Question Answering Model Implementation Report

Samaa Soliman

April 22, 2025

Question Answering Model Implementation Report

1. Data Processing Pipeline

Dataset

- Using the SQuAD dataset (Stanford Question Answering Dataset)
- Dataset is subsampled to 10,000 training examples and 2,000 validation examples
- Each example contains: context paragraph, question, and answer text

Preprocessing Steps

1. **Text Preparation**
   - Answer sequences are wrapped with special tokens:
     - Input answers: `<start> {answer}`
     - Output answers: `{answer} <end>`
2. **Tokenization**
   - Vocabulary size: 20,000 tokens
   - Special tokens: `<PAD>`, `<UNK>`, `<start>`, `<end>`
   - Custom filters to preserve `<` and `>` characters
   - Two-stage tokenization:
     1. Fit on special tokens first (ensuring low indices)
     2. Fit on all texts (contexts, questions, answers)
3. **Sequence Processing**
   - Maximum lengths:
     - Context: 300 tokens
     - Questions: 30 tokens
     - Answers: 30 tokens
   - All sequences are padded/truncated to these lengths
   - Decoder inputs are ensured to start with `<start>`
   - Decoder outputs are ensured to end with `<end>`

2. Model Architecture

Overview

The model follows the Transformer architecture with separate encoder and decoder stacks.

Components

1. **Input Processing**
   - Three input streams:

    - Context sequence
    - Question sequence
    - Answer sequence (for training)
  - Context and question are concatenated before encoding
2. **Embeddings**
  - Shared embedding layer (20,000 vocab × 256 dimensions)
  - Positional encoding added to both encoder and decoder embeddings
3. **Encoder Stack**
  - 3 identical layers
  - Each layer contains:
    - Multi-head attention (8 heads)
    - Feed-forward network (1024 units)
    - Layer normalization and residual connections
    - Dropout rate: 0.2
4. **Decoder Stack**
  - 3 identical layers
  - Each layer contains:
    - Masked multi-head self-attention
    - Multi-head attention over encoder output
    - Feed-forward network
    - Layer normalization and residual connections
5. **Output Layer**
  - Dense projection to vocabulary size
  - Temperature scaling (T=0.7)
  - Softmax activation

3. Training Process

Loss Function

- Custom masked loss function with three key features:

1. **Label Smoothing (0.1)**
  - Prevents the model from becoming over-confident
  - Instead of training with one-hot targets (e.g., [0, 1, 0]), uses smoothed targets (e.g., [0.03, 0.94, 0.03])
  - Benefits:
    - Improves model calibration (predicted probabilities better reflect true uncertainty)
    - Regularizes the model by preventing it from assigning full probability to training samples
    - Makes the model more robust to noisy labels
    - Helps prevent memorization of training data
2. **Padding Token Masking**
  - Ignores padding tokens in loss computation by applying a binary mask
  - Benefits:
    - Prevents the model from wasting capacity on learning padding patterns
    - Makes the loss value more meaningful (only reflects actual content)
    - Allows variable-length sequences in batches without biasing training
    - Improves training stability by focusing gradients on real tokens
3. **Categorical Cross-entropy**
  - Standard choice for sequence generation tasks
  - Applied only to non-padding tokens through masking

Output Layer Design

1. **Temperature Scaling (T=0.7)**
   - Applied to logits before softmax: logits = logits / temperature
   - Benefits:
     - Controls the "sharpness" of probability distribution
     - Lower temperature (< 1.0):
       * Makes distribution more peaked/confident
       * Favors high-probability tokens
       * Reduces randomness in generation
       * Better for factual QA where precision is important
     - Higher temperature (> 1.0):
       * Makes distribution more uniform
       * Increases diversity in generation
       * More suitable for creative tasks
   - Current value (0.7) chosen to balance:
     - Confidence in factual answers
     - Avoiding overly deterministic generation
     - Maintaining some diversity in valid alternative answers

Evaluation Metric

- Custom masked accuracy
- Only considers non-padding tokens
- Computes token-level accuracy

Optimization

- Adam optimizer with custom learning rate schedule
- Learning rate value: 1e-4
- Gradient clipping at norm 1.0
- Beta values: =0.9, =0.98
- Epsilon: 1e-9

Generation Strategy

- Temperature-controlled nucleus sampling
- Parameters:
  - Temperature: 0.6 (reduced from standard 1.0)
  - Top-k: 20 tokens
  - Top-p: 0.9 (nucleus sampling threshold)

Teacher Forcing

- During training, the decoder uses ground truth tokens as input for next token prediction
- Benefits:
  - Faster convergence in early training
  - More stable gradients
  - Parallel training (all decoder outputs can be computed simultaneously)
- Limitations:
  - Creates exposure bias: model never sees its own predictions during training
  - Can lead to error accumulation during inference
  - Explains discrepancy between training metrics and inference quality

