Contest 2 Experiment Report

ฐิติรัตน์ หมื่นนุช

Data

The data in the training set is drawn from https://huggingface.co/datasets/gigaword. Prior to training, the text was preprocessed by being lower-cased and tokenized.

The development set provided for evaluation contains 94,825 rows with 3 columns: the 'context' column, the 'first letter' column, and the 'prediction' column. The 'first letter' column provides the initial letter of the word to be predicted for each context, while the 'prediction' column contains the actual word that is to be generated. This development set serves as the benchmark for evaluating the accuracy of each model.

Finally, the test set consists of 94,826 rows with 2 columns: the 'context' column and the 'first letter' column. This set is used to generate predictions using the trained language generation models, with the 'first letter' column indicating the initial letter of the predicted word. The test set allows for the independent evaluation of the models' performance and generalizability to unseen data.

Model

I implemented two different models to predict the next word in a given context, including a trigram model as a baseline model, and kenlm (5-gram model) as I hypothesized that the 5-gram model would be capable of capturing more complex and nuanced language patterns and generate better results.

As for the trigram model, a counter dictionary is used to count the number of occurrences of each trigram in the training data. The model is trained in small batches, with a batch size of 2048, to accommodate the large size of the training set. Once the model has been trained, the probability of each word is computed based on its frequency in the training data, and the model is then used to predict the next word in a given context by selecting the word with the highest probability.

For kenlm (pre-trained 5-gram model), the next word in a given context is generated by looping over each word in the model's vocabulary and selecting the word with the highest probability.

Results

After using the model to generate the word for each context on the development set using both the trigram model and the pre-trained kenlm (5-gram model), a comparison was made between the generated words and the actual words provided.

To my surprise, the results indicated that both models yielded comparable accuracy rates, with the trigram model demonstrating an insignificantly higher level of performance.

Upon evaluation of the results produced by the trigram model, it was observed that some outputs yielded a NaN value. This was due to the absence of certain trigrams in the counter dictionary that had been created from the training data. In contrast, the pre-trained kenlm (5-gram model) was able to address this issue through the implementation of a backoff technique.

Given the limitations and strengths of each model, I hypothesized that combining the trigram and kenlm (5-gram) models using a hybrid approach would yield improved accuracy over either model alone. Specifically, the language generation results from the trigram model, which were found to outperform the pre-trained 5-gram model in terms of accuracy, would be used in cases where they did not yield a NaN value. In cases where the trigram model produced a NaN value, the results from the pre-trained 5-gram model would be used instead.

The resulting hybrid model yielded a higher overall accuracy rate when compared to the individual models alone as hypothesized.

	Accuracy score
Trigram model	0.547
Kenlm (5-gram model)	0.540
Combined result from both model	0.598

Conclusion

Based on the experimental results, the hybrid approach proved to be the most effective in generating accurate predictions. As such, for the test_set_pred.txt file submitted via Gradescope, I utilized both the trigram model and kenlm (5-gram model) to generate predictions for the test set and then combined the results using the hybrid approach described earlier.