**Limitations of N-gram Models**

N-gram models, although simple and intuitive, have significant limitations:

1. Data Sparsity: N-gram models, especially bigrams and trigrams, have issues with data sparsity. This occurs because the model struggles to account for unseen word combinations. This leads to poor generalization.
2. Context Size: N-gram models only consider a fixed window of previous words, meaning they fail to capture long-range dependencies. For example, a trigram model can only account for two previous words, making it unsuitable for capturing more complex word patterns.
3. High Perplexity: As seen in the MLE results, the lack of smoothing results in infinite perplexity when the model encounters unseen n-grams in the test data.

**Model Performance and Results**

I trained both bigram and trigram models on the Brown corpus, evaluating them with and without smoothing. The results of the experiments are shown below:

1. MLE (No Smoothing):
   1. Bigram Model Perplexity: inf
   2. Trigram Model Perplexity: inf

The infinite perplexity is caused by the MLE model’s inability to handle unseen n-grams, leading to zero probabilities and division by zero during perplexity calculation. This is a limitation of MLE without any smoothing.

1. Laplace Smoothing:
   1. Bigram Model Perplexity: 11664.000
   2. Trigram Model Perplexity: 11664.000

Laplace smoothing successfully prevents infinite perplexities by assigning non-zero probabilities to unseen n-grams. Howeever, it overestimates the likelihood of rare n-grams, resulting in a high perplexity score. This indicates poor model performance.

1. Kneser-Ney Smoothing:
   1. Bigram Model Perplexity: 41.65
   2. Trigram Model Perplexity: 55.40

Kneser-Ney smoothing is designed to handle both frequent and rare word sequences more effectively. It reduced the perplexity compared to Laplace smoothing. This smoothing technique is considered more sophisticated as it not only discounts probabilities for frequent n-grams but also redistributes the mass to more meaningful n-grams, resulting in better performance.

**Generated Sentences**

I used the Kneser-Ney smoothed models to generate sample sentences:

* Bigram Sentence: "gary hammond , come may total of the boys clubs and mrs. edwin socola of"
* Trigram Sentence: "in one large oversimplification , it was their conviction that the couple are ray meredith"

The sentences have a lack of grammatical structure and meaning. This shows the limitations of n-gram models, especially when only using short contexts (bigrams or trigrams).

**Reflections and Future Improvements**

The results confirm the known limitations of n-gram models, particularly regarding data sparsity and handling unseen data. Smoothing techniques such as Kneser-Ney greatly improve performance, but there are still limitations in capturing long-range dependencies.

**Potential improvements**

1. Move to Neural Networks: More advanced models like Recurrent Neural Networks (RNNs) or Transformers can capture longer-range dependencies and complex patterns in language.
2. Backoff Models: Combining n-grams of varying sizes (e.g., unigram, bigram, trigram) in a hierarchical manner could help the model fall back to simpler representations when data is sparse.
3. Interpolated Smoothing: Exploring other smoothing methods such as interpolated smoothing could lead to better perplexity scores.

**Model Perplexity Snippet**

def calculate\_model\_perplexity(model, test\_sentences):

try:

perplexity = model.perplexity(test\_sentences)

return perplexity

except ZeroDivisionError:

return float('inf')