Project 2: Sequence-to-Sequence Transformer for Sentence Recovery

By Danny Rechitsky

---

1. Introduction

The objective of this project is to build a sequence-to-sequence model capable of recovering original, lower-cased sentences (`tgt`) from preprocessed versions (`src`). The preprocessing steps applied to the source text involve lemmatization and the removal of determiners, conjunctions, and prepositions. The model aims to reverse this transformation by learning to insert the missing words and correct verb forms.

The following solution utilizes the **Transformer encoder-decoder architecture**. We investigate three key experimental dimensions: positional encoding method, decoding strategy, and architectural depth, to determine the optimal configuration for this sentence recovery task.

---

2. Methods

2.1 Model Architecture

The core of the model is the standard **Transformer**, featuring an **Encoder** for processing the source sequence and a **Decoder** for generating the target sequence autoregressively.

- **Attention Mechanisms:** The model utilizes the **Scaled Dot-Product Attention** mechanism.
    - **Padding Masks:** Used by the Encoder to prevent attending to $<\text{pad}>$ tokens in the source sequence.
    - **Causal Masks:** Used in the Decoder's self-attention to ensure a token at position i only attends to positions 0 through $i-1$, preserving the autoregressive property.

2.2 Data Preparation and Training Setup

- **Tokenizer:** A **shared vocabulary** is built using all tokens from the training data via a word-level tokenizer. Special tokens ($<\text{bos}>$, $<\text{eos}>$, $<\text{pad}>$, $<\text{unk}>$) are included and receive unique IDs.
- **Dataset:** The `SeqPairDataset` prepares sequence triplets for training:
  - **Encoder Input** (Source IDs with $<\text{bos}>/<\text{eos}>$).
  - **Decoder Input** (Target IDs shifted right, for teacher forcing).
  - **Labels** (Target IDs shifted left, used as ground truth for prediction).
- **Loss Function:** CrossEntropyLoss is used as the criterion, initialized with ignore_index set to the $<\text{pad}>$ ID to exclude padding from loss calculation.
- **Optimizer:** The **Adam optimizer** is used to update model parameters.
- **Hyperparameters (Baseline):**

Hyperparameter	Value
Epochs	3
Learning Rate (lr)	0.001
Feedforward Dimension (d_{ff})	256
Batch Size	256
Enc/Dec Layers	1
Model Dimension (d_{model})	128
Number of Attention Heads (num_heads)	2
Dropout	0.1
Max src/tgt sequenc length	50

---

3. Results

3.1 Experiment 1: Positional Encoding Strategies

Metric	Sinusoidal PosEncoding (Baseline)	Learnable PosEmbedding
Trainable Parameters	1,033,238	1,058,838
Final Test BLEU	0.749	0.815

- **Figure:** Plot of Training and Validation Loss Curves for both models.
- **Example Output:**

Ground Truth (Target)	Baseline (Sinusoidal) Prediction	Learnable (PosEnc) Prediction
`the other was a very drunk and very wealthy english gentleman .`	`the other was very worse very the english english gentleman .` (Shows incorrect word insertion, repetition)	`the other was very drunk very wealthy english gentleman .` (More fluent, but still misses articles 'a' and 'and')
`41 : 5 mine enemies speak evil of me , when shall he die , and his name perish ?`	`41 : 5 mine enemies speak the evil of me , when he shall die , and his name , and his name ?` (Includes extraneous words and repeats the final phrase)	`41 : 5 mine enemies speak evil of me , when shall he die , and his name perish ?` (**Perfect Match**)
`19 : 15 and hezekiah prayed before the lord , and said , o lord god of israel , which dwellest between the cherubims , thou art the god , even thou alone , of all the kingdoms of the earth ; thou hast made heaven and earth`	`19 : 15 hezekiah pray before the lord , saying , o lord god of israel , which between the counsel between the cherubims , and thou art god , even thou alone , and all the kingdom of earth ; and hast made the earth : and` (Severe lack of fluency, many incorrect insertions and missing conjunctions)	`19 : 15 and hezekiah prayed before the lord , saying , o lord god of israel , which dwellest between the cherubims , thou art god , even thou alone , of all the kingdom of earth ; hast made heaven earth .` (Significantly better fluency and structure)
`who , if we knew what we receive , would either no accept life offered , or soon beg to lay it down ; glad to be so dismissed in peace .`	`who , if we know what we received , would accept , and accept the life , soon , soon lay down it down ; glad was so in peace .` (Incorrect verb tense, awkward repetition of 'accept' and 'soon')	`who , if we know what we receive , would either accepted accepted life offer , soon begged it down ; glad is so dismissed in peace .` (Still has verb errors, but preserves better sentence flow)

