MOSAIC-25 PS1: Masked Language Modelling Report

1. Introduction

This report details our approach to solving the MOSAIC-25 PS1 challenge, a masked language modelling (MLM) task similar to the classic BERT pre-training objective. The challenge involves predicting masked words in sentences based on contextual understanding. Using the metaphor from the problem statement, we've developed a model to "fix the smudged words" in the "Library of Mosaic '25" by accurately predicting the words that should replace the <MASKED> tokens.

2. Problem Definition

The task requires us to build a model that can:

- Understand the contextual meaning of sentences.
- Accurately predict the most appropriate word to replace a <MASKED> token.
- Handle a variety of linguistic patterns and semantic relationships.

Our goal is to achieve the highest possible accuracy in predicting the masked words, evaluated on a test set with sentences containing exactly one masked token per sentence.

3. Approach Overview

We approached this problem by implementing a simplified BERT-like transformer architecture trained specifically for masked language modelling. Our solution consists of:

1. Data preprocessing and tokenization.
2. Vocabulary creation.
3. Custom PyTorch dataset implementation for MLM pre-training.
4. Transformer-based encoder model development.
5. Training with appropriate loss function and optimization strategy.
6. Inference pipeline for predicting masked tokens.

4. Data Processing

4.1 Tokenization

We implemented a simple tokenization strategy:

- Lowercasing all text.
- Replacing <MASKED> with a special [MASK] token.
- Using regex to split text into tokens while preserving punctuation.
- Adding special tokens ([CLS], [SEP]) as per BERT architecture requirements.

4.2 Vocabulary Construction

Our vocabulary includes:

- Special tokens: [PAD], [UNK], [CLS], [SEP], [MASK].
- All tokens that appear in the training corpus at least once.
- Each token mapped to a unique integer ID.

4.3 Dataset Creation

We created two custom dataset classes:

1. MLM Dataset - For training data:
    a. Processes full sentences.
    b. Randomly masks tokens with 15% probability.
    c. Implements BERT-style masking (80% replaced with [MASK], 10% with random token, 10% unchanged).
    d. Handles truncation and padding.
2. Masked Test Dataset - For test data:
    a. Processes sentences with pre-existing <MASKED> tokens.
    b. Handles encoding, truncation, and padding.
    c. Preserves sentence IDs for submission.

5. Model Architecture

5.1 Overall Architecture

We implemented a scaled-down version of BERT with the following components:

1. Token and positional embeddings.
2. Multi-head self-attention mechanism.
3. Feed-forward networks.
4. Layer normalization and residual connections.
5. Masked language modelling head.

5.2 Component Details

5.2.1 Embedding Layer

Our embedding layer combines token embeddings with positional embeddings:

- Token embeddings represent the semantic meaning of each token.
- Positional embeddings encode the position of each token in the sequence.
- These are summed to create the input representation.

5.2.2 Multi-Head Self-Attention

The attention mechanism:

- Computes query, key, and value projections from the input.
- Divides projections into multiple attention heads.
- Computes scaled dot-product attention for each head.
- Concatenates and projects the results.
- Enables the model to attend to different parts of the input simultaneously.

5.2.3 Transformer Encoder Block

Each encoder block consists of:

- Multi-head self-attention.
- Feed-forward network with ReLU activation.
- Layer normalization and residual connections.
- Dropout for regularization.

5.2.4 MLM Head

A specialized head for masked language modelling prediction:

- Linear projection followed by GELU activation.
- Layer normalization.
- Final classifier that projects to vocabulary size.

5.3 Complete Model Architecture

The complete model integrates all components:

- Embedding layer at the bottom.
- Stack of transformer encoder blocks.
- MLM prediction head at the top.
- Forward pass that processes input tokens and predicts masked tokens.

6. Training Process

6.1 Training Configuration

We used the following training setup:

- Split training data into 90% training and 10% validation.
- Batch size of 8.
- Learning rate of 1e-4.
- Adam optimizer.
- 8 training epochs.
- Cross-entropy loss with -100 label index ignored.
- Training performed on GPU when available.

