Recent Advancements in NLP

Attention, Transformer, BERT and more.

Moiz Saifee

Dec 26 · 8 min read ★

In a previous post, I wrote about two recent important concepts in NLP — *Word Embedding* and *RNN*. In this post, I'll cover the concept of *Attention* and *Transformer* which have become the building blocks for most of the state of the art models in NLP at present. I'll also review BERT which made the powerful concept of *transfer learning* easier *in* NLP

Like my earlier post, I'll skip most of the mathematics and focus more on intuitive understanding. The reason for this is — use of excessive notations and equations is a turn off for many, myself included. That's actually not even critical for conceptual understanding which I believe is a lot more important than understanding each and every underlying mathematical equation.

While variants of RNN like Bi-LSTM give a pretty solid performance on various NLP tasks, following are some of the key challenges that still remain:

1. **Lack of parallelization hurts performance** — LSTM needs to process the input tokens in a given sequence in a sequential manner as the output of the current step depends on the output of the previous step. There is no way to parallelize this computation, which comes with obvious disadvantages of long training time, not being able to train on very large datasets etc.
2. **Learning long term dependency remains a challenge** —While LSTMs are better at remembering context from something which appeared a while back in a sequence compared to classic RNNs, remembering the context from a word which occurred much earlier in long sentences remains a challenges, that's why performance is not as good in longer sentences/sequences compared to shorter ones.

3. **Linear increase in # operations with distance —**The number of operations required to relate signals from two arbitrary input or output positions grows linearly in the distance between positions

The above challenges were the motivation for a lot of work which happened in 2017 and forward. The concept of *attention* addresses challenge #2 above and the *transformer* architecture addresses challenge # 1 and #3.

One thing to note is — while the concepts of *attention* and *transformer* helped break established benchmarks in NLP tasks, these are generic techniques with wide application in any sequence to sequence task. For instance, the *transformer* architecture was also a building block for *AlphaStar* the *DeepMind* bot that beat the top *StarCraft II* professional player.

Attention

The *attention* is one of the most influential ideas in Deep Learning in general. While the concept was originally developed for machine translation, its use quickly spread to a lot of other areas. It was proposed by *Dzmitry Bahdanau* et al in this influential paper.

The core idea behind it is — when performing a certain task, for example, translation of a sentence from one language to the other, say English to French, each word in the output French sentence would be informed by (relevant context of) all words in the original input English sentence with varying degree of attention or importance rather than a single/constant context generated by processing the whole English sentence.

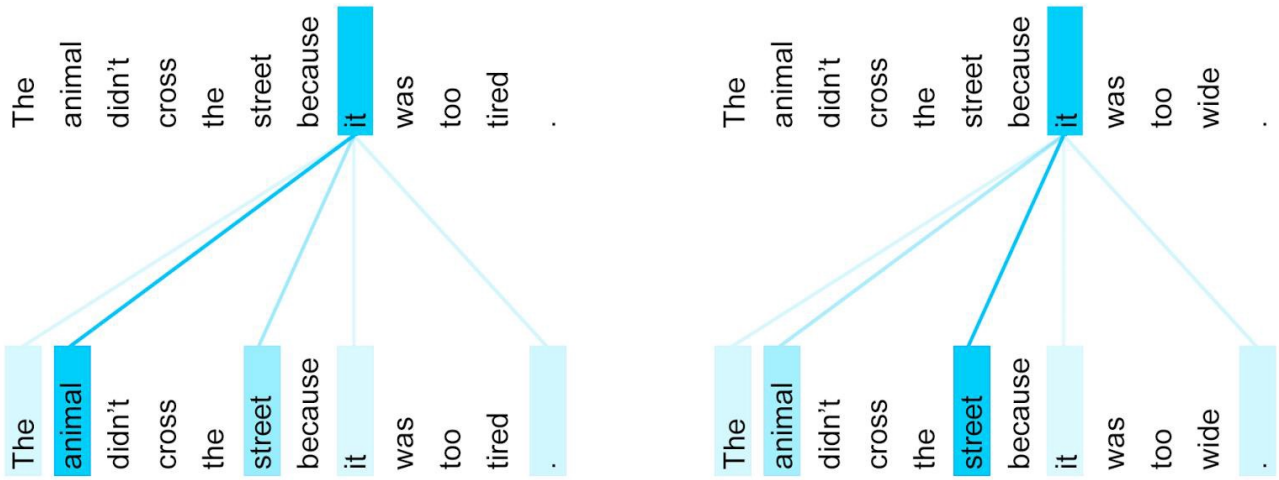
Consider the following excellent English to French translation example from *Google Blog*. The "it" in the two sentences refers to different nouns, and its translation to French differs depending on which noun the "it" refers to.

The animal didn't cross the street because it was too tired.
L'animal n'a pas traversé la rue parce qu'il était trop fatigué.

The animal didn't cross the street because it was too wide.
L'animal n'a pas traversé la rue parce qu'elle était trop large.

