NEURAL NETWORKS
HOMEWORK 2: RECURRENT NEURAL NETWORKS (RNNS)

Cătălin-Alexandru Rîpanu IA1-A
Faculty of Automatic Control and Computers
National University of Science and Technology POLITEHNICA Bucharest
E-mail:catalin.ripanu@upb.ro
Lecturer: Alexandru Sorici

30th November, 2024

ABSTRACT

This report provides an in-depth exploration of recurrent neural networks (RNNs) in predictive and translation tasks. It covers essential RNN concepts, including Teacher Forcing, Warm Start, Unroll Length, and applications in Natural Language Processing. The document is structured around two primary tasks: (1) implementing and analyzing a basic RNN using a sine wave dataset, and (2) developing a Sequence-to-Sequence model for translation tasks with Long Short-Term Memory (LSTM) networks. Key insights into the challenges and performance characteristics of these architectures are also discussed.

1 Task 1: Vanilla RNN on a Sine Wave Dataset

This task focused on training a simple RNN to model a sine wave dataset. Detailed implementation and responses to task-specific questions are available in the accompanying Jupyter Notebook. Relevant observations include:

- Without `teacher_forcing` and `warm_start`, the model struggles to generalize due to instability and lack of context, leading to underfitting.
- Overusing `teacher_forcing` can cause the model to over-rely on ground truth, impeding its ability to generalize to unseen data.
- Results may vary between training runs due to random sequence selection at initialization, which can lead the model to converge to different local minima.

To avoid truncation of output in Visual Studio Code, the `reporting_interval` was adjusted to 350.

2 Task 2: Sequence-to-Sequence Model for Translation Tasks

This task involved designing and testing Encoder-Decoder architectures based on LSTM and GRU models. The implementation included experimenting with various configurations to optimize performance within computational constraints (25 training epochs). Important aspects:

2.1 Objective 1: Basic Encoder-Decoder Architecture

The baseline architecture consisted of:

- **Encoder:**
  - Embedding layer with dropout
  - Single-layer LSTM
- **Decoder:**
  - Embedding layer with dropout
  - Single-layer LSTM
  - Fully connected layer for output generation

2.2 Objective 2: Bidirectional Encoder

In this setup, the Encoder was extended with a Bidirectional LSTM to better capture input semantics. However, the improvement in validation loss was marginal, indicating diminishing returns for this enhancement under limited training resources.

2.3 Experiment Results

Quantitative results for BLEU scores under varying configurations are summarized in the tables below.

Experiment	Emb Dim	Hid Dim	Batch Size	Teacher Forcing	Emb Dropout	BLEU Score
1	128	128	128	0	0.5	0.167
2	256	128	128	0	0.5	0.196
3	512	128	128	0	0.5	0.153

Table 1: Varied Embedding Dimensions

The increase of the Embedded Dimension results in a decrease in the model's performance due to the higher difficulty of the input sequence. It can be concluded that the Embedded Dimension of 256 likely provided better representation capacity as it allows the model to capture more nuanced relationships between words. The drop at 512 could be caused by potential overfitting due to too many parameters or because the model is too complex for the amount of training data. Larger Embedding Dimensions need more regularization (higher dropout).

Experiment	Emb Dim	Hid Dim	Batch Size	Teacher Forcing	Emb Dropout	BLEU Score
4	128	128	128	0	0.5	0.221
5	128	256	128	0	0.5	0.183
6	128	512	128	0	0.5	0.147

Table 2: Varied Hidden Dimensions

In experiment 6, starting with epoch 22, a numerical instability was produced, most probably because of exploding gradients, as the learning rate wasn't scaled based on the Hidden Dimension.

Experiment	Emb Dim	Hid Dim	Batch Size	Teacher Forcing	Emb Dropout	BLEU Score
7	128	128	128	0	0.5	0.167
8	128	128	256	0	0.5	0.094

Table 3: Varied Batch Sizes

Increasing the batch size from 128 to 256 leads to a significant drop in performance. This could be due to less frequent weight updates, reducing the model's ability to generalize efficiently during training.

Experiment	Emb Dim	Hid Dim	Batch Size	Teacher Forcing	Emb Dropout	BLEU Score
9	128	128	128	0	0.5	0.150
10	128	128	128	0.5	0.5	0.169
11	128	128	128	1	0.5	0.105

Table 4: Varied Teacher Forcing Probabilities

Introducing teacher forcing improves performance slightly, as seen in Experiment 10. However, excessive teacher forcing (Experiment 11) results in a significant drop in BLEU scores, likely due to the model becoming overly dependent on ground truth and struggling with real-world sequences.

Experiment	Emb Dim	Hid Dim	Batch Size	Teacher Forcing	Emb Dropout	BLEU Score
12	128	128	128	0	0	0.246
13	128	128	128	0	0.5	0.216
14	128	128	128	0	1	0.092

Table 5: Varied Embedding Dropout Probabilities

Dropout helps regularize the model, as evident in Experiments 12 and 13. However, excessive dropout (Experiment 14) degrades performance significantly, indicating that some information crucial for learning is lost when the dropout is too high.

The last experiment is the one using a Bidirectional Encoder, with the configs listed below:

Experiment Analysis Report

Best final_bleu:
Value: 0.3122
Configuration:

- emb_dim: 128
- hid_dim: 128
- batch_size: 1152 (128 * 3 * 3, used 3 GPUs)
- teacher_forcing_ratio: 0.0
- emb_dropout: 0.0

Best best_valid_perplexity:
Value: 53.2197
Configuration:

- emb_dim: 128
- hid_dim: 128
- batch_size: 1152
- teacher_forcing_ratio: 0.0
- emb_dropout: 0.0

Best best_valid_loss:
Value: 3.9744
Configuration:

- emb_dim: 128
- hid_dim: 128
- batch_size: 1152
- teacher_forcing_ratio: 0.0
- emb_dropout: 0.0

Parameter Impact Analysis

emb_dim analysis:

- **Impact on final_bleu:**
  Best value: 128
  Mean performance: 0.1724
  Std deviation: 0.0748
- **Impact on best_valid_perplexity:**
  Best value: 128
  Mean performance: 114.9863
  Std deviation: 53.3517
- **Impact on best_valid_loss:**
  Best value: 128
  Mean performance: 4.6579
  Std deviation: 0.4362

hid_dim analysis:

- **Impact on final_bleu:**
  Best value: 128
  Mean performance: 0.1724
  Std deviation: 0.0748
- **Impact on best_valid_perplexity:**
  Best value: 128
  Mean performance: 114.9863
  Std deviation: 53.3517
- **Impact on best_valid_loss:**
  Best value: 128
  Mean performance: 4.6579
  Std deviation: 0.4362

batch_size analysis:

- **Impact on final_bleu:**
  Best value: 1152
  Mean performance: 0.3122
  Std deviation: nan

- **Impact on best_valid_perplexity:**
  Best value: 1152
  Mean performance: 53.2197
  Std deviation: nan

- **Impact on best_valid_loss:**
  Best value: 1152
  Mean performance: 3.9744
  Std deviation: nan

teacher_forcing_ratio analysis:

- **Impact on final_bleu:**
  Best value: 0.0
  Mean performance: 0.1826
  Std deviation: 0.0811

- **Impact on best_valid_perplexity:**
  Best value: 0.5
  Mean performance: 99.8202
  Std deviation: nan

- **Impact on best_valid_loss:**
  Best value: 0.0
  Mean performance: 4.5554
  Std deviation: 0.3760

emb_dropout analysis:

- **Impact on final_bleu:**
  Best value: 0.0
  Mean performance: 0.2794
  Std deviation: 0.0465

- **Impact on best_valid_perplexity:**
  Best value: 0.0
  Mean performance: 66.8270
  Std deviation: 19.2437

- **Impact on best_valid_loss:**
  Best value: 0.0
  Mean performance: 4.1809
  Std deviation: 0.2920

