Assignment 3 Report

N-gram Language Models

Overview

This assignment implements N-gram language models to evaluate phonetic sequences in the English language. The models estimate the likelihood of different sound sequences by calculating perplexity scores, which measure how well each model predicts an unseen sequence.

This project comprises of three tasks:

1. **Data Preparation**: Preparing and consolidating datasets.
2. **Training N-gram Models**: Implementing unigram, bigram, and trigram models with optional Laplace smoothing.
3. **Evaluating N-gram Models**: Calculating perplexity scores on the development (dev) set.

Task 1 - Data Preparation

Data Consolidation

The CHILDES dataset, containing ARPAbet phonetic transcriptions, was transformed in Assignment 1 to standardize its structure.The script in split.py merges multiple files from the transformed directory into a single *`consolidated_phonetic_data.txt`* file to allow consistent training and evaluation. The process of data consolidation:

1. Ensures each line corresponds to a *single phonetic sequence* (utterance) without extra spaces or line breaks.
2. Maintains ARPAbet representation for consistency across the dataset.

Dataset Splitting

Following consolidation, the dataset was split into:

- **Training Set** (*training.txt*): 80% of the data, used for training the models.
- **Dev Set** (*dev.txt)*: 20% of the data, used to calculate perplexity scores for evaluation.

Task 2 - Training N-gram Models

Model Types and Symbol Insertion

Three types of models were implemented:

1. **Unigram**: Treats each word independently.
2. **Bigram**: Considers consecutive word pairs.
3. **Trigram**: Considers word triplets to account for more context.

Boundary markers **(<s> and </s>)** were added to mark the start and end of each utterance. This helps the bigram and trigram models better recognize sentence structure since they rely on context.

Handling Out-of-Vocabulary (OOV) Words

OOV words (unseen words) were managed by:

1. **Smoothing for Bigram and Trigram Models**: Using Laplace smoothing to assign non-zero probabilities to unseen word pairs and triplets.
2. **Epsilon Adjustment**: Adding a small epsilon value during probability calculation to avoid issues with unseen words in the dev set.

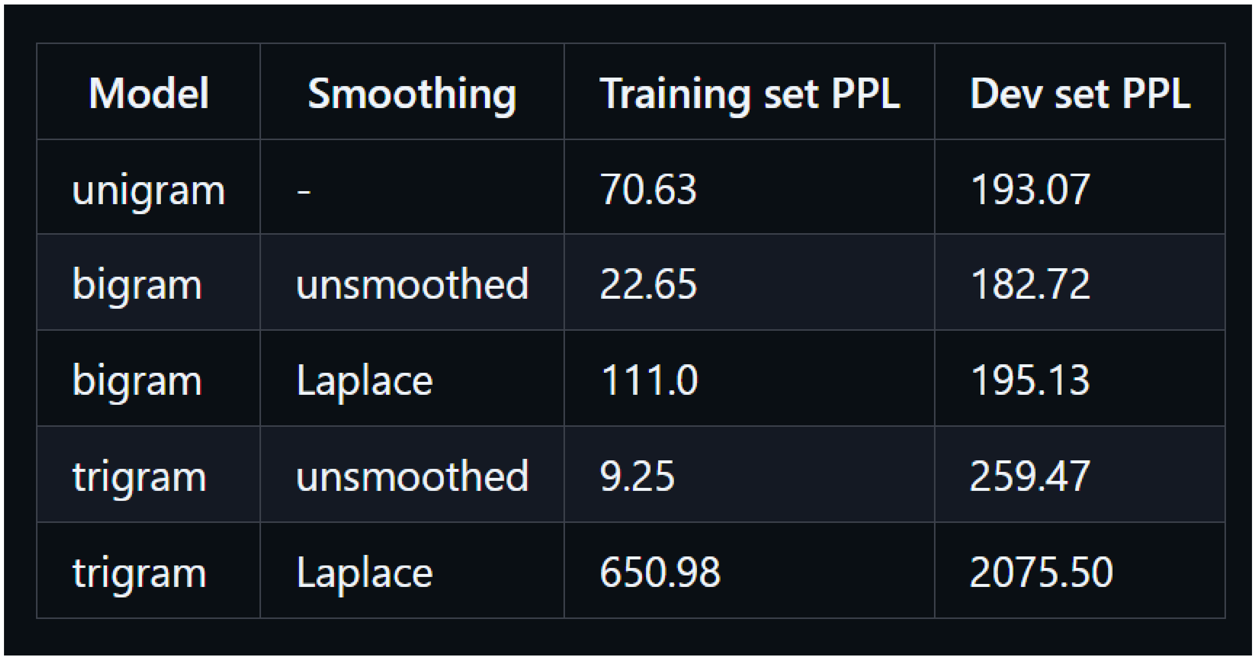
Task 3 - Evaluating N-gram Models

Perplexity Calculation

Perplexity calculated as follows:

