QuipPy : Text Summarization Model

CS 652 Project Presentation

Project Repo | Colab Document

Anisha Katiyar (202051027)
Janki Kotadiya (202051104)
Madhur Gupta (202051112)

About

- In Natural Language Processing, the summarization task can be broadly classified into two categories:
  a. Extractive Summarization
  b. Abstractive Summarization

- We have used Abstractive Summarization in our project.
  a. Heuristic approach to train the system in making an attempt to understand the whole context and generate a summary based on that understanding.
  b. Basically, it's more Human like.

- We have used Transformer model
  a. Developed by Google.
  b. Has been a pioneer to many approaches in sequence transduction tasks (converting one sequence to another)

Abstractive Text Summarization

- Abstractive Text Summarization is the task of generating a short and concise summary that captures the salient ideas of the source text.
- Unlike extractive summarization, which simply selects and combines sentences from the original text, abstractive summarization involves creating a new summary that may not contain the exact same wording as the original text.
- Abstractive summarization relies on natural language processing and artificial intelligence algorithms to analyze the original text and identify the most important information.
- We are motivated to work on this project is due to the fact that abstractive summarization closely resembles the process that humans undergo when summarizing.

Why Transformer Model?

The main issue with RNNs lies in their inability of providing parallelization while processing.
The processing of RNN is sequential, i.e. we cannot compute the value of the next time step unless we have the output of the current. This makes RNN-based approaches slow.

This issue, however, was addressed by Facebook Research wherein they suggested using a convolution-based approach that allows incorporating parallelization with GPU.

These models establish hierarchical representation between words, where the words that occur closer in sequences interact at lower levels while the ones appearing far from each other operate at higher levels in the hierarchy.

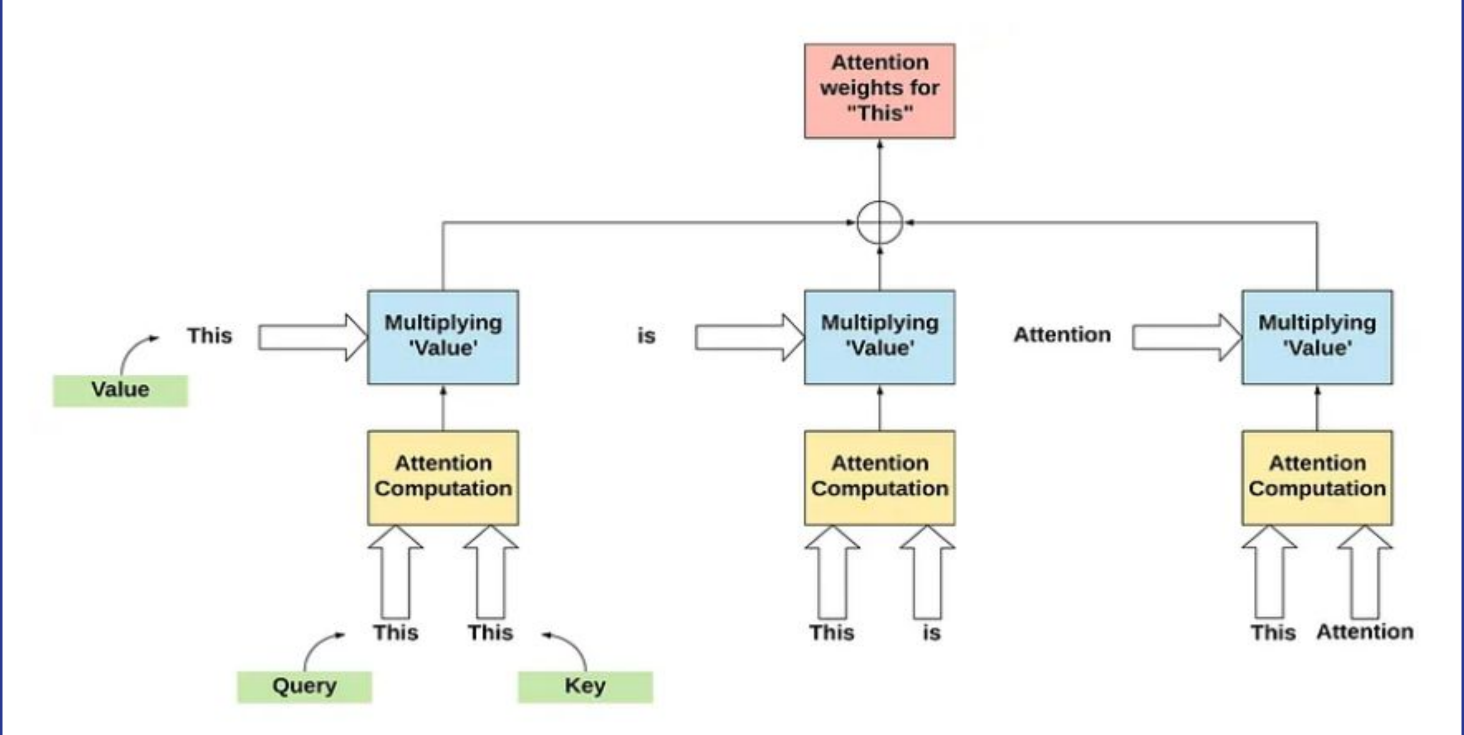
So yes, why the Transformer Model now?