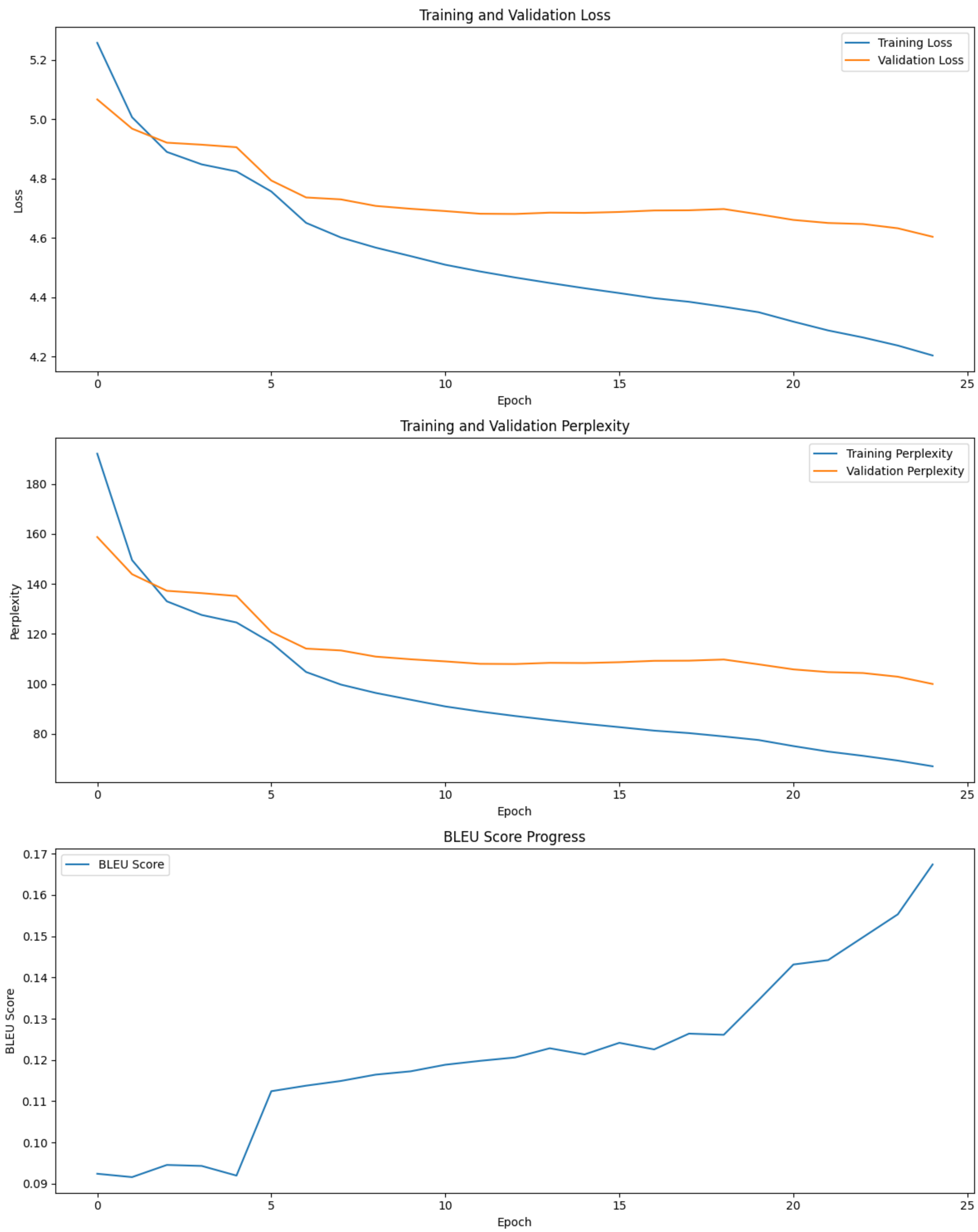

Figure 1: Experiment 1

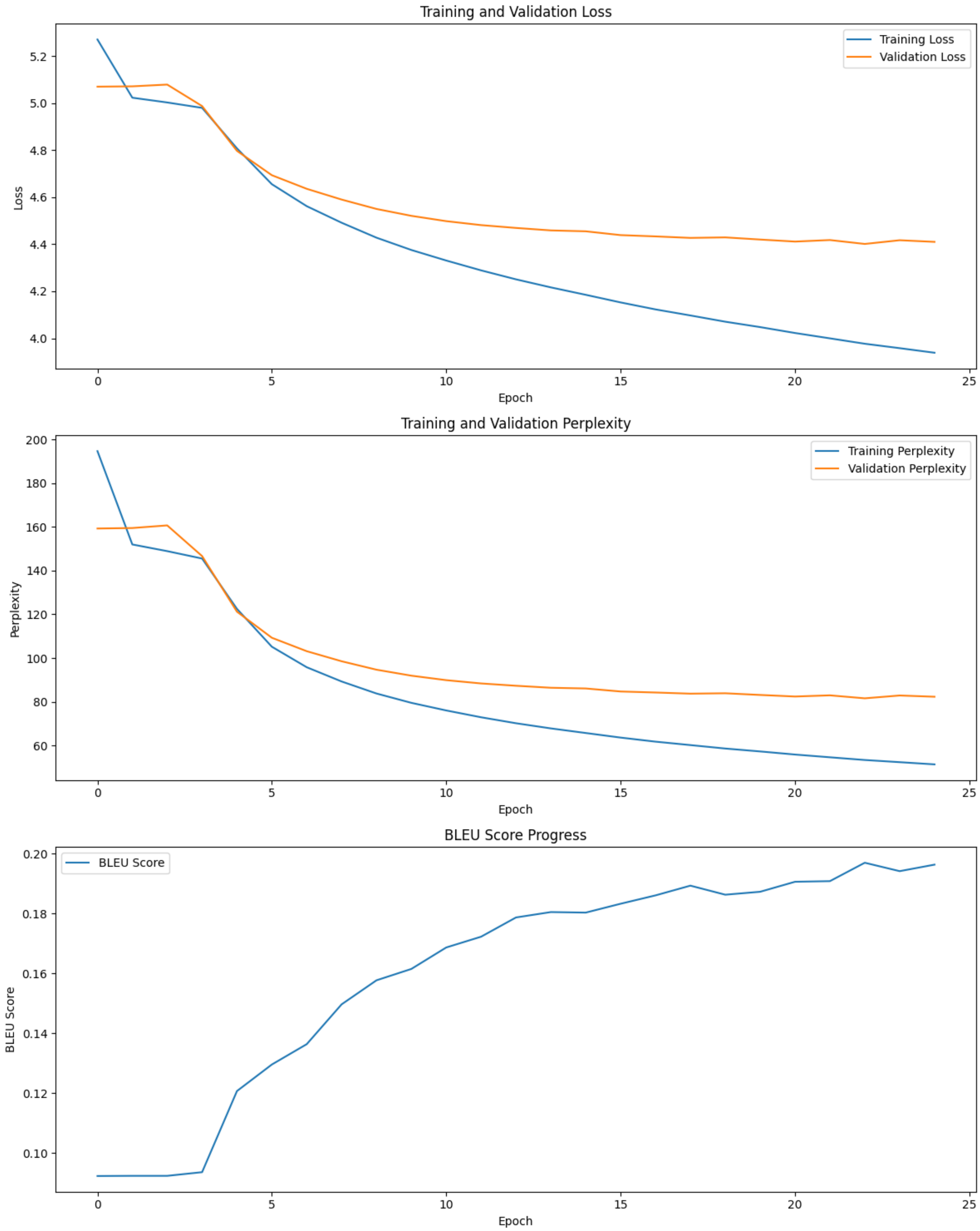

Figure 2: Experiment 2

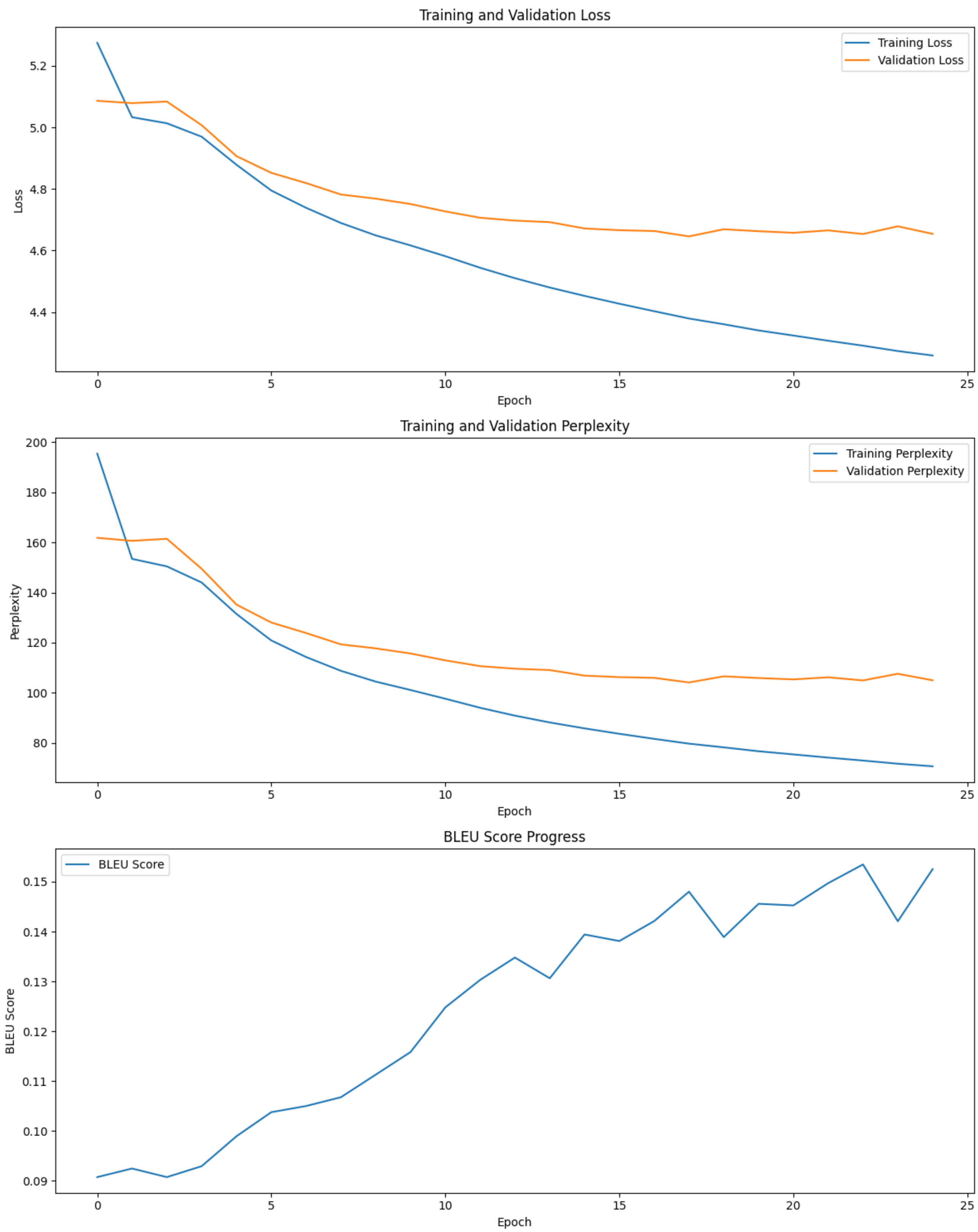

Figure 3: Experiment 3

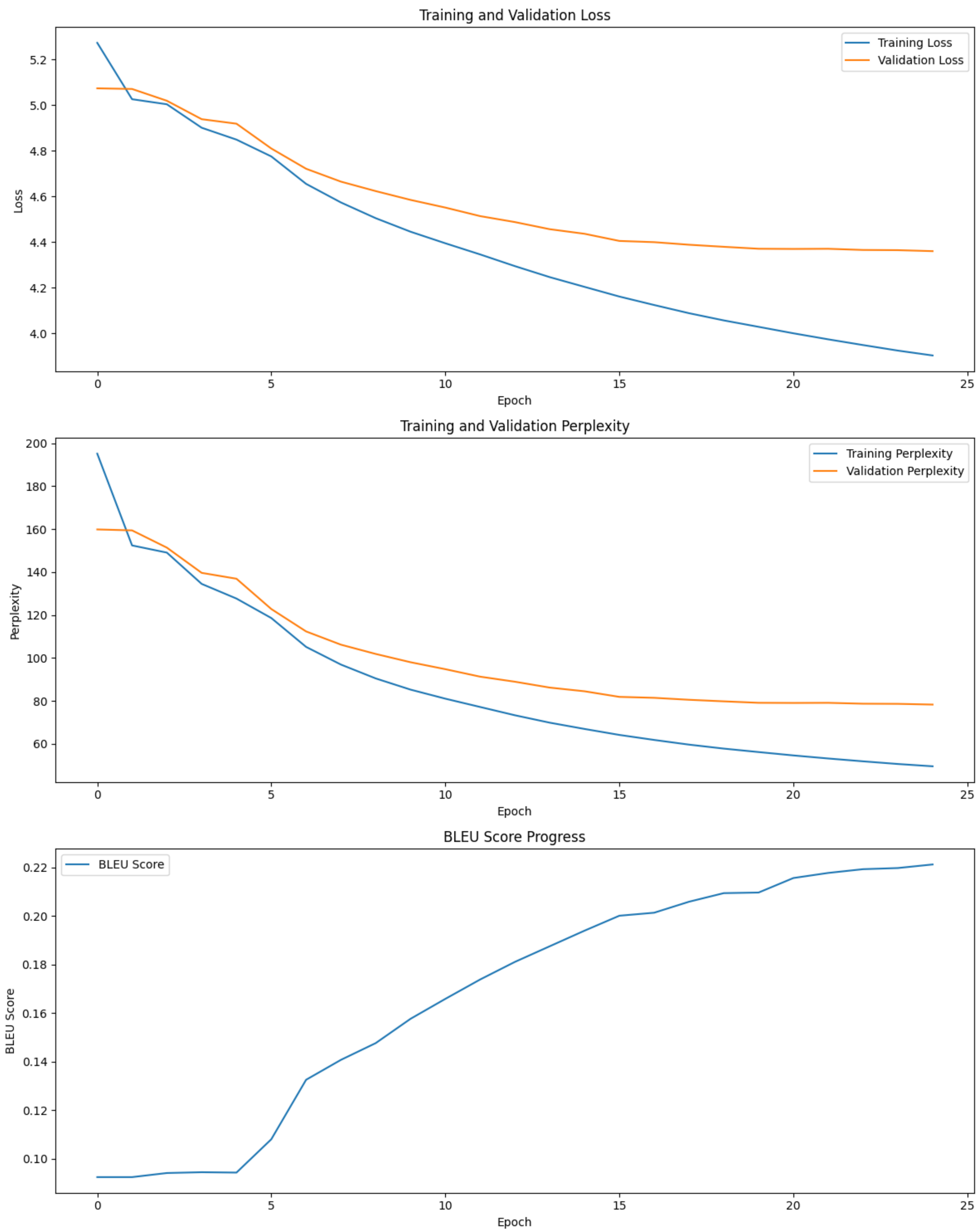

