Individual Project Report
Andy Christian
DATS 6312
May 7, 2023

Text Summarization of News Articles for Daily/Topic News App

I. Introduction

In many industries, things can change fast. Staying on-top of recent news, trends, and even rumors can be critical for industry decision makers to make well informed choices. News articles and technical reports are critical sources of this information. While the details in these articles and reports are important for implementation, the higher-level key points are more relevant for decision makers in understanding the larger picture. An NLP model capable of summarizing the main points of news articles and technical reports, as well as creating a meta-summary of all the relevant information from the article or reports examined from a given day/topic would be a useful tool in helping decision makers stay aware of their industry's situation.

The goal of the project was to provide a proof of concept of using a transformer to do text summarization as a first step for creating a Trending News/Academic Report Summary App. Further steps would be to refine the results of the model and develop strategies for how to use the model effectively at this task.

II. Description of Individual Work

NOTE: a python file showing examples of the code I contributed to the project can be found inside the Code folder of my individual folder in the project repo. I commented out the code written by others and put blocks of ##### symbols around the sections of code which I added.

Daqian chose the Transformer model to work with and wrote the initial preprocessing and training. Rajkumar put the code into an object-oriented format, refined the training and investigated optimizations. I reviewed the initial code base, identifying some errors and areas for improvement, as numerous sections of the code were having errors or not running properly. Specifically, I fixed errors with the evaluation function, and re-structured some variables as class attributes to ensure every function/step had access to the information it needed to run correctly. I also wrote a test loop using the model's generate() function and stored the results to compare with the target summaries. Additionally, I worked with Rajkumar to troubleshoot some

further problems that came up with the training loop as he was doing optimizations. However, he pushed the code as it was the section he was working on. Lastly, in order to help with quick testing/iteration as we were testing the model to make sure it worked, I wrote code to take a small sample of the data so that we could check for errors and make changes quickly.

III. Results

Two final versions were tested. These versions were the same except that one took in a max of 256 tokens, and the other took up to 1024 tokens from the original text to create the summaries with. While the cross-entropy loss was much lower for the 1024-input-tokens model, the rouge score for both models was about ~0.24. Additionally, when I read a few sample summaries of the 1024-input-tokens model, I thought that they were worse summaries. That is, even if they had more of the same words/sequences as the target summary compared to the other model, the information the larger model was choosing was not good "top-level" summary information, in my view. In fact, when I was fixing errors, I dropped the batch-size and training epochs to 1 and 2 respectively. While the loss and rouge scores this produced as I ran the model for troubleshooting purposes were lower than in the final models, the summaries read better, in my opinion.

IV. Summary and Conclusion

As a proof of concept, the project was successful in achieving the goal. However, as stated in the results sections, I felt that the summaries produced by the model that had better loss and rouge scores, were not very good *summaries*, even if they shared more tokens and sequences with the target. This has me thinking a lot about whether additional strategies could be brought to bear when working with this kind of NLP problem. For example, perhaps I could try summarizing each paragraph of an article and then combining those as the complete article summary. The point here is that while the model might produce words that share numerical similarity to the target, it did not seem to capture what was important from the article overall. Thus, if each paragraph was summarized, this would be "guiding" the model a bit to pick out the important point from each paragraph. This could even be summarized further of it still had too much text. I am also curious to do a zero-shot classification model of the original text, the human-written summary, and the model-produced summary and see how consistent those three classifications are for each input.

V. Code Breakdown

Total lines contributed: 30

Number of copied Lines: 7
Number of lines modified: 3
Number of unique Lines: 23
Score = (7-3)/(7+23)*100 = 13.3

VI. References

Raffel C., et al. (2019). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Cornell University arxiv. https://doi.org/10.48550/arXiv.1910.10683

Shankar, G.P. (2021). CNN-Daily Mail News Text Summarization [Data Set]. Kaggle. https://www.kaggle.com/datasets/gowrishankarp/newspaper-text-summarization-cnn-dailymail