To a human, It is obvious that in the first sentence pair "it" refers to the animal, and in the second to the street

The following picture depicts what happens conceptually with (self) *attention*. Rather than taking a context vector of the whole sentence and that informing the translation of the word “it”, the translation context would be informed by different words in the input sentence by a different amount as shown by the color coding (darker = more important)



Source: Google Blog

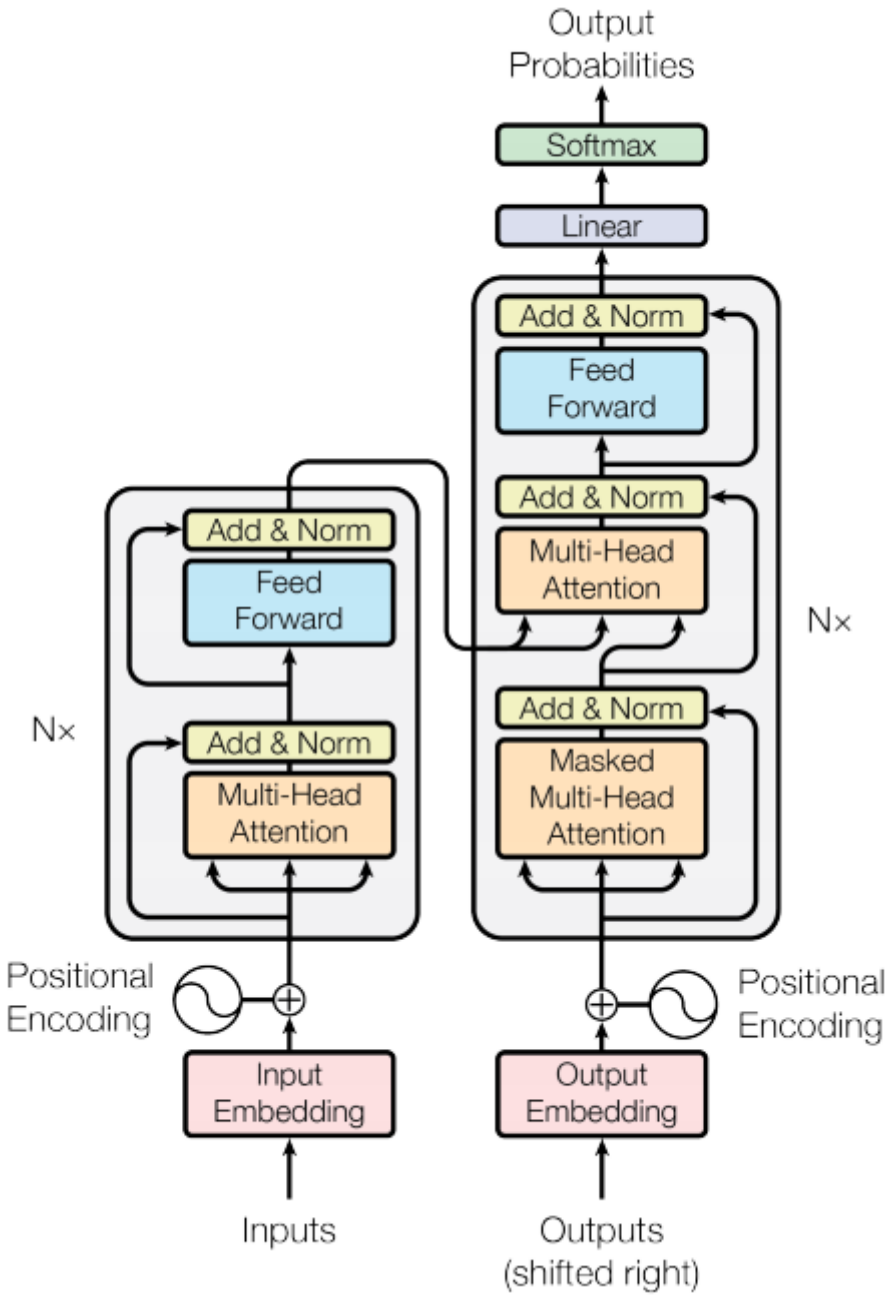
One important thing to note is — *attention* weights in the model are not fixed, like the other parameters which are learned by neural network models, but are calculated as a function of input / hidden states — that’s how the model knows how to put appropriate emphasis on relevant words in the input sentence for any given sentence.

As illustrated above, the concept of *attention* addresses the problem of long term dependencies by using more appropriate context at each step (rather than a “constant” context, example in LSTM based encoder-decoder), but the problem of non parallelism in computation remains as the computation still needs to be done sequentially. With *attention*, if anything, we make the computation even more complex compared to simple LSTMs, this is where the *Transformer* architecture came to the rescue.

