DSA4020A: Natural Language Processing
Instructor: Dr. Edward Ombui

Lab Assignment 1: Exploring Language Modeling with N-Gram Sizes and Smoothing Techniques
Due Date: 23rd Sept, 2024
Marks: 40marks

Objective

The goal of this lab assignment is to understand how different n-gram sizes and smoothing techniques affect the performance of language models. You will implement n-gram models, apply various smoothing techniques, and evaluate their performance using a sample text dataset i.e. Movie Review Dataset (available on Kaggle. Click HERE)

Background

Language modeling is a crucial task in natural language processing (NLP) that involves predicting the next word in a sequence given the previous words. N-gram models are a type of statistical language model that uses the probabilities of sequences of n words to make predictions. Smoothing techniques are employed to handle the problem of zero probabilities for unseen n-grams in the training data.

Materials Needed

- Python 3.x
- Libraries: NLTK, NumPy, Pandas, Matplotlib (for visualization)
- A text dataset (e.g., a large text corpus like the Movie Review Dataset available on Kaggle. Click HERE)

Assignment Steps

Step 1: Data Preparation

1. Select the Movie Review Dataset: It is rich in vocabulary and structure. Preprocess the Data:
    - Tokenize the text into sentences and words.

- Convert all text to lowercase.
- Remove punctuation and special characters.

2.

Step 2: Implement N-Gram Models

1. Create N-Gram Models:
   - Implement functions to generate n-grams from the tokenized data for various values of n (e.g., 1, 2, 3, and 4).
   - Store the frequency counts of each n-gram in a dictionary.
2. Calculate Probabilities:
   - For each n-gram, calculate the probability using the formula:
   - $P(wn \mid wn-1,\ldots,w1)=C(w1,w2,\ldots,wn)C(w1,w2,\ldots,wn-1)$

Step 3: Apply Smoothing Techniques

Implement and test the following smoothing techniques:

1. Laplace Smoothing (Add-One Smoothing):
   - Modify your probability calculations to include a count for unseen n-grams.
2. Good-Turing Discounting:
   - Adjust probabilities based on the frequency of observed n-grams.
3. Kneser-Ney Smoothing:
   - Implement this more advanced technique that considers lower-order n-grams for better probability estimation.

Step 4: Evaluate Model Performance

1. Perplexity Calculation:
   - Define a function to calculate perplexity for your models on a held-out test set:
   - $Perplexity=P(w1,w2,\ldots,wN)-1/N$
   - Compare perplexity across different n-gram sizes and smoothing techniques.
2. Cross-Validation:
   - Use k-fold cross-validation to ensure robust evaluation of model performance.

3.

Step 5: Visualization and Analysis

1. Plot Results:
   - Create plots comparing perplexity for different n-gram sizes and smoothing techniques.
   - Use Matplotlib to visualize how changes in n and smoothing affect model performance.
2. Analyze Findings:
   - Write a brief report summarizing your findings on how n-gram size and smoothing techniques impact language model performance.
   - Discuss any trade-offs observed between model complexity and accuracy.

Deliverables

1. A Python script or Jupyter Notebook containing your code implementation.
2. A report summarizing your methodology, findings, and visualizations.
3. Any additional insights or observations you made during the assignment.

Submission Guidelines

Please submit your completed assignment by 23rd Sept, 2024.
Ensure that your code is well-commented and organized for ease of understanding.

By completing this lab assignment, you will gain hands-on experience with language modeling concepts and develop an understanding of how different parameters influence model performance in NLP tasks. Happy coding!