Ex 1a: Text Generation using N-gram Language model

Learning Objective:

Implement an N-Gram–based predictive text model using bigram and trigram probability distributions. This helps to understand how contextual probability influences next-word Prediction.

Steps:

1. Initialize the environment by installing and importing the Natural Language Toolkit (nltk). Download the Brown Corpus (for text data) and the Universal Tagset (for simplified Part-of-Speech tags).
2. Extract raw sentences from the Brown Corpus using brown.sents().
3. Tokenization: Convert all words to lowercase and flatten the nested list into a single list of tokens to verify the total volume of training data.
4. Build an N-Gram Model for Bigram and Trigram.
5. Predict Next Word and display Top-5 Suggestions
6. Test Bigram & Trigram Prediction on certain words.

Program:

```
import nltk
from nltk.corpus import brown
from nltk.util import ngrams
from collections import Counter, defaultdict
nltk.download('brown')
nltk.download('universal_tagset')
sentences = brown.sents()
print("Total sentences:", len(sentences))
print(sentences[0])

tokens = [word.lower() for sent in sentences for word in sent]
print("Total tokens:", len(tokens))
```

```
print(tokens[:20])

def predict_bigram_prob(word, top_n=5):
    counter = bigram_model[word]

    total = sum(counter.values())

    results = []

    for next_word, count in counter.items():
        prob = count / total
        results.append((next_word, round(prob, 4)))

    results.sort(key=lambda x: x[1], reverse=True)

    return results[:top_n]

def predict_trigram_prob(w1, w2, top_n=5):
    counter = trigram_model[(w1, w2)]

    total = sum(counter.values())

    results = []

    for next_word, count in counter.items():
        prob = count / total
        results.append((next_word, round(prob, 4)))

    results.sort(key=lambda x: x[1], reverse=True)

    return results[:top_n]

print("Bigram predictions for 'the':")
print(predict_bigram_prob('the'))
```

```
print("\nTrigram predictions for ('in','the'):")
print(predict_trigram_prob('in','the'))
```

Output:

[nltk_data] Downloading package brown to /root/nltk_data...

[nltk_data] Unzipping corpora/brown.zip.

[nltk_data] Downloading package universal_tagset to /root/nltk_data...

[nltk_data] Unzipping taggers/universal_tagset.zip.

True

Total sentences: 57340

['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an', 'investigation', 'of', "Atlanta's", 'recent', 'primary', 'election', 'produced', '``', 'no', 'evidence', "''", 'that', 'any', 'irregularities', 'took', 'place', '.']

Total tokens: 1161192

['the', 'fulton', 'county', 'grand', 'jury', 'said', 'friday', 'an', 'investigation', 'of', "atlanta's", 'recent', 'primary', 'election', 'produced', '``', 'no', 'evidence', "''", 'that']

Bigram predictions for 'the':

[('first', 0.0095), ('same', 0.009), ('most', 0.006), ('other', 0.0059), ('``', 0.0058)]

Trigram predictions for ('in','the'):

[('world', 0.0149), ('first', 0.0146), ('united', 0.0123), ('same', 0.0116), ('past', 0.0098)]

Learning Outcome:

Upon completion of this experiment, the bigram and trigram probability distributions for next-word prediction was implemented and evaluated using the Brown Corpus.