Training Results

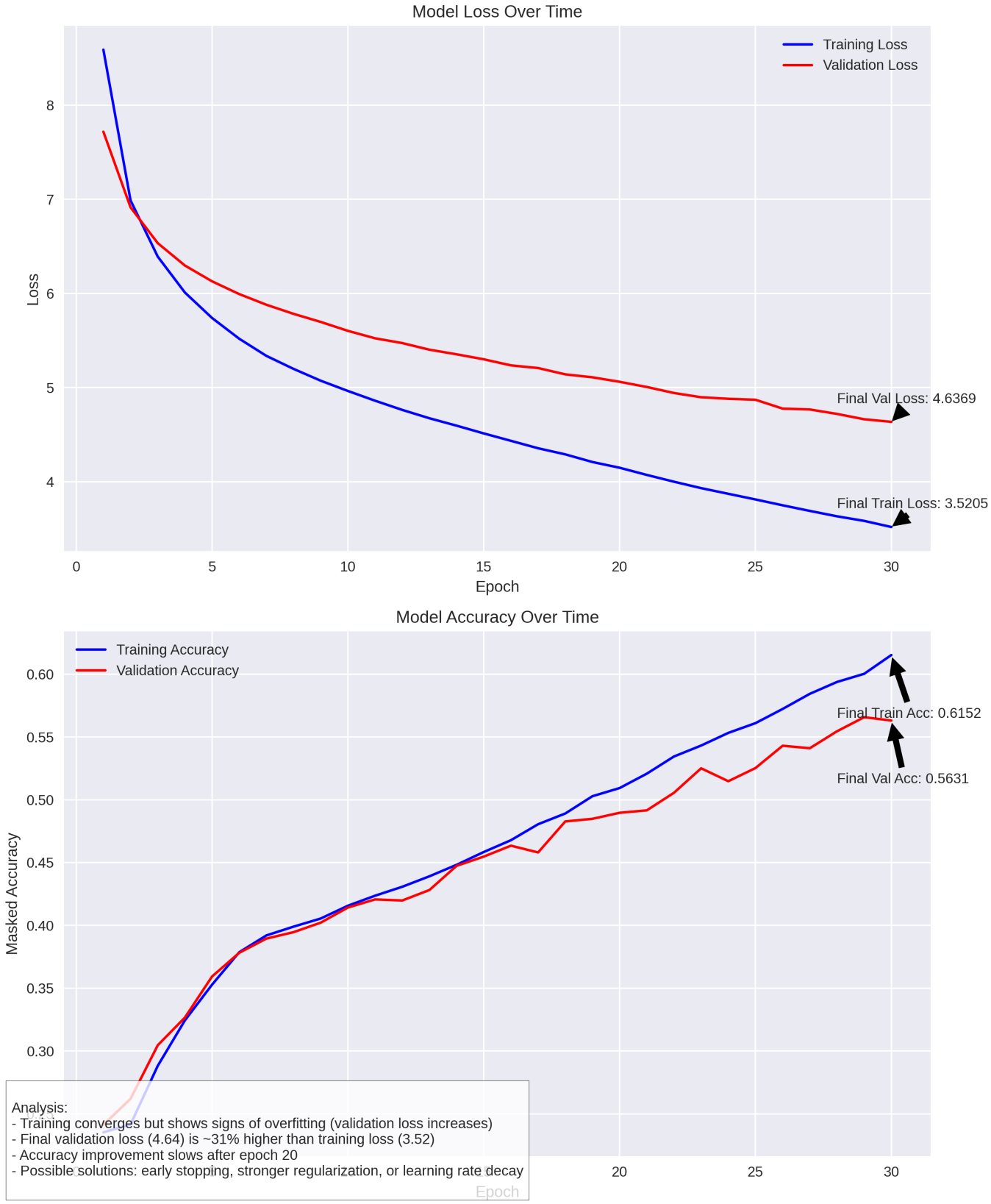

Figure 1: Training History

Final metrics after 30 epochs: - Training: - Loss: 3.5249 - Masked Accuracy: 61.83% - Validation: - Loss: 4.6369 - Masked Accuracy: 56.31%

Analysis: - Gap between training and validation metrics indicates some overfitting - Validation loss higher than training loss by ~31% - Token-level accuracy of 56.31% on validation set - Note: These are token-level metrics under teacher forcing - Real performance during inference may be lower due to: - Exposure bias from teacher forcing - Error accumulation in autoregressive generation - No teacher forcing during inference - Explains why model might struggle with longer sequences or exact number generation

4. Limitations and Potential Improvements

Current Limitations

1. **Training Data**
   - Limited to 10,000 examples
   - May not cover diverse question types
   - Potential bias in answer distribution
2. **Architecture**
   - Fixed maximum lengths may truncate important information
   - Simple concatenation of context and question
   - No explicit handling of numerical values
3. **Generation**
   - Current sampling approach may not be optimal for factual QA
   - Difficulty with exact number extraction

Suggested Improvements

1. **Data and Preprocessing**
   - Use full SQuAD dataset
   - Add data augmentation
   - Implement sliding window for long contexts
   - Better number tokenization
2. **Model Architecture**
   - Incorporate pre-trained language models (e.g., BERT)