Transformer

Transformer was introduced in 2017 in this seminal paper by Vaswani et al from Google. The network architecture of *Transformer* is solely based on *attention* mechanism and has no RNN or CNN units. Following are some of the key benefits of this architecture:

- Superior quality results on language translation tasks, beating previous benchmarks
- More parallelizable and requires significantly less time to train
- Generalizes well to a lot of other tasks



Transformer Model Architecture. Source Transformer Paper

The figure above shows the overall architecture of the *Transformer*, which is a bit scary at first blush. Let's look at high level details of the architecture:

- **Encoder-Decoder Architecture**: It follows the famous Encoder Decoder paradigm, where Encoder translates the input into a useful representation which is used by

decoder to generate the output. In the figure above, the Encoder block is on the left and Decoder block is on the right.

- **Auto Regressive**: At each step, the model consumes the previously generated output as an additional input
- **No RNNs**: There are no RNN units, only a bunch of attention layers.
- **Multiple layers**: Each of Encoder / Decoder consist of a stack of 6 (a hyperparameter in the model) identical layers
- **Embedding**: The Input / Output embedding converts text or words into numbers.
- **Positional Encoding**: The positional encoding is to encode the order of the original sequence as we no longer process the input sequentially like in RNN, so it ensures the information of ordering is not lost.
- **Mult-Head attention**: There are multiple attention layers running in parallel, for adding variety (akin to multiple convolution filters in Computer Vision).
- **Attention layers**: There are 3 types of attentions — Encoder self-attention, Encoder-Decoder Attention and Decoder self-attention.
- **Self Attention**: In case of Encoder self-attention (left-bottom attention block in figure), a layer within the Encoder tries to figure out how much attention it should pay to the output of the previous layer(of the Encoder), that is, tries to learn the relationship /context for each word in the input with the other words in the input. Decoder self-attention (right-bottom attention block in the figure)does the same thing but over Output.
- **Encoder-Decoder Attention**: With Encoder-Decoder attention (right-top attention block in the figure) The model figures out how relevant is each word in the input for each word in the output.

With those details, the results which *Transformer* achieved were impressive, however, there were some areas of improvement:

- It can only deal with fixed-length sentences.
- Larger sentences need to be broken into smaller ones for feeding into the model, which causes "*context fragmentation*"

BERT — Bidirectional Encoder Representations from Transformers

First, an honest confession — I just can't memorize the full form for BERT no matter how many times I read it, I think the full form was derived from the acronym rather than the other way around :-)

BERT was proposed by Jacob et al from Google, its a language representation model based on *Transformer.* I think the major contribution of BERT, in addition to its novel bi-directional training, etc is that it played a big role in popularizing the concept of pre-training */ transfer learning* in NLP

Before we go further — just a few words about *transfer learning*. Its one of the most elegant concepts in Deep Learning was more popular in *ComputerVision* compared to NLP. The basic idea is — one can train a (deep, that is having multiple layers) model on a generic but related task, and just fine-tune the last few layers on their task-specific data to achieve great performance.

The rationale for *transfer learning* is — the earlier layers in a deep model learn more fundamental patterns, for example, learning to detect lines and curves in a *computer vision* model, and the latter layers learn the task-specific patters, example learning whether there is a cat in the picture or not. The major benefit of this approach is — you can get great performance even with a modest amount of data on deep learning models which are pretty data hungry in general.

Following are some important details about BERT:

1. **Bi-Directional training**: Jointly trained by looking at words in both directions — left-to-right and right-to-left for better context, other language models are typically trained by looking at context in just one direction

2. **Pre-trained**: Pre-trained on huge text corpses (~3B words), so you just need to fine-tune one additional output layer, that is very few parameters (using your training data) for getting very good performance on a variety of tasks like — sentence classification, question answering system, named entity recognition etc without having to do too many task specific changes in model architecture.

3. **Training Objective:** Typically language models are trained in an unsupervised fashion, with the goal of predicting a word given words which immediately preceded it. BERT is jointly trained on two tasks — predict a word given the words on its left and right (with some details on masking to avoid leakage) and predicting

the next sentence given a sentence. The latter is not standard, and language models typically do not directly capture relationships among sentences, but doing this tends to help applications like *Question Answering* systems for example.

4. **Model Size:** The authors of the BERT paper experimented with BERT of various sizes, most notable — Base (110M parameters) and Large (340M parameters). They found that bigger is better, even with the similar model architecture and small task-specific training data, this is largely due to the pre-training on a huge corpus.

Since the Original paper of BERT in Oct'2018, there have been many variants and modifications which have been proposed. Most notable among them are — RoBERTa and mBERT by Facebook AI, XLNet by researchers at CMU and GPT-2 by Open AI.

This concludes my roundup of some exciting things off late in NLP, clearly, there is so much happening in the field and also at such a rapid pace. There is a lot which people can do with very little effort these days thanks to the *giants* in the field which have done the heavy lifting and made it easier.

Additional Resources

1. Excellent visualization of *RNN*, *Attention* etc
2. Building a Transformer from scratch
3. Excellent library for using various Transformer architectures

)