Q1- Report on Implementing an N-Gram Language Model and Testing Perplexity

N-Gram Language Model Report

Implementation Approach

Overview:

The goal of this project was to implement an N-Gram language model in Python using a provided text corpus, train it, and evaluate its performance by calculating the perplexity on a test set. The N-Gram model allows for the prediction of the next word in a sequence based on the previous n−1 words.

Data Preprocessing

1. Loading and Cleaning Data:

- The training and testing datasets were loaded from train.txt and test.txt.
- The text was converted to lowercase, and special characters were removed to ensure uniformity.
- The cleaned text was then tokenized into individual words to facilitate the construction of N-Grams.

2. Vocabulary Creation:

- A vocabulary set was created from the unique tokens in the training data. This set is used for probability calculations and to handle unseen words during testing.

N-Gram Generation

- The model supports different n values (unigram, bigram, trigram) through a flexible method for generating N-Grams.
- For each n-gram, counts were maintained to calculate probabilities later.

Probability Calculation

- The probabilities of each word given its preceding n−1 words were computed using add-one smoothing (Laplace smoothing) to account for unseen words. This smoothing technique helps to mitigate the zero probability issue by ensuring that every possible word has a non-zero probability.

Perplexity Calculation

- The perplexity of the model was calculated on the test set for different nnn values. Perplexity is a measure of how well a probability distribution predicts a sample and is defined as the exponentiation of the average negative log probability of the words in the test set.

Summary of Analysis Results

The N-Gram language model was evaluated using perplexity calculations for three different values of n:

- **Unigram Model (n=1)**:
    - **Perplexity:** 30.4586
    - **Analysis:** The highest perplexity indicates a lack of contextual understanding, as the model predicts each word independently without considering preceding words.
- **Bigram Model (n=2)**:
    - **Perplexity:** 22.4159
    - **Analysis:** The bigram model showed significant improvement over the unigram model. By incorporating one preceding word, it captures some contextual relationships, resulting in better predictions and lower perplexity.
- **Trigram Model (n=3)**:
    - **Perplexity:** 26.7567

- **Analysis:** The trigram model's perplexity increased compared to the bigram model. This suggests potential overfitting, where the model struggles with many unseen three-word sequences in the test set. While it incorporates more context, the limited training data may hinder its performance.

Overall Insights

The results demonstrate that while increasing n generally enhances model performance, it can also lead to challenges such as overfitting and higher perplexity when the training data is insufficient. The bigram model yielded the best performance in this evaluation, highlighting the importance of balancing context and data quality in N-Gram language models.