Abstractive Summarization for structured conversational text

Ayush Chordia (ayushc@stanford.edu) (Video Link: https://youtu.be/iD87gfueBCw)

CS230 FINAL PROJECT PRESENTATION, STANFORD UNIVERSITY

Motivation & Objectives

- Abstractive summarization is the task of generating a summary comprising of a few sentences that meaningfully captures the important context from given text input.
- Known challenging problem in NLP since summarization doesn't involve selecting existing sentences from the input, instead paraphrasing the main contents of the document using vocabulary previously unseen

Datasets

- Baseline metrics, the current model was trained on CNN/Daily Mail dataset as mentioned in Nallapati et al. [1], the dataset itself contains news article (781 tokens on average) paired with multi-sentence summaries (3.75 sentences or 56 tokens on average).
- Transcripts of earning call for public companies along with annotated summaries were also used during training.
- The annotated meeting conversation from AMI corpus along with their abstractive summaries were also added to training and test set
- The dataset was prepared by first splitting the sentences with Stanford CoreNLP toolkit (pre-processed using the techniques mentioned in See et al. [2].

Methods

- Neural approaches to abstractive summarization have been previously implemented by using sequence-to-sequence models where an encoder maps sequence of tokens from the source document $x = [x_1, \ldots, x_n]$ to sequence of continuous representations $z = [z_1, \ldots, z_n]$ and a decoder generates target summary $y = [y_1, \ldots, y_m]$ token-by-token

Pointer Generator Network (Model 1)

Baseline Model: Sequence-to-sequence attention model

- The tokens of article are fed into an encoder (single layer bidirectional LSTM), producing sequence of encoder hidden states. On each step t, the decoder (single layer unidirectional LSTM) receives the word embedding of the previous word.

Pointer-Generator Network

- It allows both copying words via pointing and generating words from a fixed vocabulary. For each decoder timestep a generation probability $p(gen) \in [0,1]$ is calculated, that weighs the probability of generating words from vocabulary versus copying words from source text

Coverage

- The coverage is used to solve this problem by maintaining a coverage vector which is the sum of attention distributions over all previous decoder timesteps. This coverage vector is an extra input to the attention mechanism's current decision

Pretrained Encoders using BERT (Model 2)

- Pertained language models have been used as encoders for sentence and paragraph level natural level understanding problems. In this architecture, the impact of language model pertaining was measured on summarization.
- Architecture is based on Liu et al. [3] where the document level encoder based on BERT (Devlin et al. 2019 [4] is used to encode a document and obtain representation for its sentences. The potential of BERT was explored in the second model by leveraging the encoder-decoder architecture, combining the pertained BERT encoder as described above with randomly initialized 6-layer Transformer decoder