The Transformer uses the self-attention mechanism where attention weights are calculated using all the words in the input sequence at once, hence it facilitates parallelization.

In addition to that, since the per-layer operations in the Transformer are among words of the same sequence, the complexity does not exceed cubic time polynomial.

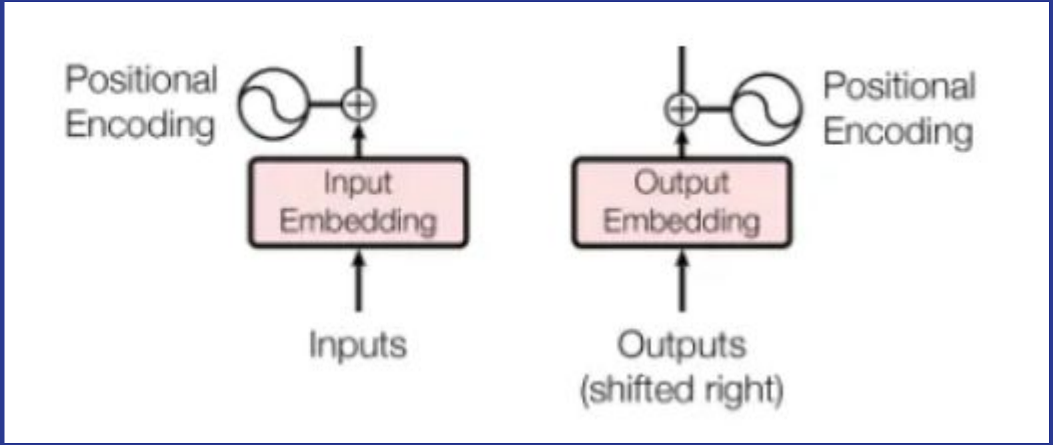
Hence, the transformer proves to be effective (since it uses attention) and at the same time, a computationally efficient model.

