N-Gram Language Modeling and Evaluation

Anirudh Krishna
University of Maryland, College Park

Abstract

This report implements and evaluates multiple N-gram language models using the Penn Treebank (PTB) dataset. The goal is to understand how model order, smoothing, and backoff strategies affect perplexity and text generation. Four approaches were implemented: Maximum Likelihood Estimation (MLE), Add-1 Smoothing (Laplace), Linear Interpolation, and Stupid Backoff. All models were trained and evaluated according to assignment specifications using train, validation, and test splits. Perplexity (PP) was used for quantitative evaluation, and the best model (Linear Interpolation) was used for qualitative text generation.

1 Dataset and Preprocessing

The Penn Treebank (PTB) dataset from Kaggle was used, with the following file paths:

- **Train:** /content/ptb.train.txt
- **Validation:** /content/ptb.valid.txt
- **Test:** /content/ptb.test.txt

Each line represents a pre-tokenized sentence. For each n-gram model, $(n-1)$ start tokens $\langle s \rangle$ and one end token $\langle /s \rangle$ were appended. Unknown words were replaced by the token `<unk>`. No punctuation or case normalization was applied to preserve the original corpus formatting.

2 Modeling and Evaluation

All models were implemented in Python using `Counter` for frequency counts. Perplexity was computed over the test corpus, and any zero-probability n-grams under MLE were reported as infinite (`INF`).

2.1 3.1 Maximum Likelihood Estimation (MLE)

The MLE model estimates probabilities as relative frequencies. Per assignment requirements, unseen test n-grams yield zero probability and thus infinite perplexity.

Model	**Test Perplexity**
Unigram (N=1)	639.30
Bigram (N=2)	∞
Trigram (N=3)	∞
4-gram (N=4)	∞

2.2 3.2 Smoothing and Backoff Strategies (Trigram)

To mitigate sparsity, smoothing and backoff strategies were applied to trigram models.

2.2.1 Add-1 (Laplace) Smoothing

Add-1 smoothing assigns one pseudo-count to all trigrams, ensuring non-zero probabilities. While this reduces infinite perplexity, it over-smooths frequent events.

Test Perplexity (Add-1): 3307.92

2.2.2 Linear Interpolation

Linear Interpolation combines unigram, bigram, and trigram probabilities as:

$$P(w_i|w_{i-2}, w_{i-1}) = \lambda_1 P(w_i) + \lambda_2 P(w_i|w_{i-1}) + \lambda_3 P(w_i|w_{i-2}, w_{i-1}) \tag{1}$$

subject to $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

Validation Tuning: Three λ combinations were tested on the validation set:

($\lambda_1, \lambda_2, \lambda_3$)	**Dev Perplexity**
(0.2, 0.3, 0.5)	224.51
(0.1, 0.3, 0.6)	256.53
(0.3, 0.3, 0.4)	211.61

The best combination was $(0.3, 0.3, 0.4)$, achieving:

Validation PP = 211.61, Test PP = 195.75

2.2.3 Stupid Backoff

The Stupid Backoff model recursively falls back to shorter contexts with a fixed discount factor $\alpha = 0.4$. It is not a normalized probability model, but a pseudo-perplexity was computed for comparison.

Stupid Backoff Test Pseudo-Perplexity: 185.67

3 Analysis and Discussion

3.1 Preprocessing and Vocabulary Decisions

The PTB corpus was treated as pre-tokenized. Each sentence was bounded with start and end tokens. The vocabulary was derived from training data, with all unseen words mapped to `<unk>`.

3.2 Impact of N-gram Order

As expected, MLE models for $N \geq 2$ encountered unseen n-grams in the test data, leading to ∞ perplexities. This illustrates how higher-order models face severe data sparsity, validating the need for smoothing methods.

3.3 Comparison of Smoothing and Backoff Strategies

Add-1 smoothing resolved infinite perplexities but inflated overall PP due to uniform redistribution of probability mass. Linear Interpolation achieved the lowest true perplexity (195.75) by optimally weighting information from multiple contexts. Stupid Backoff performed competitively (185.67 pseudo-PP) despite its simplicity, demonstrating the value of discounted backoff strategies.

3.4 Qualitative Analysis (Generated Text)

The best model (Linear Interpolation) was used to generate five distinct sentences:

```
1. the risk of soviet medical center of this year 's <unk> and owed that busin
2. a painting showed when any of that has consistently rejected states west 's
3. climate set up soon to tell them <unk> a la jolla bank to worry <unk> will
4. japan 's no. N in federal court let N holding in rockefeller buried beneath
5. the <unk> stock on pencil co japan in appears capital a drexel analyst for
```

Discussion: The generated sentences demonstrate realistic local syntactic structure but limited semantic coherence. Frequent use of `<unk>` reflects vocabulary constraints. Overall, the model generates linguistically plausible yet semantically shallow text, consistent with the limitations of statistical n-gram models.

4 Conclusion

The MLE baseline suffered from data sparsity, producing infinite perplexities for higher-order models. Smoothing and interpolation substantially improved generalization. Among all models, Linear Interpolation achieved the best perplexity–fluency trade-off, while Stupid Backoff showed strong heuristic performance. This confirms that combining contextual levels is more effective than using a single n-gram order.