N-Gram Language Modeling and Evaluation

Assignment #2

Author: Prakhar P. Tiwari
University of Maryland
Graduate Student - MS in Machine Learning

1. Overview

This project implements and evaluates statistical N-Gram Language Models using the Penn Treebank (PTB) dataset. The primary objective is to predict the likelihood of word sequences and assess model performance using perplexity. Language modeling is a cornerstone of natural language processing tasks such as speech recognition, text generation, and machine translation. By examining multiple N-gram orders (1–4), smoothing methods, and backoff/interpolation techniques, this work demonstrates how data sparsity and context size affect prediction accuracy.

2. Key Features and Implementation

The project includes several core components implemented in Python with modular scripts for preprocessing, model training, evaluation, and generation. Each module interacts seamlessly to support end-to-end language modeling.

2.1 Model Types

- **Maximum Likelihood Estimation (MLE)** — Computes probabilities directly from observed counts.
- **Add-1 (Laplace) Smoothing** — Addresses data sparsity by adding one to all counts, ensuring nonzero probabilities.
- **Linear Interpolation** — Combines unigram, bigram, trigram, and 4-gram models using weighted λ_1–λ_4 parameters, tuned on the validation set via randomized parallel search.
- **Stupid Backoff** — A simpler yet effective alternative where lower-order models are used recursively with a backoff factor α when higher-order counts are unavailable.

2.2 Technical Highlights

- Efficient text preprocessing and tokenization with start and end tokens.
- Parallelized tuning for λ and α parameters using multithreading.
- Checkpointing to skip retraining on subsequent runs.
- Unified text generation framework supporting all model types.
- Automated logging of model performance to `results/summary.csv`.
- UTF-8 safe I/O and dynamic CSV logging for reproducible experiments.

3. Experimental Analysis

3.1 Evaluation Metrics

Model performance is evaluated using **perplexity**, which measures how well a language model predicts a sample. Lower perplexity indicates better performance.

3.2 Results Summary

The table below presents the observed test-set perplexities across all models after tuning:

```
Model,Setting,Test_Perplexity
1-gram,MLE,536.1817
2-gram,MLE,inf
3-gram,MLE,inf
4-gram,MLE,inf
1-gram,Add-1,536.6617
2-gram,Add-1,463.1116
3-gram,Add-1,1966.4405
4-gram,Add-1,4348.2567
Interpolation,"Tuned λ₁-λ₄ (0.292, 0.526, 0.182, 0.0)",129.1554
Stupid Backoff,α=0.6,128.8281
```

As expected, higher-order models achieve lower perplexities. Interpolation and Stupid Backoff provide significant improvements over unsmoothed models by effectively balancing high-order precision and low-order generalization.

3.3 Generated Text Examples

To qualitatively assess model fluency, sentences were generated using the Interpolation and Stupid Backoff models.

1. Fluency and Grammar

- **Linear Interpolation:**
  Sentences are often fragmented, filled with placeholder tokens like <unk> and stray symbols (>, N).

- Example:

“embassy president the new contract rose N a loss makers applying <unk> looks like an extended in”

→ The grammar is broken; subjects and verbs don't align.

- **Stupid Backoff:**
  Also contains <unk> tokens, but generally maintains more coherent syntactic patterns.

- Example:

"bill clinton of arkansas announced that this was not the other N <unk> such as < > funds"

→ Reads closer to a proper sentence, resembling real news language.

✅ **Winner:** *Stupid Backoff* (fewer grammatical disfluencies)

🔹 **2. Lexical Diversity**

- **Linear Interpolation:** Uses repetitive tokens and many "N" placeholders, suggesting that rare words dominated probabilities in interpolation weights.
- **Stupid Backoff:** Exhibits wider lexical variety ("coalition," "shareholders," "contractor," "utilities"), giving a more natural distribution of vocabulary.

✅ **Winner:** *Stupid Backoff*

🔹 **3. Contextual Coherence**

- **Linear Interpolation:** Appears confused across domains (finance, politics, and weather in the same lines). The <unk> frequency and disconnected phrases show it over-relied on lower-order n-grams.
- **Stupid Backoff:** Maintains domain coherence—many sentences resemble *Wall Street Journal*-like corporate or financial news content, consistent with the PTB dataset.

✅ **Winner:** *Stupid Backoff*

🔹 **4. Overall Human-Likeness**

Criterion	Linear Interpolation	Stupid Backoff
Fluency	❌ Fragmented	✅ Mostly readable
Grammar	❌ Disordered	✅ More consistent
Vocabulary	❌ Repetitive	✅ Diverse
Context Coherence	❌ Mixed domains	✅ Consistent domain

✅ **Overall Verdict:**
The Stupid Backoff–generated text is more human-like — it produces smoother, domain-consistent sentences with clearer structure, while Linear Interpolation outputs appear mechanically blended and more error-prone.

4. Conclusion

This project highlights the strengths and trade-offs among various N-gram models. While higher-order models capture richer context, they also suffer from data sparsity, which is mitigated through smoothing, interpolation, and backoff techniques. Linear Interpolation and Stupid Backoff outperform simple Add-1 smoothing, achieving the lowest perplexity and most natural generated text. The modular design and parallelized tuning framework make the system scalable and adaptable for future extensions such as Kneser-Ney smoothing or neural language models.