NLP Milestone 2

Report

Nayera Mahran 52-26764
Mariam Wael 52-1647
Khadiga Yehia 52-1145

1. Overview

In this milestone, we aimed to build a shallow neural network model to solve an information retrieval and question-answering task using the SQuAD dataset. The goal was to enable the model to extract relevant answers from a given context in response to a question.

2. Dataset Description

We used a subset of the Stanford Question Answering Dataset (SQuAD), which consists of context paragraphs from Wikipedia articles and corresponding question-answer pairs. The dataset was filtered to include **10,000 rows** .

Each data point includes:

- A **context** paragraph
- A **question**
- The **answer text**
- The **start index** of the answer in the context

3. Preprocessing

3.1 Dataset Loading and Cleaning

- The **SQuAD** dataset was loaded using the Hugging Face `datasets` library.
- From the dataset, only the `context`, `question`, and `answers` fields were extracted.
- Answer text and start position were parsed, and samples with invalid or missing answers were removed.
- An `answer_end` column was computed for easier span alignment.

3.2 Subset Selection

- Contexts were sorted by length to improve tokenization efficiency.
- The **shortest 10,000** samples were selected for training and validation.

3.3 Input Formatting

- Each sample was formatted as a **combined sequence**:
  `question + ' [SEP] ' + context`
  This helped distinguish question tokens from context tokens during tokenization.

3.4 Tokenization

- A `TextVectorization` layer was used to tokenize and convert text to integer sequences with:
    - Max vocabulary size: 15,000 tokens
    - Sequence length: 125 tokens
        - 125 tokens were chosen based on the longest length of tokens where the context is concatenated to the question.
        - 125 is used also in padding so if the total length of the input is less than 125 so the rest index will be padded with 0's.

3.5 Mapping char index to token index

- The answer's character positions were converted to token indices using a certain implemented logic:
    - The answer start and end positions were mapped to their corresponding token indices within the combined sequence.

4. Model Architecture

In this milestone, a shallow Transformer-based neural network was developed to perform extractive question answering on the SQuAD dataset. The model takes a tokenized input of `question + [SEP] + context` and predicts the start and end token positions of the answer.

4.1 Input

- **Input Shape**: Sequences of length 125.
- **Input Tokens**: Integer-encoded sequences using a vocabulary of 15,000 tokens.

4.2 Embedding + Positional Encoding

- An `Embedding` layer maps input token IDs to dense vectors.
- A custom **Positional Encoding** layer is added to inject token position information. It uses a learnable embedding for each position and adds it to the token embeddings.

4.3 Transformer Encoder Block

A single encoder block was built with:

- **Multi-head Self-Attention** (`8 heads`, `key_dim=256`)
- **Feed-forward Network**:
    - Dense layer with 2048 units and ReLU activation
    - Followed by another Dense layer to project back to 256 dimensions
- **Layer Normalization** and **Dropout** (0.1) are applied for stability and regularization.

4.4 Output Layers

- Two parallel dense layers are used to predict the **start** and **end** token positions:
    - Each produces a vector of size `(batch_size, sequence_length)`
    - `Softmax` activation is used to convert the logits to probability distributions over the token positions

4.5 Compilation and Training

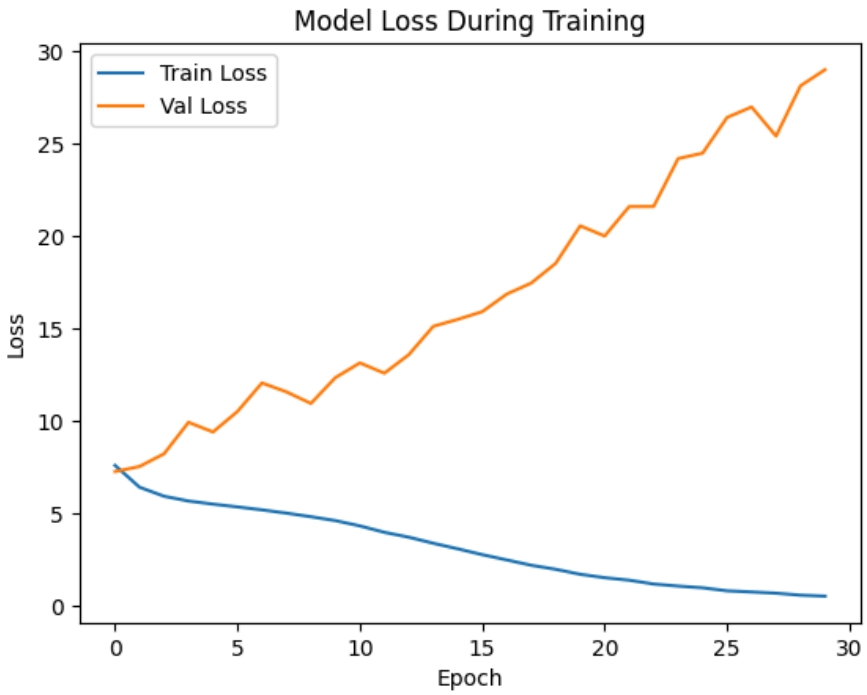
- **Loss Function**: `sparse_categorical_crossentropy` for both outputs.
- **Optimizer**: `Adam`
- **Metrics**: `accuracy` is tracked for both start and end predictions.
- **Training Parameters**:
    - Epochs: 30
    - Batch Size: 64
    - Validation split: 15% of the data