3.2 Experiment 2: Decoding Algorithms

Best Parameters Greedy Decoding : model depth = 4 (other parameters same as baseline)

Final Test BLEU : 0.749

Example Output:

Ground Truth (Actual)	Predicted Output
`the other was a very drunk and very wealthy english gentleman .`	`the other was very drunk very whole gentleman .`
`41 : 5 mine enemies speak evil of me , when shall he die , and his name perish ?`	`41 : 5 mine enemies to the evil of me , when shall he die , and his name perish perish ?`
`19 : 15 and hezekiah prayed before the lord , and said , o lord god of israel , which dwellest between the cherubims ,`	`19 : 15 and hezekiah prayed before the lord , saying , o lord god of israel , which dwellest between the`

Ground Truth (Actual)	Predicted Output
`thou art the god , even thou alone , of all the kingdoms of the earth ; thou hast made heaven and earth`	`cherubims , thou art god , even thou alone , of all the kingdom of the earth ; and hast made the earth .`
`who , if we knew what we receive , would either no accept life offered , or soon beg to lay it down ; glad to be so dismissed in peace .`	`who , if we know what we receive , would either accept life and offer the offer , soon lay it down ; glad is so immediately in peace .`

3.3 Experiment 3: Model Architecture Variants

Experiment 3: Model Architecture Variants Comparison

Configuration	Enc/Dec Layers	Attention Heads	Model Dim (d_{model})	Final Val Loss (Epoch 2)	Test BLEU Score
Baseline	1	2	128	1.7231	0.7495
4 Heads	1	4	128	1.7620	0.7509
8 Heads	1	8	128	1.7899	0.7488
2 Layers	2	2	128	1.2402	0.7905
4 Layers	4	2	128	**1.0778**	**0.8076**

Figure: Plot of Validation Loss Curves for the different architectural variants.

---

4. Analysis

4.1 Model Architectural Variants (Experiment 3)

Two primary architectural modifications were tested: increasing the number of attention heads and increasing the model's depth (number of encoder/decoder layers).

Impact of Multi-Head Attention (num_heads)

- **Learning and Generation Quality:** Increasing the number of attention heads (from 2 to 4 and 8) only yielded marginally better validation loss rates and final BLEU scores. The effect in this case of multi-headed attention on learning and generation quality appear quite negligible.

Impact of Model Depth (num_enc_layers / num_dec_layers)

- **Performance Improvement:** Deeper architecture did noticeably improve performance. Comparing the baseline (1 layer, BLEU score of 0.7495) to the deeper configurations (2 layers and 4 layers), performance increased significantly, culminating in the highest BLEU score for the 4-layer model. The **Baseline (1 Layer)** model achieved a final Validation Loss of **1.7231** and a Test BLEU score of **0.7495**. Increasing the depth to **2 Layers** dropped the loss to **1.2402** and boosted the BLEU score to **0.7905**. The **4 Layers** configuration provided the best result, achieving the lowest loss of **1.0778** and the highest BLEU score of **0.8076**, confirming that increased depth was the most critical scaling factor.
- **Trade-offs:** The trade-off for the better performance of deeper architecture is increased computational expense and potential inefficiency. Increasing model depth dramatically increases the number of trainable parameters, leading to:
    1. **Longer Training Time:** More parameters require more computation time per epoch.
    2. **Higher Memory Usage:** The increased number of layers requires more memory to store parameters and intermediate activations.

