Experiment No 5

Aim: Machine Translation Using Transformer Models.

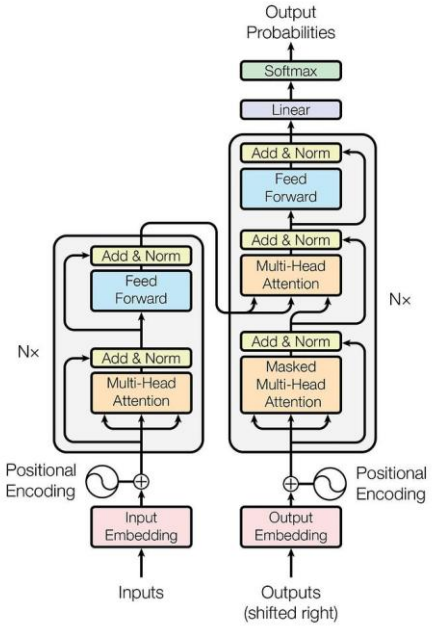
Theory:

Introduction

The Transformer model, introduced in the paper "Attention is All You Need" (Vaswani et al., 2017), revolutionized the field of natural language processing (NLP) by eliminating the need for recurrent neural networks (RNNs) and convolutional networks for sequence modelling tasks. Instead, the Transformer relies entirely on a mechanism called self-attention to process and relate elements in a sequence.

Key Components of the Transformer:

1. **Self-Attention Mechanism:**
   - The self-attention mechanism allows the model to focus on different parts of an input sequence to compute a representation of the sequence. It captures dependencies between tokens, regardless of their position in the sequence.
   - Given an input sentence, self-attention computes the relevance (attention scores) of each word with respect to every other word in the sentence.
   - For example, in the sentence "The cat sat on the mat," self-attention allows the model to understand that "the" relates to both "cat" and "mat," despite being separated by other words.
2. **Multi-Head Attention:**
   - To capture different relationships between words, the Transformer uses **multi-head attention**, where the self-attention mechanism is applied multiple times in parallel, and the outputs are concatenated. This enables the model to focus on different aspects of the sequence simultaneously.
3. **Positional Encoding:**
   - Since the Transformer does not have a built-in notion of word order (unlike RNNs), it uses **positional encodings** to inject information about the position of words in the sequence. These encodings are added to the input embedding, ensuring that the model can differentiate between words based on their position in a sentence.
4. **Encoder-Decoder Structure:**
   - The Transformer consists of an **encoder** and a **decoder**, each composed of several identical layers.
     - **Encoder:** The encoder processes the input sequence by applying self-attention and feed-forward layers. It generates a set of encoded representations.
     - **Decoder:** The decoder takes the encoder's output and the target sequence (shifted by one step) as input. It uses both self-attention and **encoder-decoder attention** (which focuses on the encoder's output) to generate the final translated output.



5. **Feed-Forward Networks:**
   - Each attention mechanism is followed by a **position-wise feed-forward network**, which consists of two fully connected layers applied independently to each position in the sequence. This helps in learning non-linear transformations of the input.
6. **Layer Normalization and Residual Connections:**
   - Each sub-layer in the encoder and decoder has a **residual connection** around it, followed by layer normalization. This helps stabilize training and allows the model to learn more effectively.

Here are the key steps to train a transformer model for **Neural Machine Translation (NMT)** from scratch:

1. Data Collection

The first and most crucial step is to gather a **large parallel dataset** consisting of sentence pairs from the source and target languages (e.g., English-German). A parallel dataset contains text in both languages that are aligned on a sentence level.

- **Open datasets**: Common parallel datasets include Europarl, TED talks, or datasets from the WMT (Workshop on Machine Translation).
- **Custom datasets**: You might also need to create your own dataset, especially if you're working with specialized domains or less common languages.

For example:

- **Source sentence (English)**: "I am learning."

- **Target sentence (German)**: "Ich lerne."

