Homework 2: N-gram Language Model Report

1. Introduction

The goal of this assignment was to implement, train, and evaluate a series of N-gram language models. We explored the performance of Maximum Likelihood Estimation (MLE) models of varying N-gram orders (N=1, 2, 3, 4) and implemented two different smoothing strategies (Add-1 and Linear Interpolation) for a Trigram model to handle data sparsity. All models were evaluated using the perplexity (PP) metric on the Penn Treebank (PTB) test dataset.

2. Methodology

2.1. Preprocessing

The raw PTB data (`ptb.train.txt` , `ptb.valid.txt` , `ptb.test.txt`) was processed as follows:

1. **Vocabulary Creation:** A vocabulary was built from the training data (`ptb.train.txt`). Any word appearing more than once was included. All other words (including single-occurrence words in the training set) were mapped to a single unknown token, `<unk>` . The special tokens `<s>` , `</s>` , and `<unk>` were explicitly added to the vocabulary. Our final vocabulary size, including special tokens, was **9,971 words**.
2. **Sentence Boundaries:** For all models, each sentence in every dataset (train, valid, test) was padded with `(n-1)` start tokens (`<s>`) and one end token (`</s>`).

2.2. Language Models

- **MLE Models (N=1, 2, 3, 4):** We implemented standard MLE N-gram models. The probability of an n-gram is calculated as the count of the n-gram divided by the count of its `(n-1)` -gram prefix.

  ```
  P(w_i | w_i-n+1...w_i-1) = C(w_i-n+1...w_i) / C(w_i-n+1...w_i-1)
  ```

- **Trigram with Add-1 (Laplace) Smoothing:** This model adds 1 to every n-gram count, effectively "donating" probability mass from seen n-grams to unseen ones. `V` is the size of the vocabulary.

  ```
  P(w_i | w_i-2, w_i-1) = (C(w_i-2, w_i-1, w_i) + 1) / (C(w_i-2, w_i-1)
  + V)
  ```

- **Trigram with Linear Interpolation:** This model computes the probability by combining the MLE probabilities of the trigram, bigram, and unigram models, weighted by lambda (λ) parameters.

```
P_interp(w_i | w_i-2, w_i-1) = λ_3 * P_MLE(w_i | w_i-2, w_i-1) + λ_2 * P_MLE(w_i | w_i-1) + λ_1 * P_MLE(w_i)
```

2.3. Hyperparameter Tuning

As required, the lambda weights (λ_1, λ_2, λ_3) for the Linear Interpolation model were tuned using the `ptb.valid.txt` validation set. We performed a simple grid search, testing all combinations of weights from 0.0 to 1.0 in steps of 0.1 (where $\lambda_1 + \lambda_2 + \lambda_3 = 1.0$).

The set of lambdas that yielded the lowest perplexity on the validation set was chosen. Our tuning process yielded a perplexity of **199.49** on the validation set. The selected weights were: **λ_1=0.3, λ_2=0.5, λ_3=0.2**.

This indicates that the model learned to rely most heavily on the bigram model (λ_2=0.5), then the unigram model (λ_1=0.3), and least on the sparse trigram model (λ_3=0.2).

2.4. Evaluation Metric

All models were evaluated using **Perplexity (PP)** on the `ptb.test.txt` dataset. Perplexity is the exponent of the cross-entropy. A lower perplexity score indicates a better model. `N` is the total number of tokens in the test set, including `</s>` tokens.

```
PP = 2 ^ (-1/N * Σ log₂(P(sentence)))
```

---

3. Results

The perplexity scores for all implemented models on the test set are as follows:

Model	Perplexity on Test Set
MLE Unigram (N=1)	inf
MLE Bigram (N=2)	inf
MLE Trigram (N=3)	inf
MLE 4-gram (N=4)	inf
Trigram + Add-1 Smoothing	3716.67
Trigram + Linear Interpolation	**161.58**

(Note: `inf` (infinity) is reported for MLE models as per the assignment guidelines, as they encountered an n-gram in the test set that was never seen in training, resulting in a zero probability).