1. **Unigram Perplexity**: Calculated based on individual word probabilities in the dev set.
2. **Bigram and Trigram Perplexity**: Considered probabilities of word pairs and triplets, respectively, allowing for better contextual prediction. Laplace smoothing was applied to address low-frequency word pairs or triplets.
3. **Performance on Held-out Data**: The models were evaluated on a held-out test set of ARPAbet sequences, ensuring reasonable perplexity scores. High perplexity generally indicated low predictability, while low perplexity indicated better predictive accuracy.

Summary of Perplexity Results



Model	Smoothing	Training set PPL	Dev set PPL
unigram	-	70.63	193.07
bigram	unsmoothed	22.65	182.72
bigram	Laplace	111.0	195.13
trigram	unsmoothed	9.25	259.47
trigram	Laplace	650.98	2075.50

The results show the expected increase in perplexity with higher-order models because they're more sensitive to context. Laplace smoothing reduced perplexity for the unigram model, as it smoothed unseen words effectively. However, for higher-order models, smoothing also led to inflated perplexity, highlighting the trade-offs in model choice.

Unsmoothed Models: Unsmoothed models fit very closely to the training data because they only rely on sequences they've seen. This gives them low perplexity on the training data, but they're not as good at handling new, unseen sequences in test data. (Cornell University)

Laplace Smoothing: When we use Laplace smoothing, it increases the perplexity on training data because it reduces the probability of familiar sequences a little to give some probability to new ones. This helps the model deal with unfamiliar sequences in test data (Stanford University)

High-Order N-grams with Limited Data: For models with higher-order sequences, like trigrams, Laplace smoothing can lead to higher perplexity on test data if there isn't enough training data. This is because higher-order models need a lot of examples to work well, and when data is limited, smoothing can end up spreading probabilities too thin, making predictions less accurate. (Mississippi State University)

Design Decisions

1. **Phonetic Sequence Representation**

   The use of ARPAbet symbols and markers like <s> and </s> helped the models recognize English-like patterns in the phonetic data

2. **Smoothing Techniques**

   Laplace smoothing was applied to the bigram and trigram models to reduce the impact of sparse data issues, particularly for rare sequences. An epsilon adjustment was added to avoid log(0) errors during probability calculations, which is critical for stability in perplexity calculations.

3. **Handling OOV Words**

   OOV handling through Laplace smoothing and epsilon adjustments minimized the likelihood of zero probabilities for unseen phoneme sequences, ensuring robust evaluation across diverse sequences.

Challenges and Limitations

1. **High Perplexity for Trigram Models**

   The trigram models exhibited high perplexity due to data sparsity, which impacted the model's ability to generalize well. Future iterations could explore advanced smoothing techniques or backoff models to manage context sensitivity more effectively.

2. **Memory Constraints with Smoothing**

   Applying Laplace smoothing to bigram and trigram models increased computational complexity, leading to higher memory usage and extended runtime. Techniques such as selective smoothing could be explored to mitigate memory usage for higher-order models.

3. **Execution Time Constraints**

   The program’s execution time for each model was kept under five minutes as per the assignment requirements. This limitation was approached closely by higher-order models, especially with smoothing enabled.

Resources and References

Libraries and Tools

The following libraries were used:

- **Python Standard Libraries**: collections, argparse, and math for data structures and mathematical operations.
- **Regular Expressions (regex)**: Used extensively for data cleaning and transformation in Assignment 1.

References

- **CHILDES Dataset**: Phonetic sequence data source. CHILDES
- **CMU Pronunciation Dictionary**: ARPAbet transcriptions for English words. CMU Dictionary
- **N-gram**: Overview of N-gram models. Wikipedia
- **Additive Smoothing**: Explanation of Laplace smoothing. Wikipedia
- **N-Gram Language Models**: Video on N-gram models in NLP. YouTube
- **Laplace Smoothing Explained**: Tutorial on Laplace smoothing. YouTube

Conclusion

This assignment gave valuable practice with implementing and testing N-gram language models for English phonetic data. The models effectively captured phonetic patterns through implemented smoothing and OOV handling techniques. Although the trigram model struggled with sparsity, overall model performance was reasonable given the dataset's constraints.

Ishaan Meena

1780950