Notes on Natural Language Processing with Transformers

Benjamin Santana

January 2024

1 Introduction

A Recurrent Neural Network is used when working with sequence as your dataset. A sequence is simply a set of data where one data item depends on the item that precedes or follow it. Say for example speech, the very meaning of this sentence you are reading is defined by the order in which the words are organized.

Modeling sequences involves maintaining a hidden state. When the model *process* each item (for example a word in a sentence) the hidden state is updated. This hidden state represents everything seen by the sequence so far. We usually represent this hidden state as a vector. This allows an RNN to keep track of information from previous steps and use it for outputs predictions.

We are going to focus on a popular architecture on machine translation systems, where we want to map one a sequence of words into another language. Here a encoder-decoder architecture is used. The goal of this architecture is to encode the input into a numerical representation called *last hidden state* and then pass it to a decoder which generates the output sequence.

This approach,although elegant, has an issue in this *last hidden state*. It has to represent the meaning of the whole input, because it is all of what the decoder is going to see. If the sequence is too long we can start seeing a problem.

To solve this issue, we can allow the decoder to have access to the encoder's hidden states. The way we do this is with a method called *attention*.

The main idea behind *attention* is that instead of producing just one hidden state for the whole input, the encoder outputs a hidden state at each step and the decoder can access it whenever it wants. Not only that, but it will have a mechanism that prioritize which hidden state is more valuable than others. Giving it more weight on the output, or putting it in other words. It can give more *attention* to different hidden states than others. Unfortunately, due to the nature of the process, it can not be paralleled. It is sequential.

This is finally where the *transformer* comes into place. It basically removes the whole sequential thing, and does something called *self-attention*. What does that mean? Imagine you're reading a sentence. With self-attention, you don't read each word one after the other; you can pay attention to all words at once, giving more weight to important ones. We will talk more about self-attention later, but for now it is important to understand that this makes it way more efficient.

1.1 Transferring Learning in NLP

In computer's vision, say a CNN (convolutional neural networks) we can transfer the learning from a network trained in a large-scale dataset into a more specific one with the purpose of training networks that solve more complex tasks. What this means is that the models are trained on huge datasets that contain millions of images, we call this process *pretraining*. After that we go to a *fine tuning* stage where we now train the model with a more specifc smaller set of data.

For example, say we want to train a network to identifying types of flowers, we would use a pretrain model that already learnt the basic features of images, such as edges colors and so on. Then the fine tuning we would classify more specifc flower things.

Why are you telling us about all this computer vision stuff? I just want to know how to make a chatgpt. Well, hold your horses, I am telling you all this because we someone smart at OpenAI discovered how we could apply this to a RNN! A framework *ULMFiT* explains how to do this transfer learning stuff in a LSTM. It involves 3 main steps:

1. *Pretraining* this means predicting the next word based on the previous words. This task is referred to as language modeling. The elegance of this approach lies in the fact that no labeled data is required

2. *Domain Adaptation* We adapt it to the in-domain corpus

3. *Fine-Tuning* Fine tuned with a classification layer

After this was developed, 2 transformers were deployed based on the self-attention and transfer learning approach. The famous *GPT* and *BERT*