---

4. Analysis & Discussion

4.1. Preprocessing

Our preprocessing strategy involved padding all sentences with `(n-1)` start tokens (`<s>`) and one end token (`</s>`). The vocabulary was built from words in the training set with a frequency greater than 1. This strategy captures the most common words while collapsing all rare words and out-of-vocabulary words (from the dev and test sets) into the `<unk>` token. This resulted in a vocabulary of 9,971 words. This is a crucial step; without it, the model would have no way to handle words it has never seen.

4.2. Impact of N-gram Order

The results table clearly shows the trade-off of the N-gram order.

- The **MLE models for N=1, 2, 3, and 4** all resulted in infinite perplexity. This is the expected outcome and perfectly illustrates the problem of **data sparsity**. As N increases, the number of possible n-grams explodes exponentially. It becomes statistically impossible for a training corpus (even one of 42,000 sentences) to contain every possible n-gram that might appear in a new (test) set.
- The moment the model encounters an unseen n-gram, its count is 0, its probability is 0, and the `log(0)` calculation makes the perplexity infinite. This shows that while higher-order N-grams (like N=3 or N=4) make a more reasonable Markov assumption (that a word depends on its recent history), they are unusable in their basic MLE form.

4.3. Comparison of Smoothing/Backoff Strategies

- **Why MLE Fails:** As noted in 4.2, all MLE models failed because they assign a probability of zero to any unseen event. This is a fatal flaw, as any real-world test set is guaranteed to contain n-grams not present in the training set.
- **Add-1 Smoothing:** This strategy "solves" the zero-probability problem, but its perplexity (**3716.67**) is exceptionally high. This is because the vocabulary size (`V` = 9,971) is very large. Add-1 smoothing subtracts a large amount of probability mass from the n-grams we *did* see and redistributes it evenly among the *billions* of possible unseen n-grams. This "over-smoothing" results in a terrible model.

- **Linear Interpolation:** This model was the clear winner with a perplexity of **161.58**. Instead of just "donating" probability like Add-1, it intelligently combines probabilities from different N-gram orders. If an unseen trigram is encountered, the model "backs off" to the more robust bigram and unigram probabilities. Because the weights (λ_1, λ_2, λ_3) were tuned on the validation set, the model learned the optimal balance for combining these estimators, resulting in the lowest and most reasonable perplexity.

4.4. Qualitative Analysis (Generated Text)

The following 5 sentences were generated using our best-performing model (Trigram with Linear Interpolation).

1. `but government bonds`
2. `but they were proposing will be <unk> quota that the death toll there might be reached historically low of the`
3. `commercial scale to right test winning streak while georgia-pacific 's access <unk> to acknowledge that similar tariff cuts in the`
4. `earlier this point <unk> in nature of rebound in september <unk> said in a minority member of things being of`
5. `already my based on the drug enforcement`

Discussion:
The generated text is surprisingly fluent in short bursts. Phrases like "but government bonds" and "already my based on the drug enforcement" are very common and well-formed. The model successfully generates realistic-looking financial/news-style sentences because the PTB corpus is heavily composed of such text.

However, the model lacks any true understanding or long-range coherence. The sentences are grammatically plausible but have no connection to each other or any deeper meaning. The model generates these sequences by starting with a context (e.g., `<s> <s>`) and sampling a word from the probability distribution it has learned. It then appends that word to the context and samples again. This "local" decision-making is why the 2-3 word sequences look good, but the overall sentences are often nonsensical.

5. Conclusion

This assignment successfully demonstrated the implementation of N-gram language models. We confirmed that simple MLE models are unusable for N>1 due to data sparsity, resulting in infinite perplexity. We also showed that a simple, un-principled technique like Add-1 smoothing performs very poorly on large vocabularies. Finally, a tuned Linear Interpolation model that "backs off" to lower-order models provides a much more robust and effective language model, as evidenced by its significantly lower and more realistic perplexity of **161.58**.