Assignment 1: Transformer from scratch

Course : Advanced NLP

Deadline: September 10th, 2025 — 23:59

General Instructions

1. The assignment must be implemented in Python.
2. You are allowed to use AI tools or any internet resources as long as your implementation is in accordance with the assignment guidelines.
3. A single `.zip` file needs to be uploaded to the Moodle Course Portal.
4. Please start early since no extension to the announced deadline would be possible.
5. **Do not use the readily available torch modules for positional encoding, encoder stack, decoder stack, multi-head attention or any other part of the transformer architecture. You are expected to implement them from scratch.**

1 Transformers for Machine Translation

Machine translation, the task of automatically translating text from one language to another, has significantly evolved with the advent of neural networkbased approaches. Traditional machine translation systems, such as rule-based or statistical approaches, faced challenges in capturing complex linguistic patterns and contextual dependencies. The introduction of neural networks revolutionized this field by leveraging their ability to learn intricate relationships directly from data. Sequence to sequence models, initially introduced for various natural language processing (NLP) tasks, were adapted for machine translation due to their inherent suitability for handling sequential data.

The transformer architecture, introduced in the paper "Attention is All You Need" by Vaswani et al. in 2017, emerged as a groundbreaking innovation in sequence to sequence modelling. Transformers rely heavily on self-attention mechanisms, enabling them to weigh the importance of each word in the input sequence dynamically. This attention mechanism is what allows the model to capture long-range dependencies and contextual information effectively. The attention mechanism aims to assign weights to different positions in the input sequence, and this allows the model to focus on relevant parts during the encoding and decoding processes. By attending to specific words or tokens based on their importance scores, the model creates a context-aware representation of the input sequence, which is crucial for high-quality translations.

Transformers are trained using parallel datasets, where source sentences and their corresponding target translations are aligned. The model learns to minimize a loss function, often based on maximum likelihood estimation or other variants, to optimize the translation performance.

In this assignment, we will delve into the practical implementation of a simple transformer for machine translation, understanding its components and applying the theoretical concepts discussed above.

2 Build the Transformer

Implement an encoder–decoder Transformer architecture **from scratch** in PyTorch. Your model should include:

(i) **Core components:**
Your model should include all components of the original transformer architecture. You are not allowed to use any readily available Torch modules related to transformers, such as the decoder module, encoder module, etc.

(ii) **Positional Encodings**:
You have to implement two types of positional encoding strategies.

(a) Rotary positional embeddings (RoPE).
(b) Relative position *bias* (additive bias to attention scores).

The code should allow switching between RoPE and relative position bias via configuration.

(iii) **Training:**
You are provided with a corpus of parallel Finnish and English sentences. You must implement a full training loop with teacher forcing to train your transformer on Finnish to English machine translation. Save model checkpoints and log training curves

3 Implementing Decoding Strategies

Implement and evaluate the following decoding algorithms for inference:

(i) **Greedy decoding** – always select the token with the highest probability at each step.

(ii) **Beam search** – maintain the top-B sequences (beams) at each step and choose the sequence (or "beam") with the highest overall score.

(iii) **Top-k sampling** – sample the next token from the top-k most probable tokens.

Your code should allow easy switching between decoding strategies via a command-line argument or configuration file.
Analyze and compare the outputs of the three decoding strategies and report your observations.

4 Dataset

The dataset can be downloaded here. Remember to split the dataset into training, validation and test datasets.

5 Report and Analysis

In your report, you must include a comparative analysis based on the following aspects:

1. **Translation accuracy:** Evaluate all configurations on the test set and report their final BLEU scores in a table. Analyze the impact of different decoding strategies on translation accuracy and support your observations.

2. **Convergence Speed:** Plot the training loss against epochs for both positional encoding methods on the same graph. Indicate which configuration converges faster and support your observations.

6 Deliverables

A single `<rollnumber>_assignment1.zip` containing the following files:

1. `encoder.py` – Implement the encoder class and relevant classes/functions.
2. `decoder.py` – Implement the decoder class and relevant classes/functions.
3. `train.py` – Main code to train the transformer for machine translation.
4. `test.py` – Code to test the pre-trained model on the test set. Load the model saved during training.
5. `utils.py` – Include any other helper functions or classes.
6. `report.pdf` – Your analysis of the results and observations.
7. `README.md` – Details for running the code, along with a link to the pretrained Transformer model (`.pt` file).

7 Grading

Evaluation will be individual and will be based on your viva, report and submitted code review. You are expected to walk us through your code, explain your experiments, and report. You will primarily be graded based on your conceptual understanding.

1. **Positional encodings** – 10
2. **Decoding Strategies** – 15
3. **Correctness of architecture** – 10
4. **Report and Analysis** – 5
5. **Viva** – 50

8 Resources

1. Understanding Decoding Strategies
2. You can also refer to other resources, including lecture slides!