4.2 Positional Encoding Strategies (Experiment 1)

Experiment 1 compared the **Sinusoidal PosEncoding** (fixed, non-trainable baseline) against **Learnable PosEmbeddings** (trainable parameters). The results demonstrated a clear performance advantage when the positional information was learned by the model.

- **Sinusoidal Baseline:** This configuration achieved a final validation loss of **1.7231** and a Test BLEU score of **0.7495**. While it provided necessary positional context, the fixed, mathematical pattern restricted the model's ability to optimize how position influenced word meaning.
- **Learnable Embeddings:** This configuration significantly outperformed the baseline, finishing with a validation loss of **0.9204** and a Test BLEU score of **0.8159**. The added capacity of the learnable embedding layer allowed the model to fine-tune the positional vectors specifically for the characteristics of the sentence recovery task, resulting in faster convergence and superior generalization.

4.3 Patterns in Model Capacity and Performance

- **Overall Relationship:** In this case, whether marginal (attention heads) or significant (model depth), **greater model capacity has yielded better performance**. This is generally expected, as increasing the number of parameters allows the model to better predict the output even of complicated systems.

- **Which Configuration Works Best:** The configuration with **4 encoder and 4 decoder layers** yielded the best overall performance, demonstrating that **depth** was the most crucial architectural dimension to scale for this sentence recovery task.

4.4 Diminishing Returns and Overfitting

- **Overfitting Risk:** Greater model complexity may lead to **overfitting**. If a model has great depth or height (complexity), it might be too large if trained on a small number of data samples. It will then learn the noise and idiosyncrasies of the training data, leading to poor generalization on unseen data.
- **Diminishing Returns:** Diminishing returns were not seen. In fact, performance still improved up to 4 layers. It is, nevertheless, an expected trade-off. At some point, increasing the complexity further (e.g., 8 layers) will only result in minor performance gains that do not justify the dramatically increased training time, memory cost, and risk of overfitting.

---

5. Conclusion

Summary of Findings

The experiments successfully demonstrated the critical role of architectural scaling in achieving high performance for the sentence recovery task.

- **Optimal Configuration:** The deepest model, utilizing **4 Encoder and 4 Decoder layers** ($\text{num\_enc\_layers} = 4$, $\text{num\_dec\_layers} = 4$), proved to be the most effective, yielding the lowest validation loss and highest final BLEU score.
- **Positional Encoding:** The **Learnable Positional Embeddings** significantly outperformed the Sinusoidal baseline, suggesting that training positional knowledge is a more effective strategy than relying on fixed trigonometric functions.
- **Capacity vs. Width:** Increasing the model's **depth** (layers) resulted in substantial, non-diminishing returns on performance, whereas increasing **width** (attention heads) in the shallow baseline model had a negligible effect.

Limitations

Despite achieving strong results, the current implementation has several limitations that constrain peak performance and efficiency:

- **Tokenizer Simplicity:** The use of a simple word-level tokenizer restricts the model's ability to handle morphology (word parts) and manage out-of-vocabulary ($<\text{unk}>$) words effectively.
- **Hyperparameters:** Only two of the hyperparamers were experimented with (model depth and width). Doing a full hyperparameter sweep would yield the best possible configuration.

Future Improvements

Based on the limitations and experimental results, future work should focus on the following areas:

- **Subword Tokenization:** Implementing a more advanced tokenization scheme (e.g., BPE or WordPiece) would reduce the size of the overall vocabulary and improve generalization on rare words.
- **Exploration of Advanced Optimizers:** Implementing learning rate scheduling (e.g., warm-up and decay) or alternative optimizers beyond basic Adam could further accelerate convergence and improve generalization.