Figure 4: Experiment 4

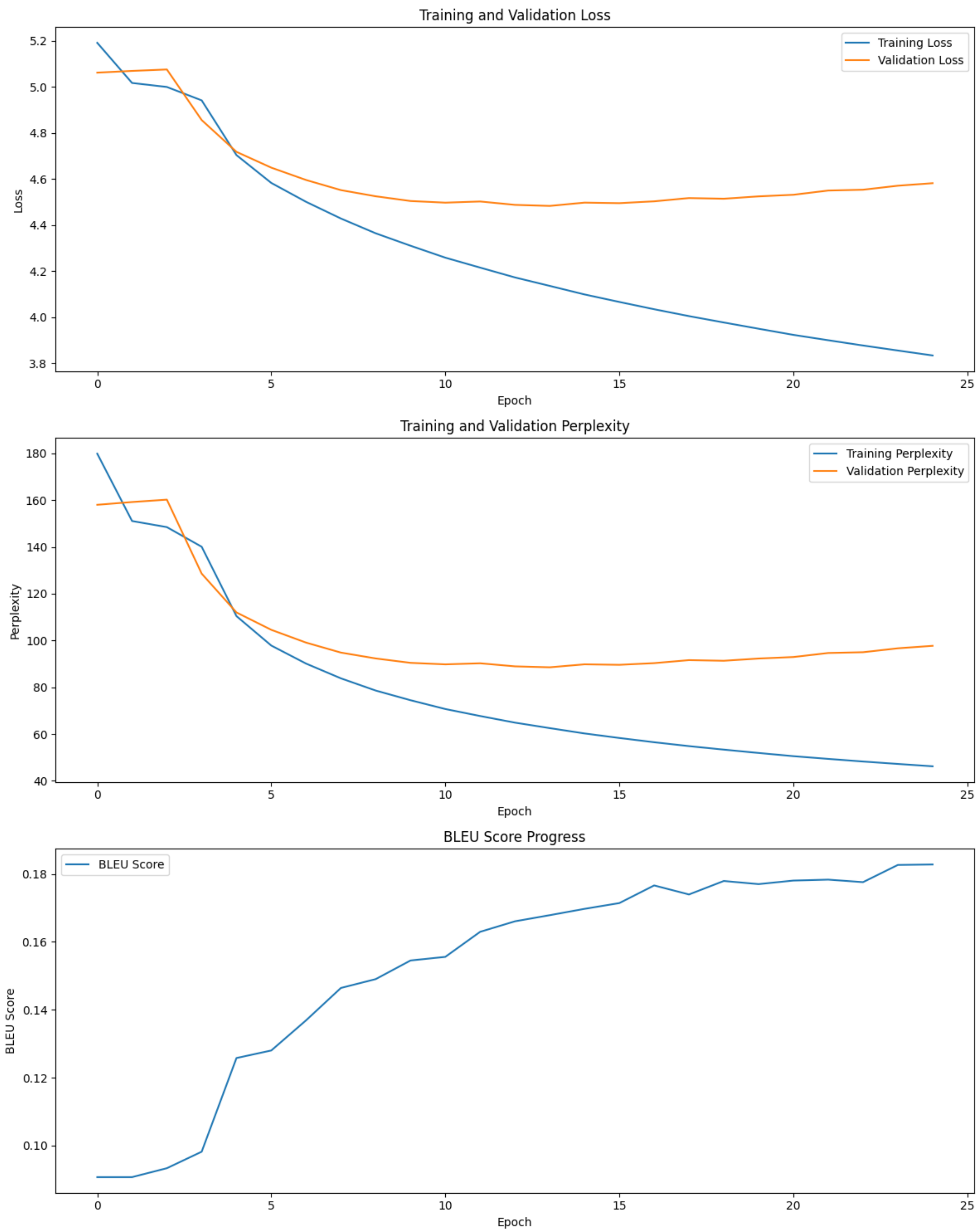

Figure 5: Experiment 5

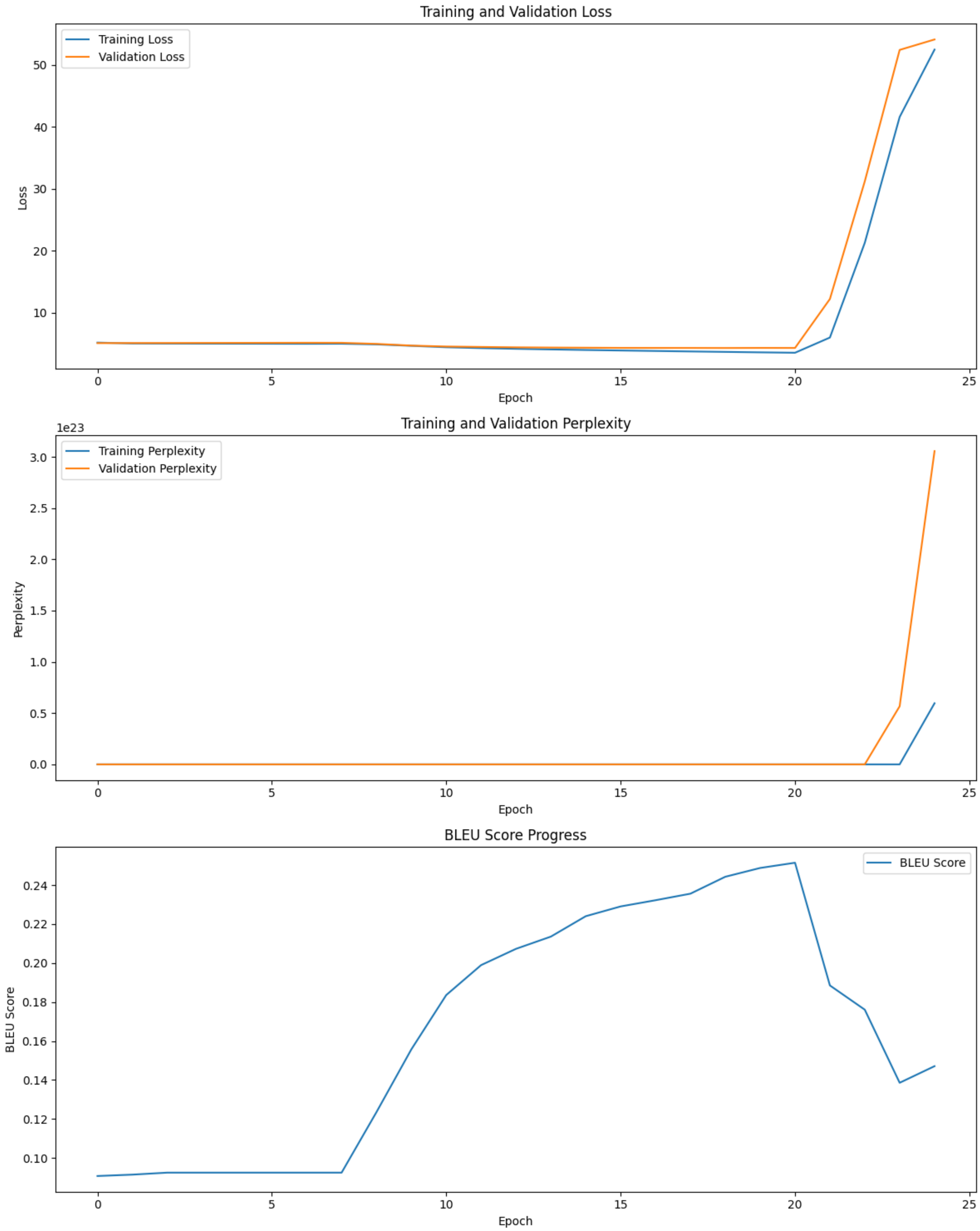

Figure 6: Experiment 6

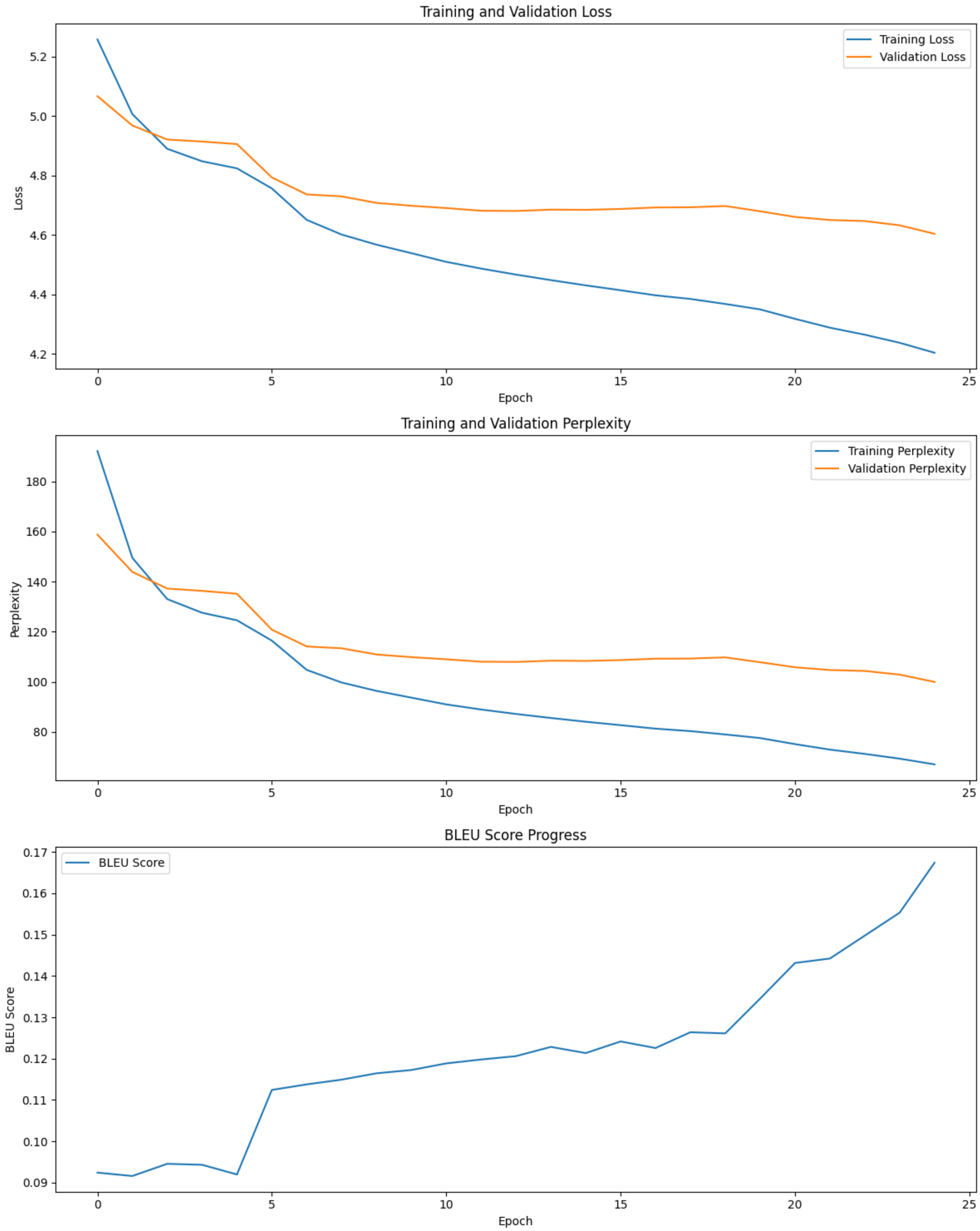

Figure 7: Experiment 7

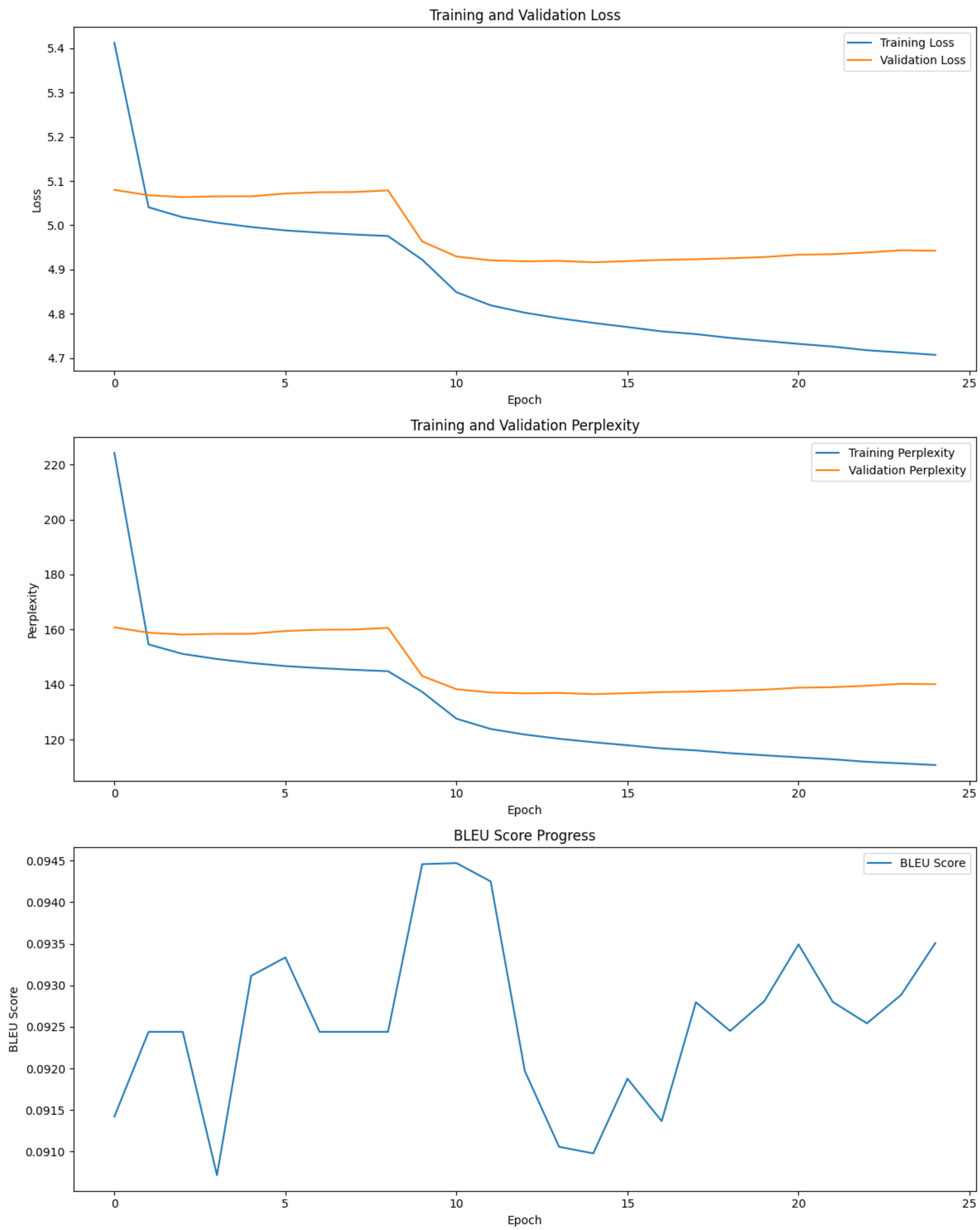