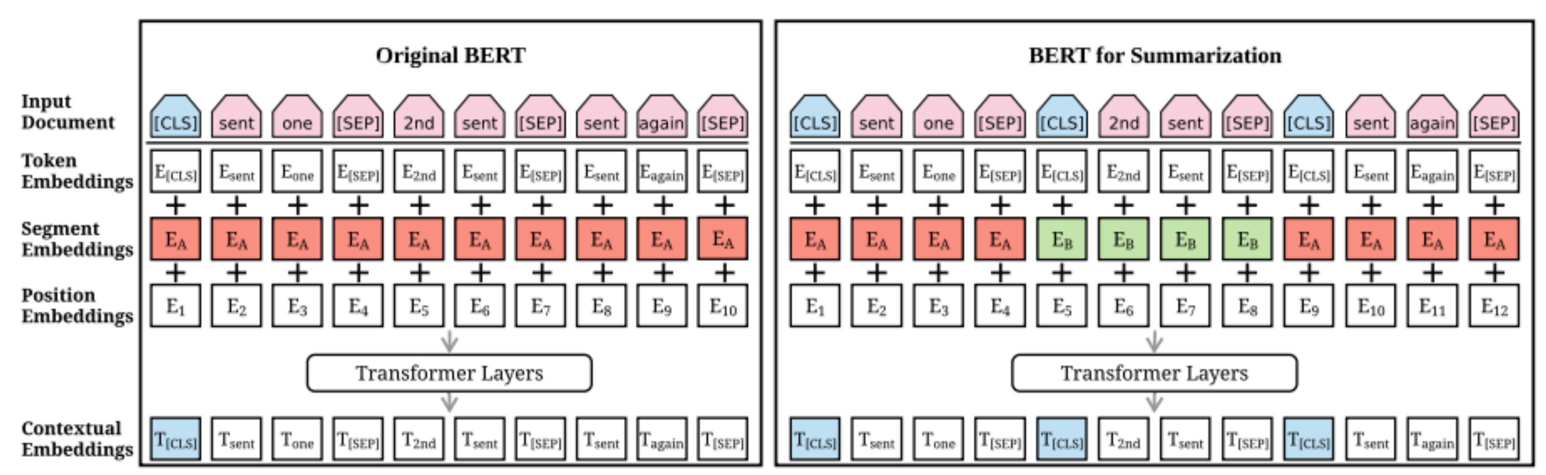

Figure 1: Architecture of the original BERT model (left) and Pretrained Encoders using BERT for summarization (right). The sequence on top is the input document, followed by the summation of three kinds of embeddings for each token.

Experiments

Model 1

- One of the limitations of the architecture proposed in See et al. [4] was that the article was truncated to 400 tokens during training and test time and limits the length of summary to 100 tokens for training and 120 tokens for testing
- The current model was trained on Quadro P400 GPU with batch size of 16 and trained for 75,000 iterations and it took 3 day 16 hours for the current checkpoint with the 50k vocabulary

Model 2

- I used the Pytorch, OpenNMT and the bert-base-uncased version of BERT, both source and target texts were tokenized with BERT's subwords tokenizer
- In the abstractive model, dropout (with probability 0.1) was applied before all linear layers, label smoothing with smoothing factor 0.1 was also used.
- The model was trained on 2 Tesla P100 GPU and it took 4 days to train the model to 156,000 iterations.

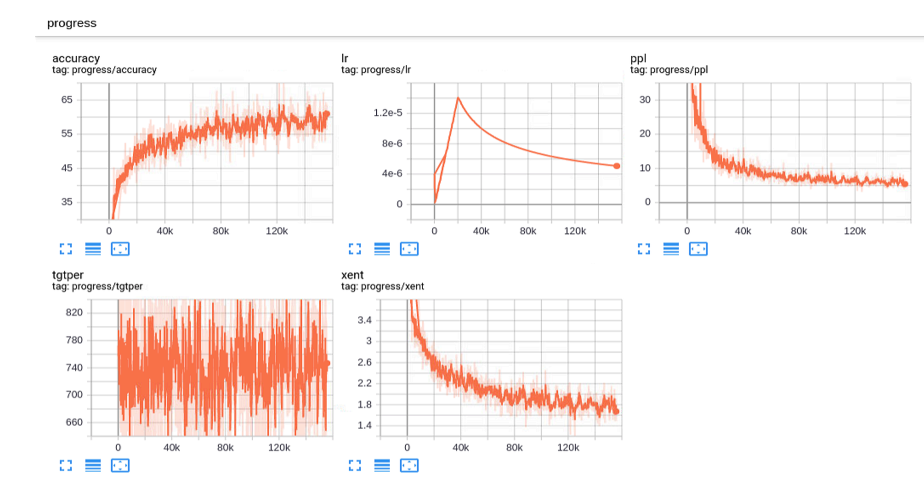

Figure 2: Current Metrics on train and validation set

Results

- ROUGE score with standard options was used the metric for evaluation. The idea behind Rouge score is to count the number of overlapping unites between generated and referenced summaries.
- We plan to report the F-measures of ROUGE-1 (R-1), ROUGE-2 (R-2), ROUGE-L(R-L). The current test set compromised of 12,000 input text and corresponding summaries

Model	ROUGE-1	ROUGE-2	ROUGE-L
Model 1	0.3642	0.1552	0.3322
Model 2	**0.4095**	**0.1864**	**0.3793**

References

[1] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre and Bing Xiang "Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond". In: arXiv preprint arXiv:1602.06023 (2016).
[2] Abigail See, Peter J Liu, Christopher D Manning "Get to the Point: Summarization with Pointer Generator Network". In: arXiv preprint arXiv:1704.04368 (2017)
[3] Yang Liu, Mirella Lapata "Text Summarization with Pretrained Encoders". In: arXiv preprint arXiv:1908.08345 (2019)
[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies