Project Report: N-Gram Language Models

Jayani Tripathi

Introduction

This project aimed to develop a sentence autocomplete feature based on N-gram language models, focusing on predicting the next character in a sequence. The model utilizes a training document to create frequency tables for various N-grams (unigrams, bigrams, trigrams, etc.), calculates conditional probabilities, and uses them to predict the most likely next character.

Objective

The goal of the project was to understand and apply the theoretical principles behind N-gram language models and use them to implement a working autocomplete system. The project involved:

- Building frequency tables from the training document
- Calculating probabilities for character sequences
- Implementing a function to predict the next character in a sequence

Approach

1. **Data Preprocessing**
   The first step was to preprocess the training document by cleaning and organizing the text to be used for the model. I used a small corpus to simplify the task, ensuring the characters of interest (e.g., 'a', 'b', 'c', 'd') were part of the vocabulary.

2. **Creating Frequency Tables**
   I implemented a function to generate frequency tables for n-grams of varying sizes (up to N). For each table, I recorded how often a character (or sequence of characters) occurred in the document. This process involved:
   - **Unigram Frequencies**: The frequencies of individual characters.
   - **Bigram Frequencies**: The frequencies of two-character sequences.
   - **Trigram Frequencies**: The frequencies of three-character sequences.
   - These frequency tables helped in understanding the dependencies between characters and building a probabilistic model.
3. **Calculating Conditional Probabilities**
   The core of the model involved computing the probability of a character given the previous n characters. I applied the chain rule of probability to model the dependencies. For example, in a trigram model, the probability of the next character depends on the previous two characters. The conditional probability for each character sequence was computed using the formula:
   $P(X4 | X1,X2,X3)=f(X1,X2,X3,X4)f(X1,X2,X3)P(X_4 \mid X_1, X_2, X_3) = \frac{f(X_1, X_2, X_3, X_4)}{f(X_1, X_2, X_3)}$
   This approach allowed for predicting the next character in the sequence by using the

precomputed frequency tables.

4. **Predicting the Next Character**
   Using the frequency tables and calculated probabilities, I implemented a function to predict the most likely next character in a given sequence. The function iterated over the vocabulary and computed the probability of each potential next character, returning the one with the highest probability.

Conclusion

In this project, I successfully implemented an N-gram language model for sentence autocomplete. The approach involved constructing frequency tables, calculating conditional probabilities, and predicting the next character in a sequence. As the n-value increased, both time and memory constraints became limiting factors, making it challenging to scale for larger corpora.

Future improvements could include optimizing the table generation process and implementing smoothing techniques to handle rare n-grams. Additionally, using a larger, more diverse corpus would enhance the model's accuracy in real-world applications.

Note: Please feel free to email me for the results and code of this project!