1. The Self Attention

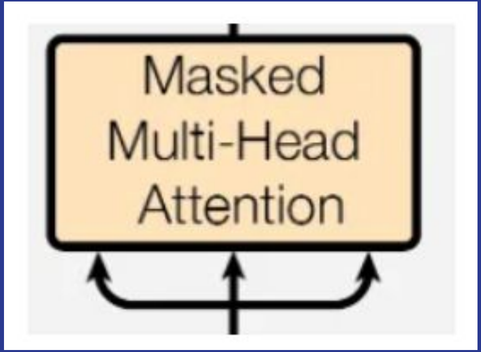


Self-Attention

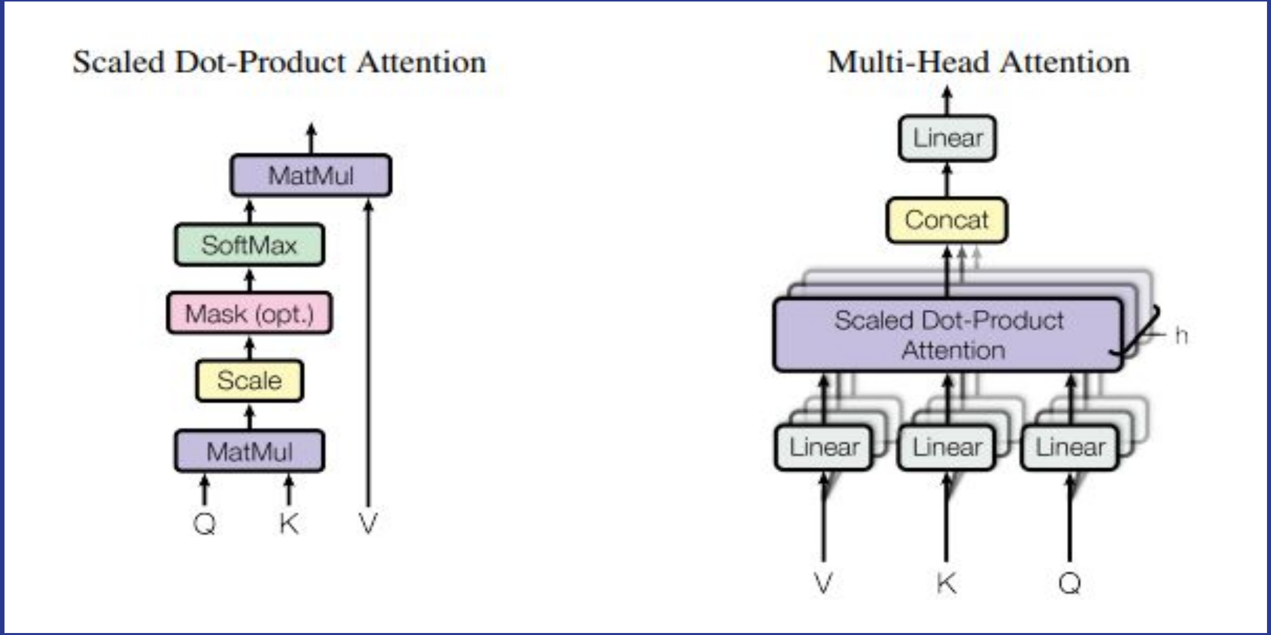
2. Positional Encoding



3. Masking



4. Scaled Dot-Product and Multi-Head Attention



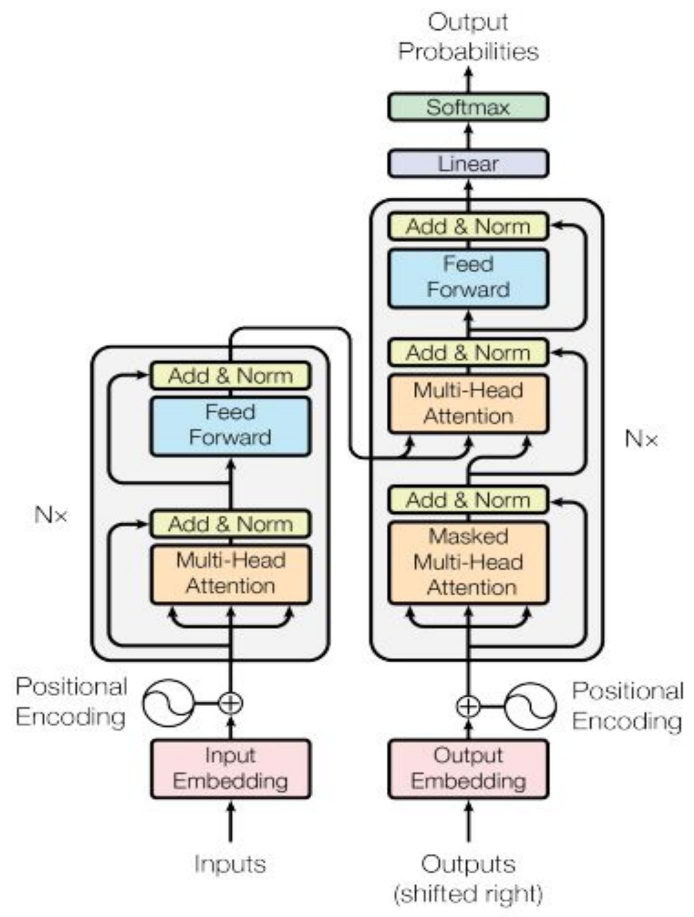
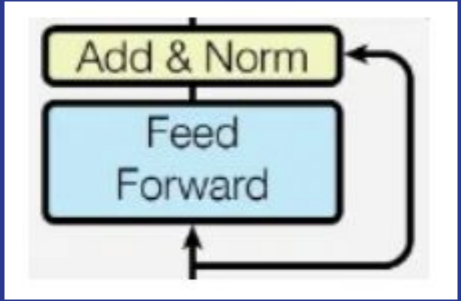


Figure 1: The Transformer - model architecture.

5. Point-Wise Feed Forward Network and Residual Dropout



---> Final Architecture Model of Transformer

THE WORKING AND THE CODE

--THE END --