Figure 8: Experiment 8

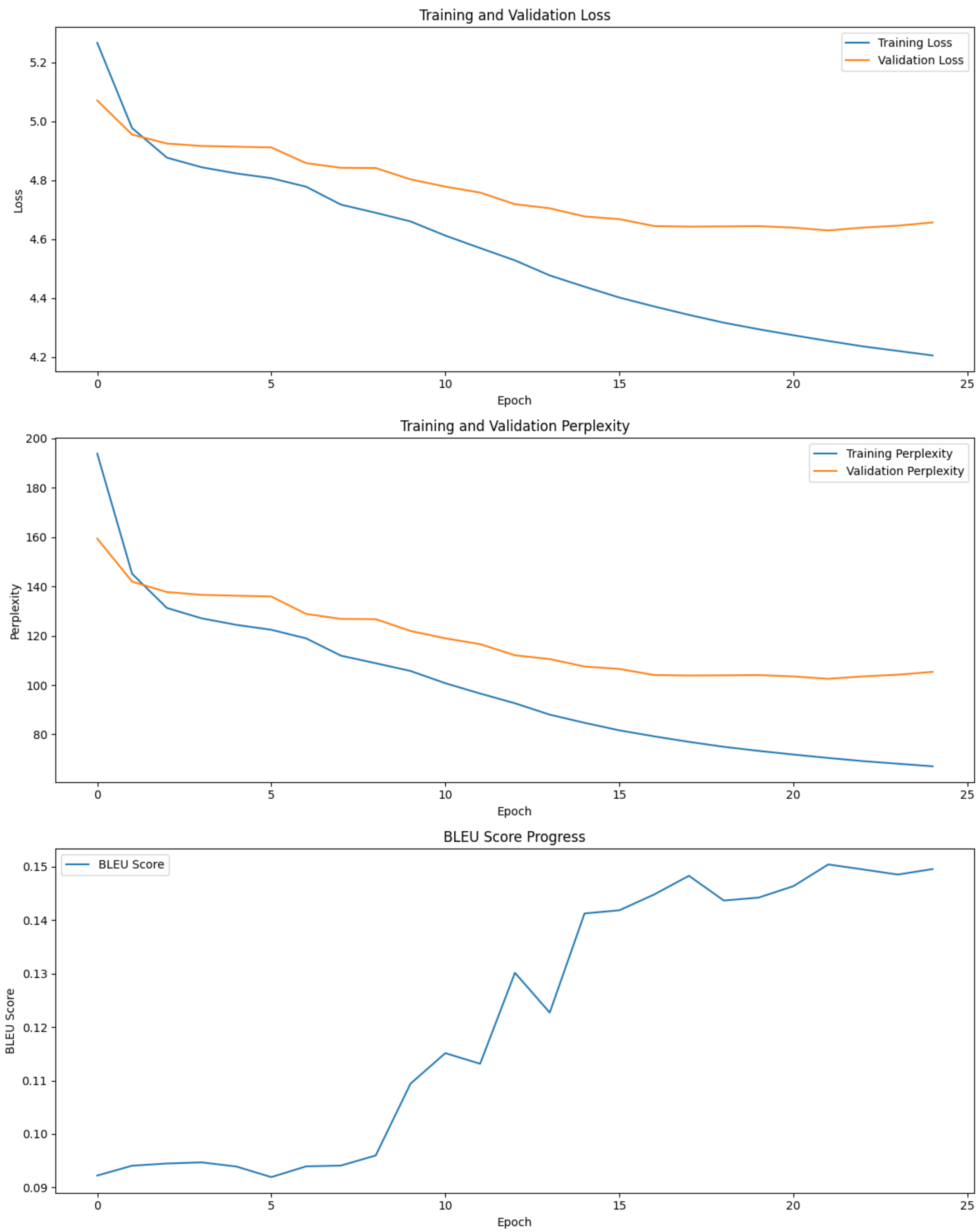

Figure 9: Experiment 9

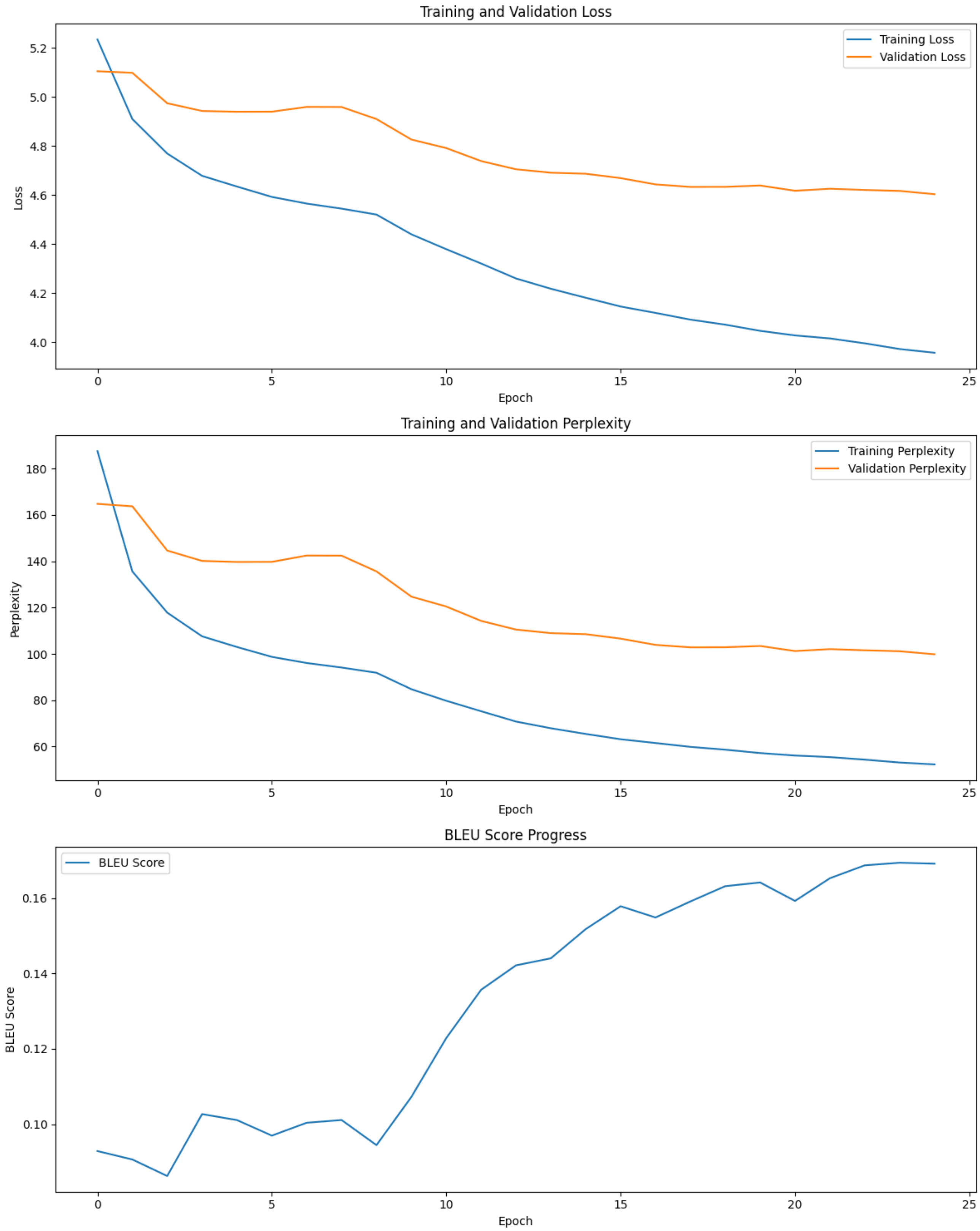

Figure 10: Experiment 10

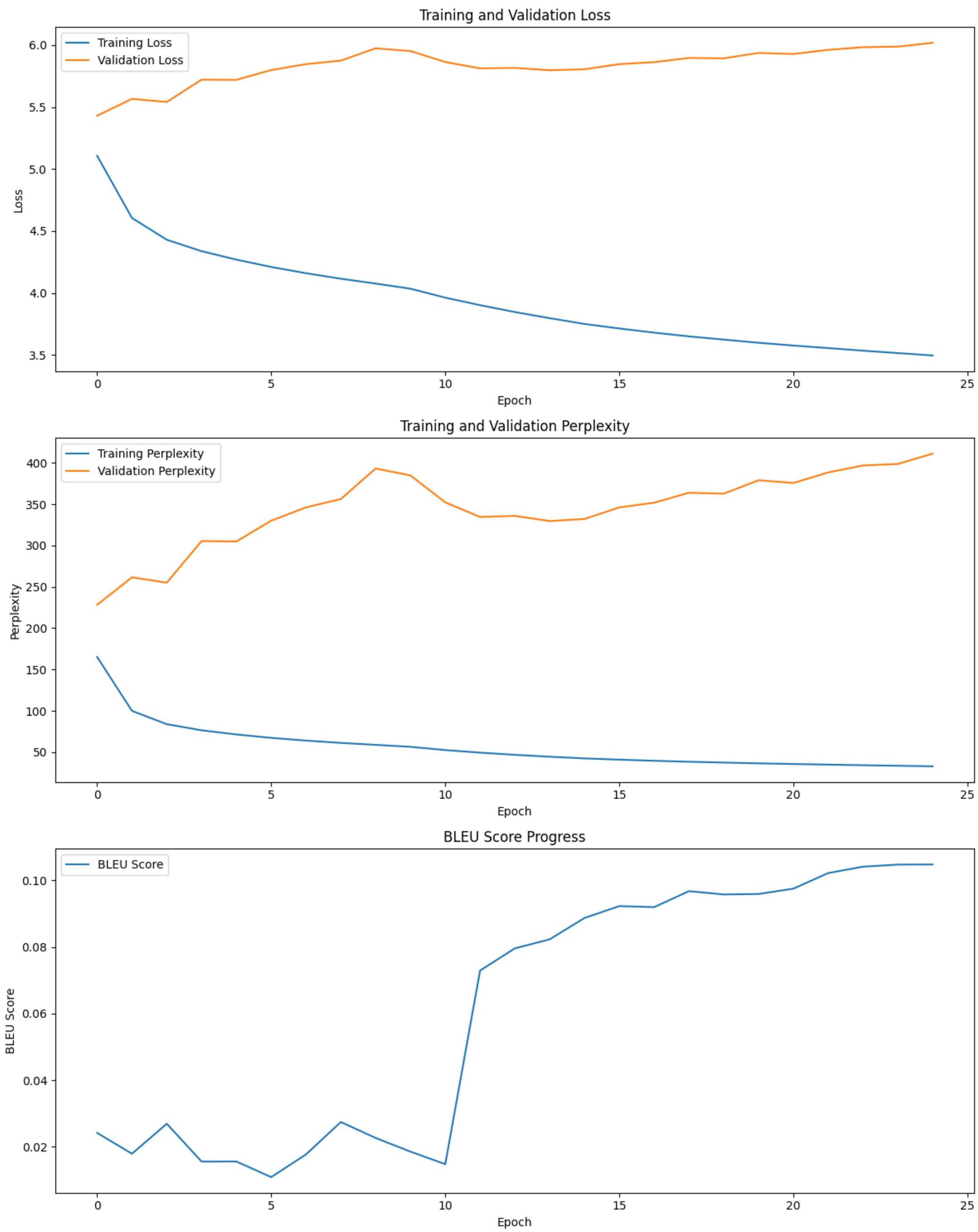

Figure 11: Experiment 11

