Language Generation with Long Input Sequences

This review will be about deep learning language generation such as abstractive summarization using long documents as input sequences. We will focus on the progression of encoder-decoder neural network architectures in Natural Language Processing and the systems surrounding these architectures to handle particularly long sequences. Particularly, we will address the many problems that different neural networks have such as LSTMs with retaining contextual information across long sequences and Transformers having a quadratic memory overhead with respect to sequence length. We will also discuss the different strategies made to overcome the limitations of these networks. We will focus on popular two-step strategies where long documents are processed first in an extractive summarization step then fed as input into an abstractive summarization step. And we will also discuss the newer transformer architecture known as Longformer that tries to tackle the problem of long input sequences by introducing the idea of sliding-window attention to reduce memory consumption.

Long Short Term Memory, LSTM, is one of the earlier architectures that introduced the idea of applying deep learning to the field of Natural Language Processing. It is based on the Recurrent Neural Network architecture, RNN in which each word's neural network weights is shared with the next in the sequences otherwise known as parameter sharing. LSTM extends this idea by introducing notions of gates which control when weights are shared in the sequences which is helpful in retaining some notion of memory or context information across a sequence. LSTMs are good at relating information like this with short sequences but are actually fairly bad at doing this across longer sequences such as paragraphs. Because all the information is stored in its weights, you can imagine that these weights will constantly be altered over a very long sequence, so relating the beginning of one paragraph to the next is not really a feature of this architecture.Therefore, the newer architecture, the Transformer has been shown to be much better at handling longer input sequences.

Transformers are superb at relating contextual information across long sequences while also taking optimal advantage of computer parallelization. The architecture utilizes neural networks to generate items called query, key, and value for each word in a sequence. For example, a single word will be a query item, then its key item will be every other word in the sequence, then a similarity mechanism such as dot product will be used to give an attention score to every query-key pair, and a value item will subsequently give another weight to this score among all other generated attention scores. Essentially, this model is much better at relating words very far apart from one another because they are given a weight directly determining their correlation to each other in the context of the problem at hand. However, this model suffers from quadratic memory with respect to sequence length as each word must be compared to every other word in the sequence, which makes it infeasible for longer documents. Most transformers have maximum sequence lengths of 512 tokens while some have a maximum of 1024 but rarely do they go above that limit.

Therefore, in handling long documents for language generation, the standard approach has been to take two steps: first, use an extractive summarization step to condense different parts of the text, then use an abstractive summarizations step that takes the condense text input and generate an output with an encoder-decoder architecture such as a Transformer. In one paper from Google Brain, they were able to generate wikipedia articles by using the methodology above. For the extractive summarization step, they used methods such as tf-idf and search query evaluation to rank wikipedia document paragraphs with its title as a search query.

Recently, newer architectures based on the Transformer have been released specifically to handle long input sequences. For example, the Long-Document Transformer, Longformer, uses sliding-window attention to essentially reduce memory consumption of Transformers from quadratic to linear. Sliding window attention is when instead of comparing each word to every other word in a sequence, only words within a window are compared to each other to compute attention scores, and then this window slides over the entire sequence performing the same operation. Since Transformers stack encoder and attention layers on top of each other, eventually words will not only be compared to each other in its starting window but also to words very far apart as the encoding layers go deeper and more words share information across the sequence. With this sliding window operation, the Longformer has a linear memory consumption with respect to the input's sequence length, but in the paper, the architecture actually uses a window of size 512, pretty much as long as normal Transformers. So it does not necessarily make Transformers more accessible to smaller computing power, but rather is used solely for handling long input sequences such as those that go up to 4096 tokens. Additionally, Longformer encoder-decoder architectures have been shown to work well for summarization tasks; however, memory consumption is still a large issue as this architecture for 4096 max sequence length still only work on 48 GB GPU memory sizes, which is still not a very popular type of GPU make at the present.

The problem of handling long documents as input sequences is still one that needs to continue to be tackled and solved. Though with the many different techniques arising such as sliding window attention, long input sequences will likely be much better handled in the near future.

References

LSTM: https://www.bioinf.jku.at/publications/older/2604.pdf

Transformer: https://arxiv.org/pdf/1706.03762.pdf

Wikipedia Generation: https://arxiv.org/pdf/1801.10198.pdf

Longformer: https://arxiv.org/abs/2004.05150