4.6 Evaluation

- The model was evaluated on the validation set after training.
- Final **validation loss and accuracy** values were printed for both the start and end position predictions.

5. Loss Function

We used sparse categorical cross-entropy as a loss function. We calculated the loss for the start and the end tokens. Then we plotted the loss for the training and validation.



6. Evaluation metrics

1. Accuracy: Measures how often the predicted start and end token positions exactly match the ground truth.
   Why did we use it?
   Provides insight into how well the model is identifying the boundaries of the answer span.
2. Exact Match (EM) Accuracy: The percentage of predictions where both the start and end token positions match exactly with the true answer.
   Why did we use it?
   EM is a strict and standard metric in QA benchmarks like SQuAD. It tells us how often the model's predicted answer is exactly correct.
3. F1-Score: mean of precision and recall, calculated based on the overlap between the predicted and actual answer.
   Why did we use it?
   Useful when the predicted answer is partially correct. It accounts for how much of the model's answer intersects with the true answer.

```
Evaluation Results:
Start Token Accuracy: 5.60%
End Token Accuracy:   6.80%
Exact Match Accuracy: 2.87%
Average F1 Score:     6.06%
```

7. Hyperparameters and optimization

- Optimizer: The model uses the Adam optimizer with a learning rate of 3e-4, which is a reasonable choice for most deep learning tasks.
- Batch Size: A batch size of 64 is typical, we tried many batch sizes and this was the good one.
- Epochs: Training for 30 epochs.

8. Post Processing

After the model predicts the start and end token positions for the answer,
post-processing is applied to improve the final extracted answer from the context. This helps correct mismatches between the predicted answer and the actual answer text.

1. **Punctuation Removal:**
   If the extracted answer ends with punctuation (e.g., a comma, period, or semicolon), it is removed to clean the result.
2. **Possessive Form Removal:**
   If the answer ends with `'s` (indicating a possessive), this suffix is removed if the resulting word (without `'s`) is found in the context.

9. Limitations:

1. Our network should be deep to extract more features.
2. We need to use a large dataset, so the model have enough different data to learn from.
3. Our laptops can’t handle running this large model.
4. we're using two transformer layers with 4 attention heads. That’s not enough capacity to model long contexts or complex dependencies.
5. Only 1 Transformer Encoder block is used with a single attention layer that is not deep enough to model complex interactions between question and context. State-of-the-art models like BERT use 12–24 layers.
6. Vocab size = 15000 is small for SQuAd dataset so OOV rate will be high.
7. Not using sinusoidal or global position-aware encodings, hurts generalization to unseen sequence lengths.
8. The model uses sparse categorical cross-entropy for both the start and end token predictions. While this is a standard choice for token classification, it does not fully capture the sequential nature of the problem. A better approach would be to use Span-based Loss that takes into account the entire span (start to end) rather than treating each token independently.

9. Conclusion:

In this project, we successfully built a Transformer-based question answering model trained on the SQuAD dataset. The model utilized key components such as an embedding layer with positional encoding, multi-head attention, and feed-forward networks to learn contextual relationships between question and context tokens.

We evaluated the model using multiple metrics: start and end token accuracy, exact match (EM), and F1 score. These metrics provided a comprehensive view of the model’s performance, highlighting both its precision in identifying exact answer spans and its ability to approximate partially correct answers.