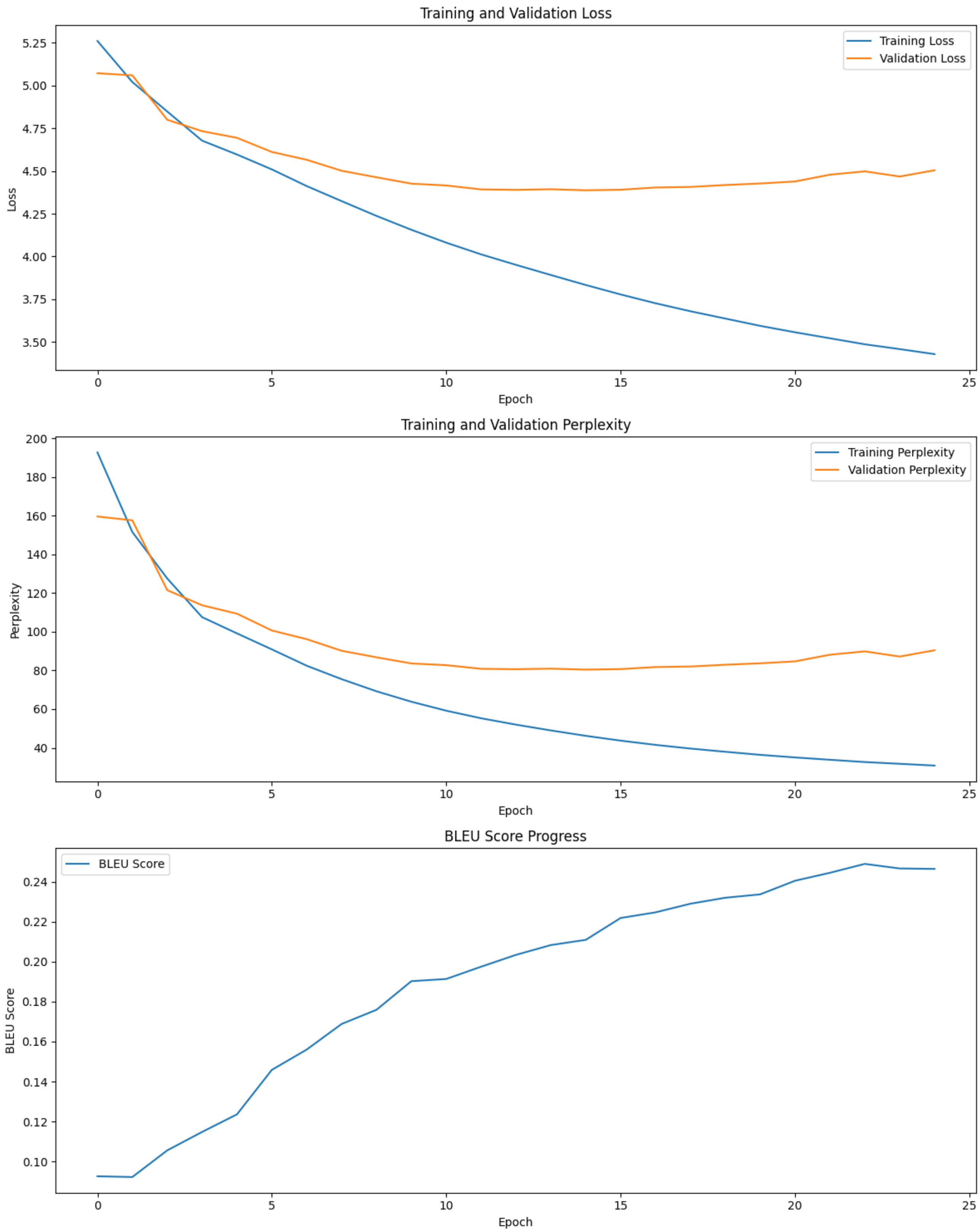

Figure 12: Experiment 12

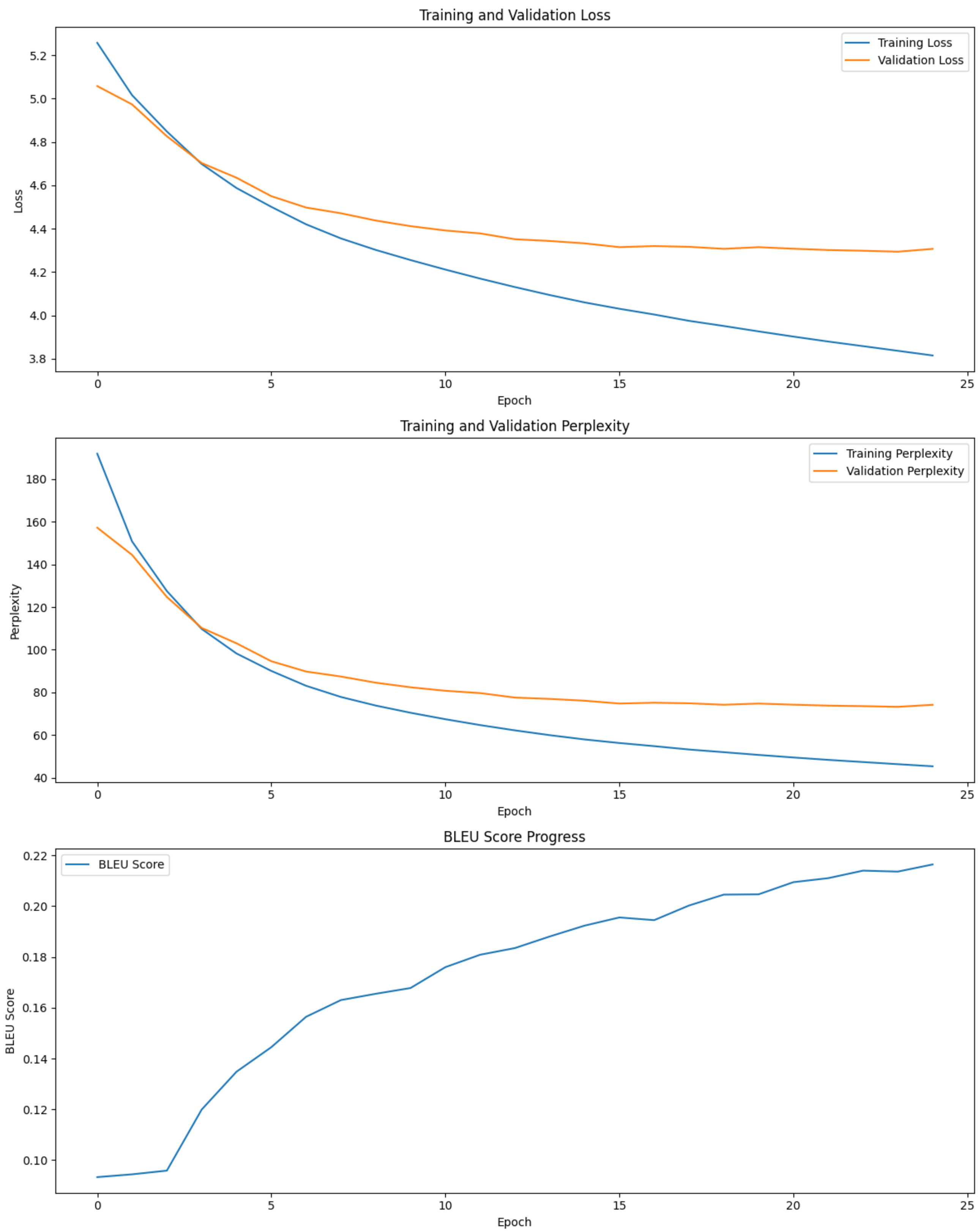

Figure 13: Experiment 13

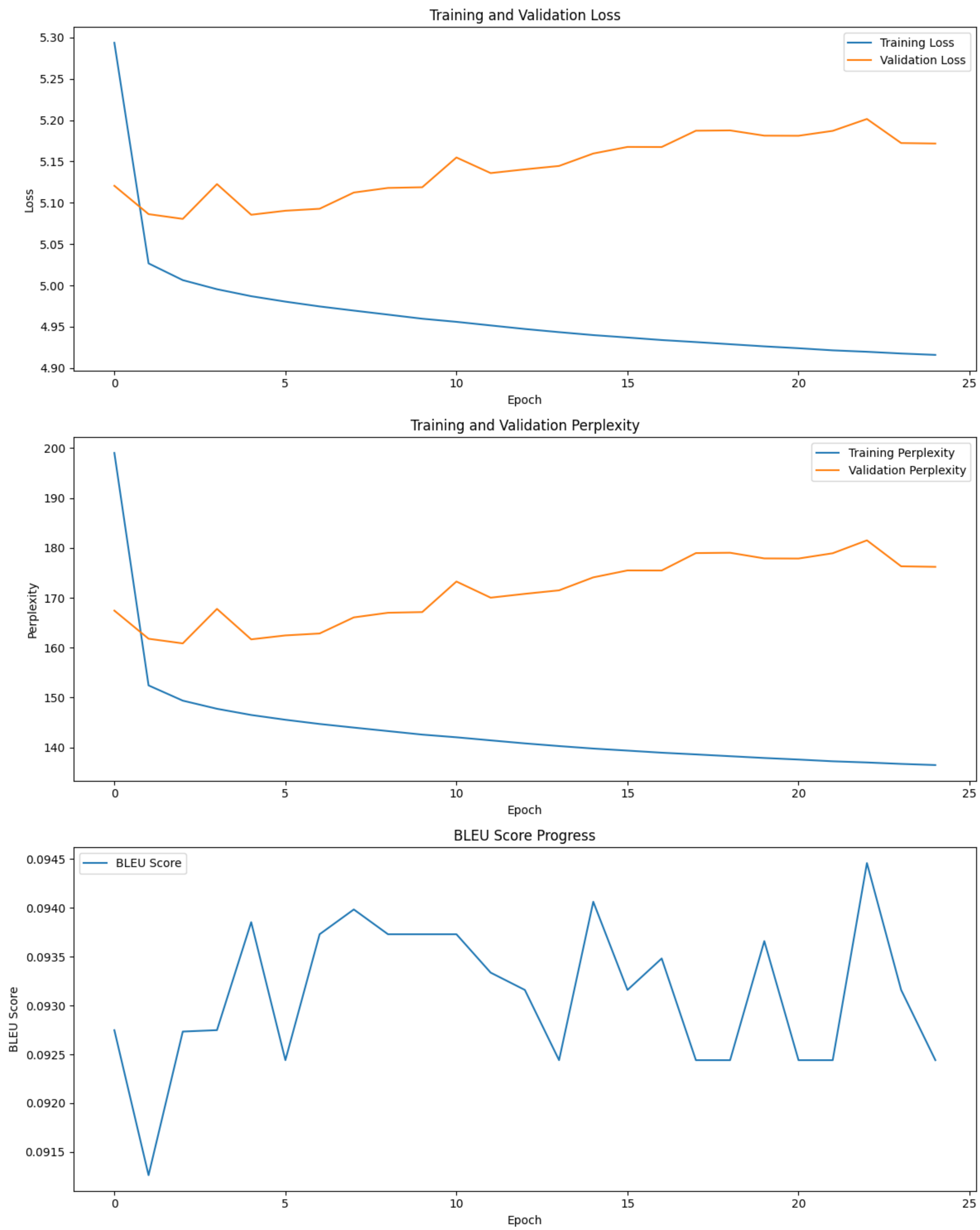

Figure 14: Experiment 14

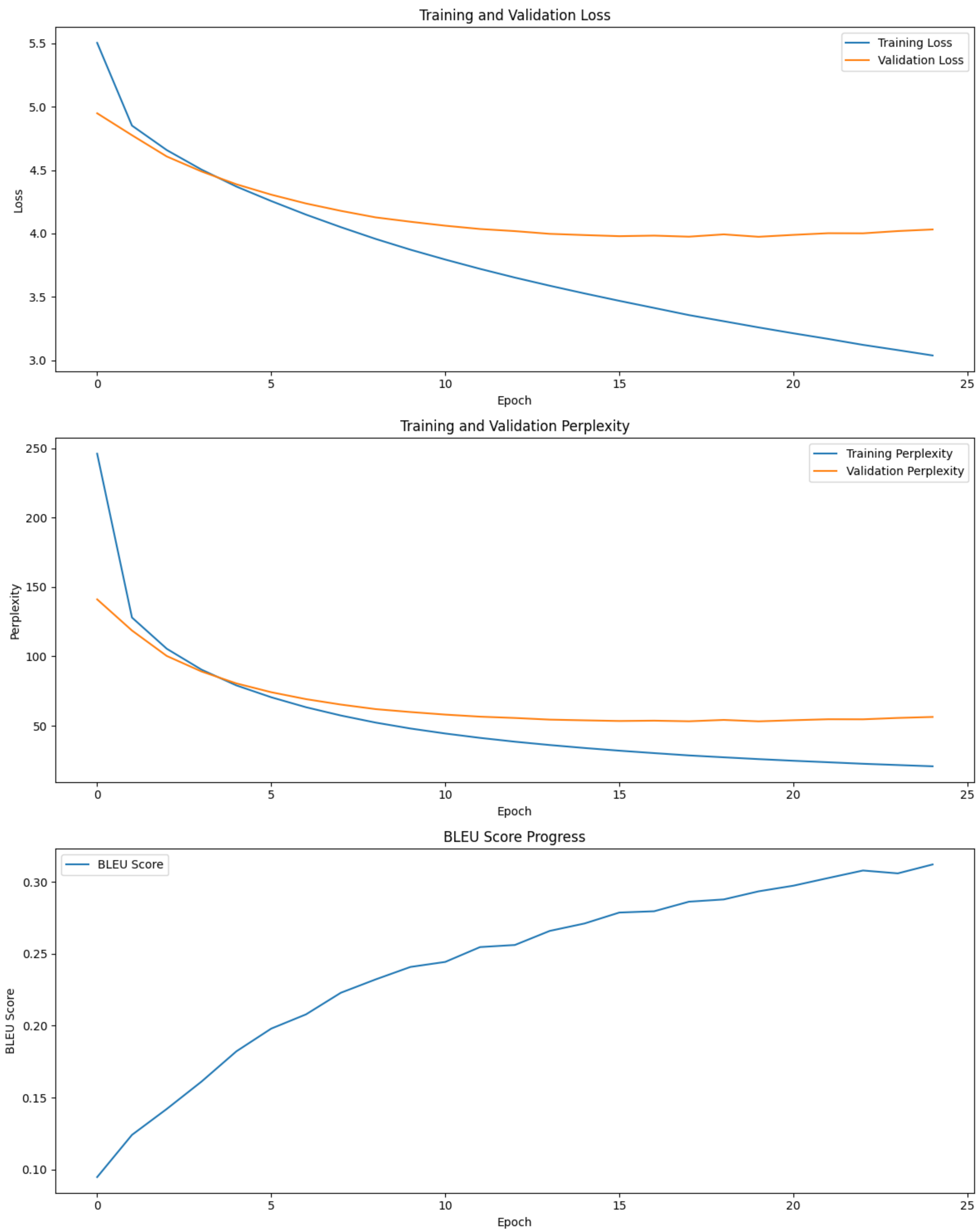

