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 sets.
- 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 $x = \{x_1, ..., x_n\}$ to sequence of continuous representations $z = \{z_1, ..., z_m\}$ and a decoder generates target summary $y = \{y_1, ..., y_m\}$ token-by-token.
- 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.

Results

- ROUGE score with standard options was used the metric for evaluation. The idea behind ROUGE score is to count the number of overlapping units between generated and referenced summaries.
- We plan to report the F-measures ROUGE-1 (R1), ROUGE-2 (R2), ROUGE-L(R-L). The current test set comprised of 12,000 input text and corresponding summaries.

<table>
<thead>
<tr>
<th>Model</th>
<th>ROUGE-1</th>
<th>ROUGE-2</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>0.3642</td>
<td>0.1552</td>
<td>0.3322</td>
</tr>
<tr>
<td>Model 2</td>
<td>0.4095</td>
<td>0.1864</td>
<td>0.3793</td>
</tr>
</tbody>
</table>

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.

References