NLP Corpus Analysis Tools

Project Overview

This project provides two Python scripts for analyzing text corpora and performing various natural language processing (NLP) tasks. The scripts apply n-gram models to analyze word frequencies, calculate probabilities, implement smoothing techniques, and evaluate language models through perplexity.

Features

- Unigram and bigram frequency analysis
- Maximum Likelihood Estimation (MLE) for probability calculation
- Add-one (Laplace) smoothing implementation
- Perplexity calculation for language model evaluation
- Random sentence generation based on n-gram models
- Interactive menu-based interface

Requirements

- Python 3.10 or newer
- No external dependencies (uses only standard library modules):
    - argparse for command-line argument parsing
    - collections for efficient counting
    - random for sampling operations
    - math for logarithmic calculations

File Structure

```
project_root/

├── ForAnalysis.py      # Script with additional option to print top 10 n-grams

├── Updated_T1-T4.py    # Base script for corpus analysis and NLP tasks

├── TheStory.txt        # Text corpus for analysis

└── README.txt          # This documentation file
```

Installation

Simply download the files and ensure Python 3.10+ is installed on your system.

Usage Instructions

Basic Command Syntax

```
python3 Updated_T1-T4.py corpus_path

python3 ForAnalysis.py corpus_path
```

Where corpus_path is the path to your text corpus file (e.g., TheStory.txt).

Interactive Options

After running either script, an interactive menu will be displayed with the following options:

For Updated_T1-T4.py:

1: Reprint probabilities

2: Generate sentence

3: Apply Add-1 smoothing

4: Calculate Perplexity on test data

5: Quit

For ForAnalysis.py:

1: Reprint probabilities

2: Generate sentence

3: Apply Add-1 smoothing

4: Calculate Perplexity on test data

5: Quit

6: Print top 10 uni|bi grams

Example Usage

1. Run the basic analysis script:
   ```
   python3 Updated_T1-T4.py TheStory.txt
   ```
2. Run the extended analysis script (includes top 10 n-grams option):
   ```
   python3 ForAnalysis.py TheStory.txt
   ```
3. For perplexity calculation (Option 4), you'll be prompted to enter the path to a test corpus file.

Menu Options Explained

Option 1: Reprint probabilities

Displays the first five unigram and bigram probabilities calculated from the corpus.

Option 2: Generate sentence

Creates a random 15-word sentence using the calculated n-gram model. If smoothing has been applied, the sentence will use the smoothed probabilities.

Option 3: Apply Add-1 smoothing

Applies Laplace (add-one) smoothing to the probability distributions to handle zero-probability n-grams.

Option 4: Calculate Perplexity on test data

Prompts for a test corpus path and calculates the perplexity score, which evaluates how well the n-gram model predicts the test data.

Option 5: Quit

Exits the program.

Option 6: Print top 10 uni|bi grams (ForAnalysis.py only)

Displays the top 10 most probable unigrams and bigrams in the corpus.

Technical Details

Tokenization

The scripts use a custom tokenizer that identifies words as sequences of alphanumeric characters and underscores.

Probability Calculation

- Unigram probabilities are calculated as: $P(w) = count(w) / N$ (where N is the total number of tokens)
- Bigram probabilities are calculated as: $P(w2|w1) = count(w1,w2) / count(w1)$

Add-One Smoothing

Implements Laplace smoothing where:

- Smoothed unigram probability: $P(w) = (count(w) + 1) / (N + V)$
- Smoothed bigram probability: $P(w2|w1) = (count(w1,w2) + 1) / (count(w1) + V)$ Where V is the vocabulary size.

Perplexity

Calculated as $2^{(-L)}$, where L is the average log (base 2) probability of the test corpus according to the language model.

Sentence Generation

Uses the calculated probability distributions to randomly generate sentences, with words selected proportionally to their likelihood in the model.