2. Data Pre-processing

Once you have the dataset, the next step is to prepare the data for training. This includes:

- **Cleaning the data**: Remove noisy or corrupted sentences, sentence pairs that are not properly aligned, etc.
- **Tokenization**: Split the sentences into smaller subword tokens. The transformer model works at the subword level rather than whole words. For example:
  - "I am learning" might be tokenized into ["I", "am", "learn", "ing"].
  - "Ich lerne" might be tokenized into ["Ich", "lern", "e"].
- **Create vocabulary**: Build a vocabulary for the source and target languages based on the tokens in your dataset. This vocabulary is used to convert tokens into numerical representations (IDs).
- **Padding and truncation**: Sentences will vary in length, so shorter sentences need to be padded to match the length of the longest sentence in a batch. Longer sentences might be truncated to a maximum length.

3. Model Architecture Setup

Now that your data is ready, you need to build the transformer architecture. The main components include:

- **Encoder**: This part of the model processes the source language sentence (e.g., English) and converts it into a sequence of hidden representations.
  - It uses multiple layers of self-attention and feed-forward networks.
- **Decoder**: The decoder generates the translated sentence (e.g., German) from the hidden representations produced by the encoder.
  - It also uses self-attention and attention mechanisms to focus on the relevant parts of the source sentence.

The transformer model typically consists of multiple layers of encoders and decoders (e.g., 6 layers each), and each layer includes:

- **Multi-head self-attention**: To allow the model to focus on different parts of the sentence.
- **Feed-forward networks**: To transform the attention outputs.

You can initialize the transformer model's weights randomly since you're training from scratch.

4. Training the Model

Once the model is set up, the training process begins. This involves multiple steps:

a) Loss Function

- The model is trained to minimize the **cross-entropy loss**, which measures how well the predicted output matches the actual target sentences. The goal is to increase the probability of correct translations and decrease the probability of incorrect ones.

b) Optimization

- You typically use optimizers like **Adam** or **Adafactor** to update the model's weights during training.
- **Learning rate schedules**: The learning rate is gradually increased in the initial stages (warm-up) and then decreased later.

c) Training Strategy

- You input batches of source sentences (e.g., English) and their corresponding target translations (e.g., German).
- For each batch:
    1. The **encoder** processes the source sentence and generates hidden representations.
    2. The **decoder** uses these hidden representations to predict the target sentence, one token at a time.
    3. The model compares its prediction with the actual target sentence, calculates the loss, and updates the weights accordingly.

d) Teacher Forcing

- During training, **teacher forcing** is used, where the correct previous token is given as input to the decoder rather than the decoder's own previous prediction. This helps the model learn more effectively, especially in the early stages of training.

5. Evaluation Metrics

Once the model is trained, it is essential to evaluate its performance.

- **BLEU Score**: The most common metric for evaluating machine translation models. It compares n-grams (consecutive sequences of tokens) in the machine-generated translation with reference translations.
- **Validation**: You should periodically evaluate the model on a validation set (a part of the dataset not used during training) to ensure it is not overfitting.

6. Hyperparameter Tuning

- **Batch size**: Adjust the number of examples processed at a time.
- **Learning rate**: Fine-tune the learning rate schedule for better performance.
- **Number of layers**: You may adjust the number of encoder and decoder layers based on the complexity of the dataset.

- **Dropout**: Use dropout regularization to avoid overfitting.

7. Model Evaluation and Fine-Tuning

After training the model, you need to test it on unseen data (test set) to evaluate how well it generalizes. You can also fine-tune it on more specific data if needed.

8. Deploying the Model

Once the model has been trained and evaluated, you can deploy it for real-world applications:

- **Batch translation**: Translate large volumes of text efficiently.
- **Real-time translation**: Use the model in applications such as live chat translation or multilingual customer service.

Lab Assignment:

Perform machine translation using Transformers on the following dataset.

cfilt/iitb-english-hindi · Datasets at Hugging Face