Figure 15: Bidirectional Experiment

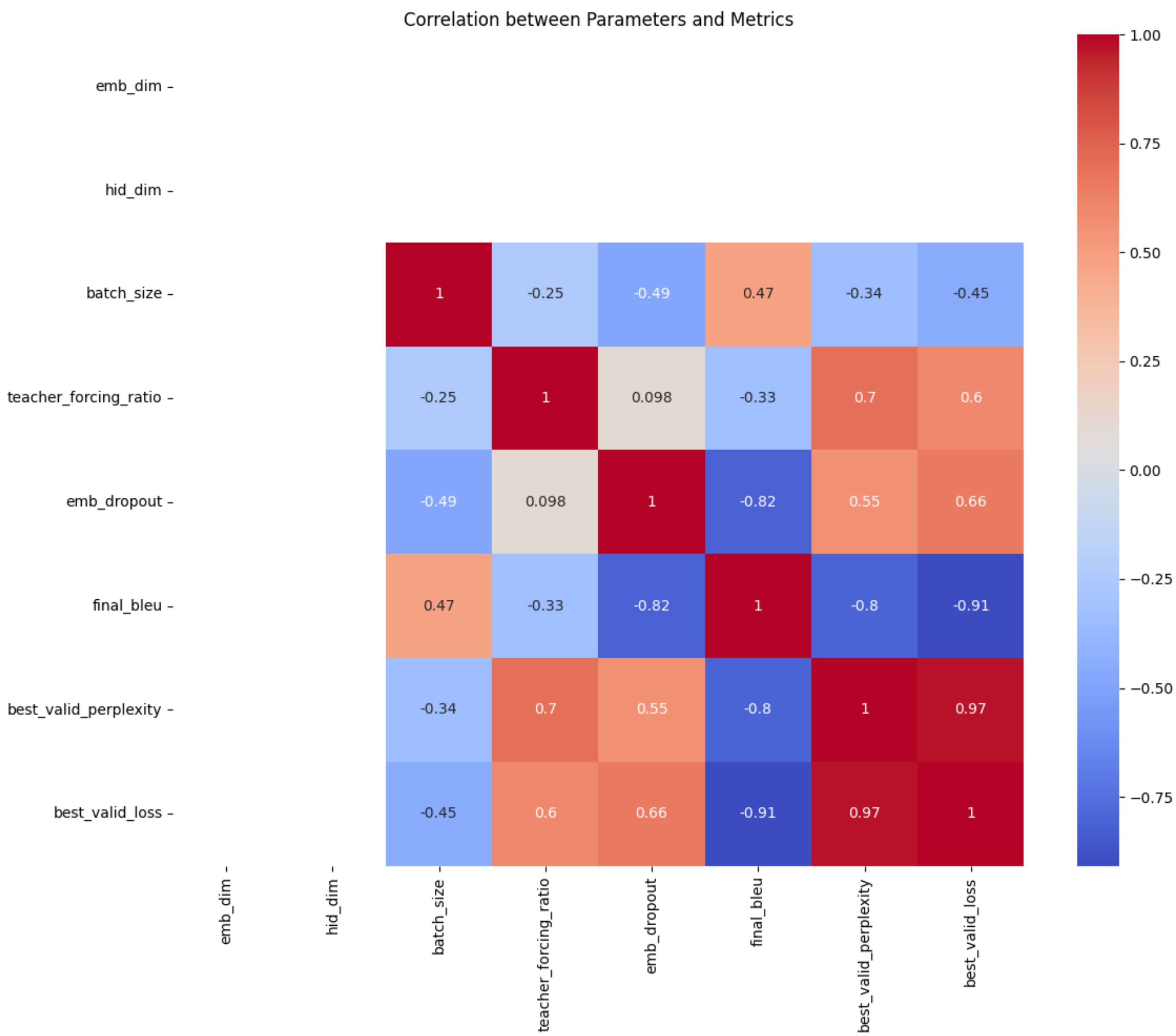

Figure 16: Correlation Matrix

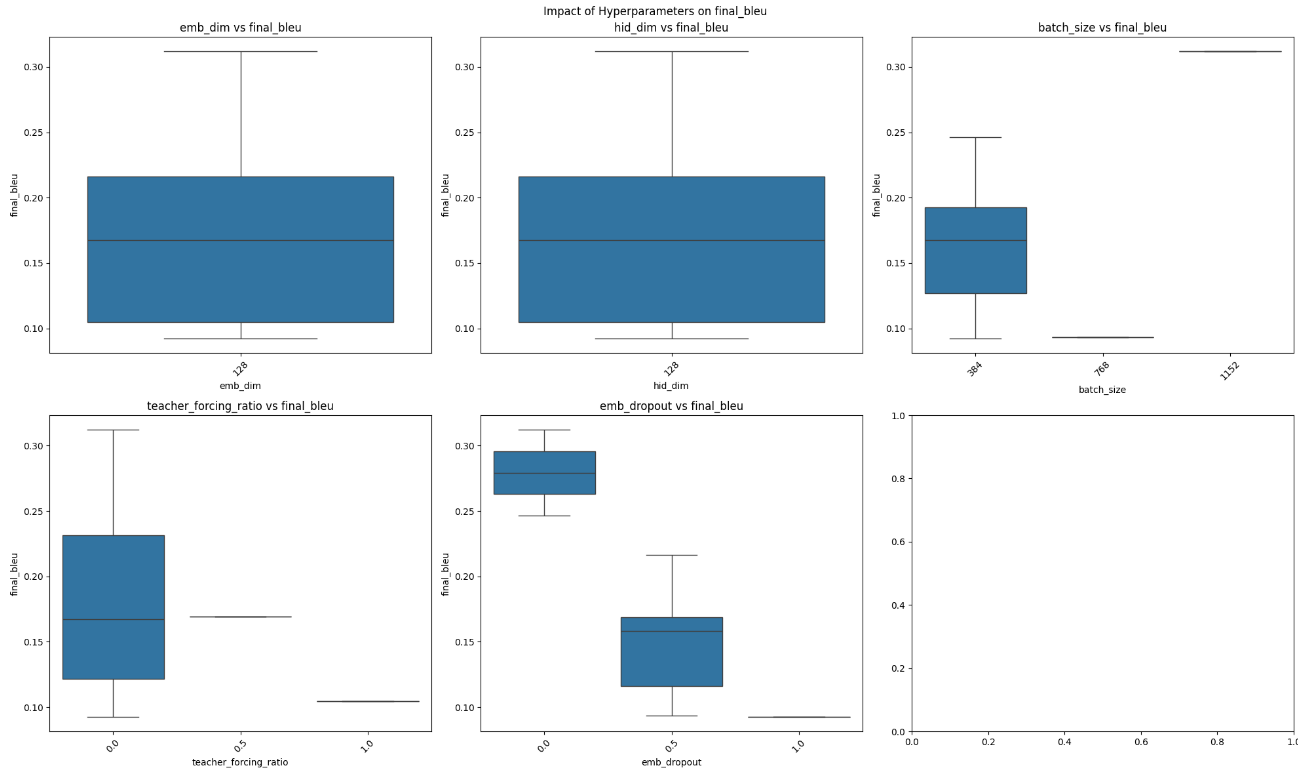

Figure 17: Final Bleu Analysis

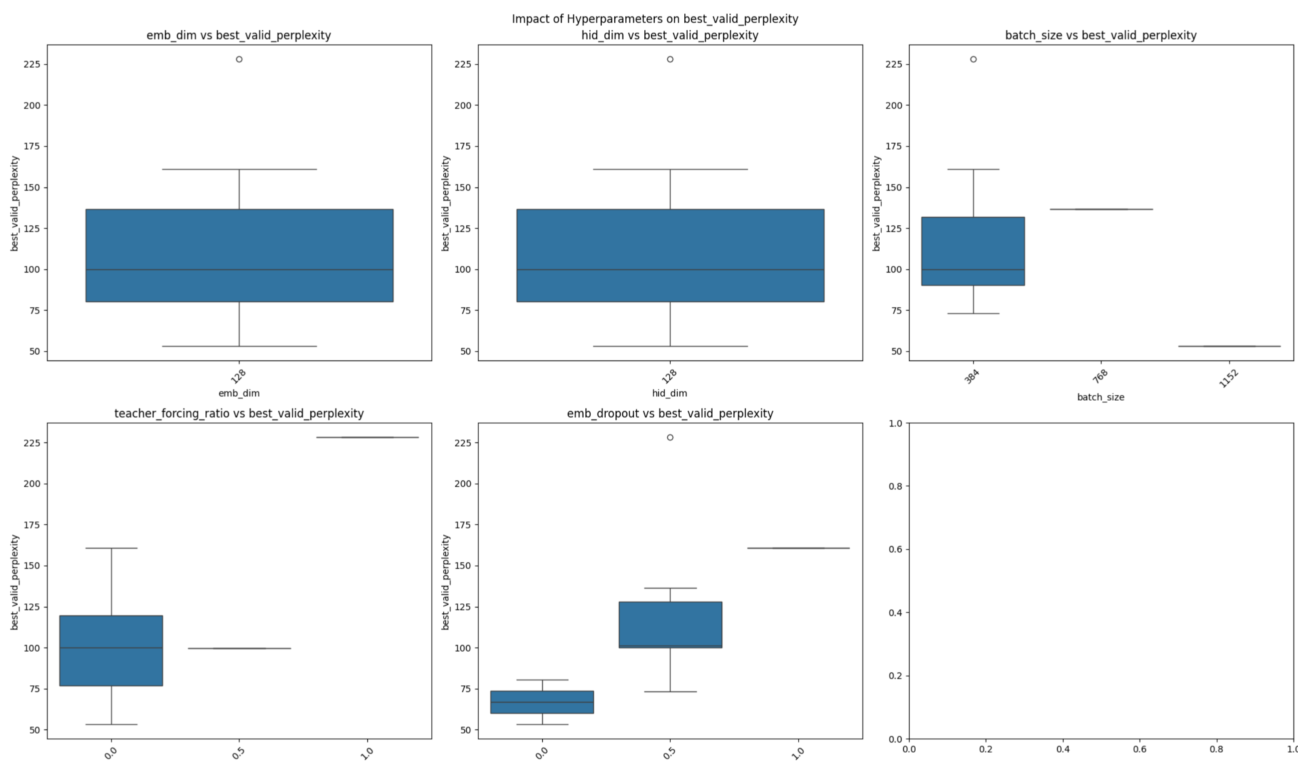

Figure 18: Final Perplexity Analysis