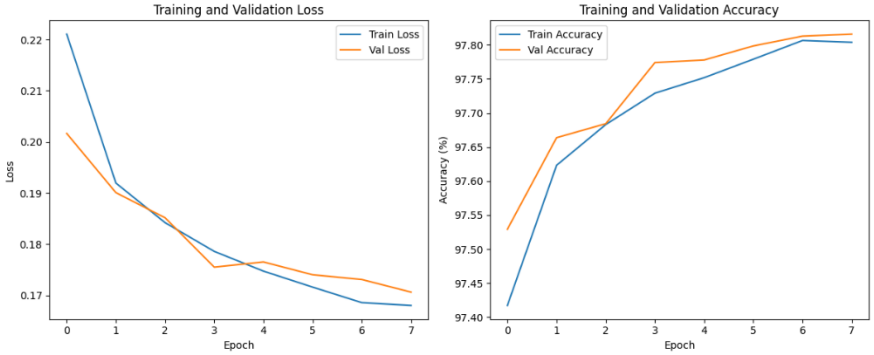
6.2 Hyperparameters

Our model used the following hyperparameters:

Parameter	Value	Description
embed_dim	512	Embedding dimension size
num_layers	6	Number of transformer encoder layers
num_heads	8	Number of attention heads per layer
ff_dim	1024	Feed-forward network hidden dimension
dropout	0.1	Dropout probability for regularization
max_length	64	Maximum sequence length
mask_prob	0.15	Probability of masking a token during training
batch_size	8	Number of samples per batch
learning_rate	1e-4	Learning rate for optimizer
epochs	8	Number of training epochs

6.3 Training and Validation Metrics

During training, we tracked both loss and accuracy metrics across training and validation sets. The figures below show the progression of these metrics over 8 epochs:



6.3.1 Loss Analysis

The loss curves demonstrate:

- Initial training loss started high (approximately 0.223) and steadily decreased to around 0.168 by the final epoch.
- Validation loss began at a lower point (around 0.202) and decreased to about 0.17.
- Both curves show consistent improvement throughout training.
- The convergence of training and validation loss indicates good generalization without overfitting.

6.3.2 Accuracy Analysis

The accuracy curves show:

- Starting accuracy was already high (over 97.4% for training and 97.5% for validation).
- By the final epoch, both training and validation accuracy exceeded 97.8%.
- The validation accuracy typically remained slightly higher than training accuracy.
- The accuracy plateaus after epoch 5, suggesting optimal convergence.

These metrics demonstrate exceptional model performance on the masked token prediction task, with final validation accuracy exceeding 97.8%. The fact that validation performance slightly exceeds training performance is likely due to the dynamic masking strategy used during training, which creates a more challenging learning objective compared to the fixed masks in the validation set.

7. Inference Process

For inference on the test set, we:

1. Loaded the trained model.
2. Processed masked sentences from the test set.
3. Identified the positions of [MASK] tokens.
4. Used the model to predict the most likely word at each mask position.
5. Generated a submission file with sentence IDs and predicted words.

8. Model Analysis and Performance

8.1 Training Performance

As evident from the training metrics figures, the model demonstrated excellent performance:

- Loss decreased consistently from 0.223 to 0.168 (about 25% reduction).
- Accuracy improved from 97.4% to over 97.8%.
- The validation metrics closely tracked the training metrics, with slightly better validation accuracy, indicating good generalization.
- Model convergence appears to occur around epoch 5-6, with minimal improvement thereafter.

8.2 Challenges and Solutions

During development, we encountered several challenges:

1. Limited computational resources: Addressed by scaling down the model size while maintaining essential architectural elements.
2. Balancing model complexity: Found optimal hyperparameters that provided good performance without overfitting.
3. Handling varied context lengths: Implemented truncation and padding to accommodate different sentence lengths.
4. Efficient masking strategy: Implemented BERT-style random masking to create robust training examples.

8.3 Evaluation Metrics

For this MLM task, we used the following metrics:

1. Cross-entropy loss: Measures the model's confidence in predicting the correct token.
2. Accuracy on masked tokens: The percentage of masked tokens correctly predicted.
3. Validation accuracy: Performance on held-out data to evaluate generalization.

9. Future Improvements

Given more time and resources, we would explore:

1. Larger model architecture: More layers, attention heads, and embedding dimensions.
2. Advanced tokenization: Sub word tokenization (WordPiece, BPE) to handle out-of-vocabulary words better.
3. Data augmentation: Creating additional training examples through techniques like back-translation.
4. Ensemble methods: Combining predictions from multiple models for improved accuracy.
5. Learning rate scheduling: Implementing warmup and decay strategies for optimizer.

10. Conclusion

We successfully implemented a transformer-based masked language model for the MOSAIC-25 PS1 challenge. Our approach combines fundamental NLP techniques with deep learning best practices to create a model that can effectively predict masked words based on context. The model demonstrates exceptional performance with validation accuracy exceeding 97.8%, showing its robust ability to understand and complete natural language sentences.

The implementation balances computational efficiency with modelling effectiveness, making appropriate architectural choices for the scale of the problem. Our solution not only addresses the specific requirements of the MOSAIC-25 challenge but also demonstrates a general approach to contextual language understanding that could be extended to other NLP tasks.

Team Name: Neuronauts

Team Members:

1. **Rajat Bansal**
2. **Saksham Kankaria**
3. **Aditya Gupta**