using various machine learning techniques. In this response, I'll provide a simple example of how to predict the next word in a sentence using the n-gram language model in Python.

First, you'll need to install the Natural Language Toolkit (NLTK) library, which provides various tools and resources for NLP tasks. You can install it using pip:

pip install nltk

Next, you'll need to download the necessary resources from NLTK, including the corpus of text data used for training the language model. Open a Python shell and enter the following commands:

import nltk

nltk.download('punkt')

nltk.download('corpora/brown')

Once you have NLTK and the necessary resources installed, you can use the following code to train a simple n-gram language model and use it to predict the next word in a sentence:

```
import nltk
from nltk.corpus import brown
from nltk.util import ngrams

# Load the corpus of text data
corpus = brown.sents()

# Compute the frequency distribution of n-grams for n=1,2,3
ngram_freqs = {}
for n in range(1, 4):
    ngram_freqs[n] = nltk.FreqDist(ngrams([word.lower() for sent in corpus for word in sent], n))

# Predict the next word in a sentence using the language model
def predict_next_word(sentence):
    # Compute the probability distribution of the next word for n=1,2,3
    probs = {}
    for n in range(1, 4):
        context = sentence.split()[-n:]
```

```
        prob_dist = nltk.MLEProbDist(ngram_freqs[n][tuple(context)], bins=len(ngram_freqs[n]))
        probs[n] = prob_dist.max()

    # Choose the n-gram with the highest probability and return the next word
    best_n = max(probs, key=probs.get)
    context = sentence.split()[-best_n:]
    return ngram_freqs[best_n].max(context)

# Example usage
sentence = "The quick brown"
next_word = predict_next_word(sentence)
print(f"The predicted next word for '{sentence}' is '{next_word}'.")
```

This code first loads the Brown Corpus, a collection of text data from various sources, using NLTK's brown.sents() function. It then computes the frequency distribution of n-grams for n=1,2,3 using NLTK's FreqDist() function and stores them in a dictionary ngram_freqs.

The predict_next_word() function takes a sentence as input and uses the language model to compute the probability distribution of the next

word for n=1,2,3 using NLTK's MLEProbDist() function. It then chooses the n-gram with the highest probability and returns the next word using the max() function of the FreqDist object.

Finally, the code demonstrates the usage of predict_next_word() by predicting the next word in the sentence "The quick brown" and printing the result.

Note that this is a simple example and there are many ways to improve the performance of the language model, such as using a larger corpus of text data, smoothing techniques, and more sophisticated algorithms.