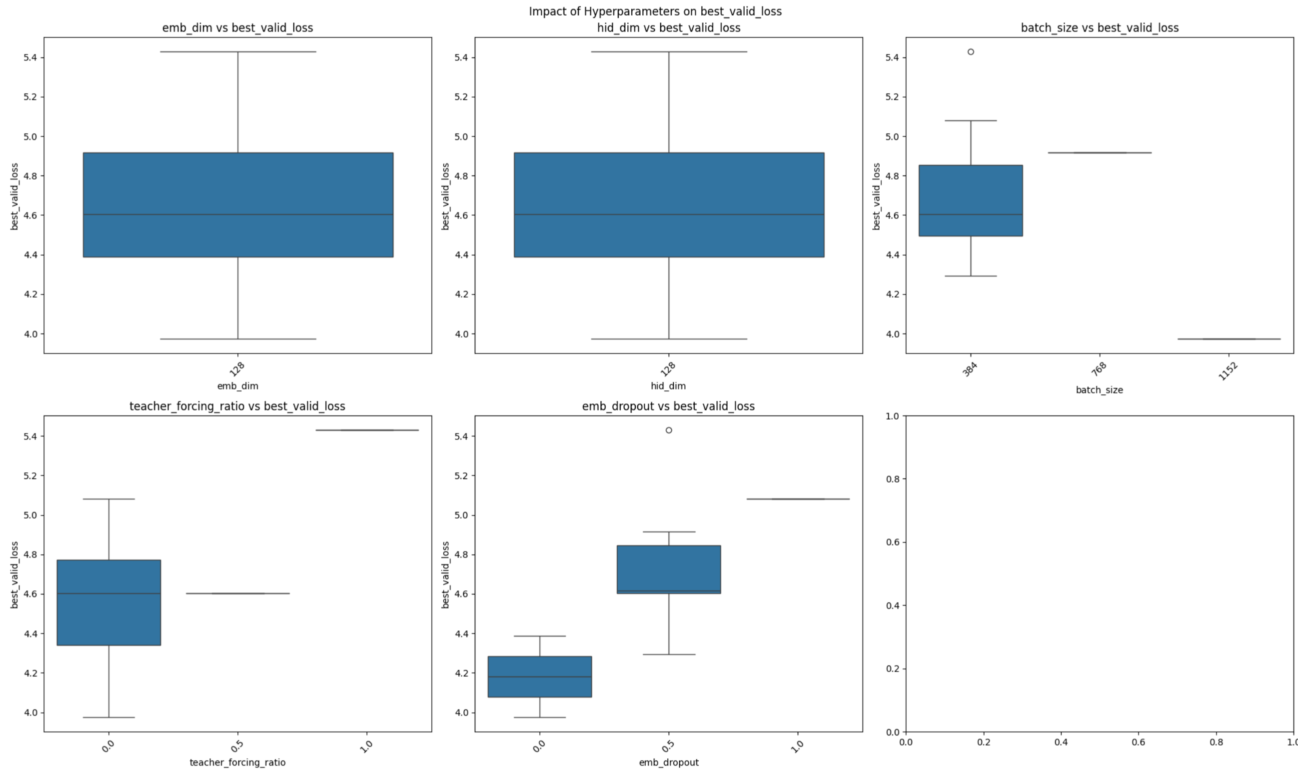

Figure 19: Final Loss Analysis

3 Conclusion

This report tells the challenges and trade-offs in designing RNN-based models for sequential data tasks. While enhancements such as bidirectional encoders and increased hidden dimensions can improve model performance, resource constraints often limit these gains. Future work could explore deeper architectures, extended training epochs, and attention mechanisms to further improve results.