Evaluating Model Architectures for Extractive Summarization using BERT Embeddings

Constance J. Lam

Department of Computer Science, Yale University, New Haven, CT

LILY Lab

Introduction

Automatic text summarization has been an emerging field as the research and industry demand is growing for succinct, comprehensible, and accurate summaries from longer texts. Currently, there are two main approaches to automatic text summarization: extractive and abstractive. Extractive summarization involves content selection via extracting phrases or sentences from the text to generate a summary, while abstractive summarization involves generating original summaries by paraphrasing the intent of the original text [1]. Recent literature on various approaches to extractive summarization has suggested that there is much room for improvement in sentence representations for summarization. The Kedzie and McKeown paper focuses on sentence extractive summarization, where the basic unit of extraction is a sentence with a word limit (budget), and looks at averaging encoders, RNN encoders, and CNN encoders, as well as various approaches to sentence extractors. The paper found that deep learning models for summarization are not necessarily higher performing than simpler models, such as word embedding averaging, and that pre-trained word embeddings are as good, or better than, learned embeddings [2]. As a result of these findings, my project attempted to test the efficacy of BERT (Bidirectional Encoder Representations from Transformers), which are pre-trained but contextual embeddings based off of a language representation model developed by the Google AI Language group. My project aimed to see whether BERT embeddings, jointly conditioned on both left and right context in all layers, could be applicable to extractive summarization contexts.

Materials and Methods

The method to test BERT embeddings with extractive summarization involved running the BertModel (a pre-trained language model that produces embeddings upon receiving the data) and BertTokenizer with pyTorch in the Kedzie models. The Kedzie models were also tested as a comparison point with the Bert+Kedzie models. Glove embeddings and BERT embeddings that update during training were used. The dataset utilized in my project was the New York Times (NYT) corpus, which contains two types of abstracts (archival and online teaser) for its articles. Following the steps of the Kedzie paper, this project used all the articles that have a concatenated summary length of at least 100 words; this served as the gold standard summary. The articles were then divided into training, validation, and test splits based on date. Due to the large size of BERT embeddings, memory issues affected how large the training epochs for the models could be; thus, instead of maximum epoch sizes of 50, as used in Kedzie, this experiment ran all models with a maximum epoch size of 5.

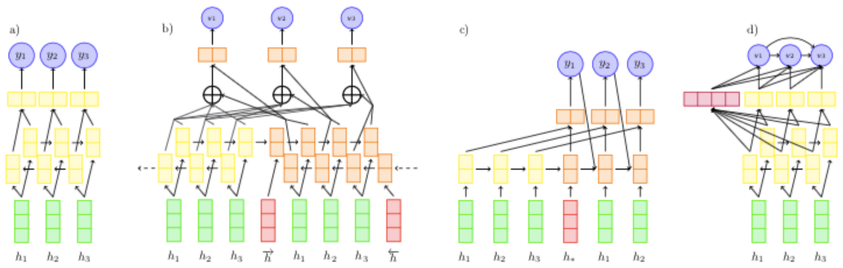

Figure 1: Sentence extractor architectures from Kedzie [2]: a) RNN, b) Seq2Seq, c) Cheng & Lapata, and d) SummaRunner.

Extractor	Enc.	CNN/DM		NYT		DUC 2002		Reddit		AMI		PubMed	
		M	R-2	M	R-2	M	R-2	M	R-2	M	R-2	M	R-2
Lead	–	24.1	24.4	30.0	32.3	25.1	21.5	**20.1**	**10.9**	12.3	2.0	15.9	9.3
Seq2Seq	Avg.	**25.2**	25.4	29.8	34.7	**26.8**	22.7	**20.4**	**11.4**	**17.0**	**5.5**	19.8	17.0
	RNN	**25.1**	25.3	30.2	**35.9**	**26.7**	22.5	**20.5**	**12.0**	16.1	**5.3**	19.7	16.7
	CNN	25.0	25.1	29.9	35.1	**26.7**	**22.7**	**20.7**	**13.2**	14.2	2.9	19.8	16.9
Cheng & Lapata	Avg.	25.0	25.3	30.4	**35.6**	**27.1**	**23.1**	**20.9**	**13.6**	**16.7**	**6.1**	**20.1**	**17.7**
	RNN	25.0	25.0	**30.3**	**35.8**	**27.0**	**23.0**	**20.3**	**12.6**	**16.3**	**5.0**	19.7	16.7
	CNN	**25.2**	25.1	29.9	35.0	**26.9**	**23.0**	**20.5**	**13.4**	14.3	2.8	19.9	16.9
Summa Runner	Avg.	25.1	25.4	30.2	35.4	26.7	22.3	**21.0**	**13.4**	**17.0**	**5.6**	19.9	17.2
	RNN	25.1	25.2	30.0	35.5	26.5	22.1	**20.9**	**12.5**	**16.5**	**5.4**	19.7	16.5
	CNN	24.9	25.0	29.3	34.4	26.4	22.2	**20.4**	**12.3**	14.5	3.2	19.8	16.8
Oracle	–	31.1	36.2	35.3	48.9	31.3	31.8	24.3	16.2	8.1	3.9	24.1	25.0

Figure 2: METEOR (M) and ROUGE-2 recall (R-2) results across all extractor/encoder pairs from Kedzie [2].

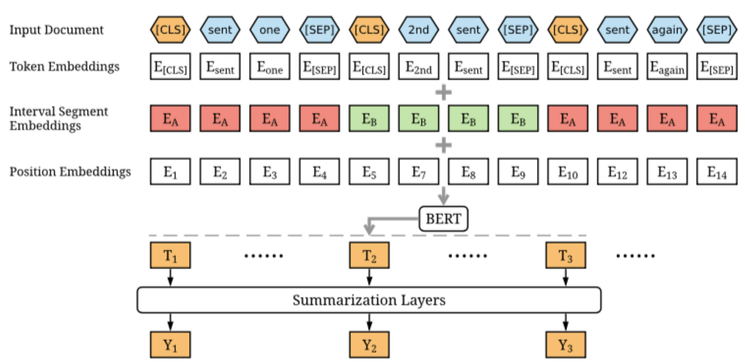

Figure 3: The overview architecture of the BERTSUM model. [3]

```
# Tokenized input
text = "[CLS] Who was Jim Henson ? [SEP] Jim Henson was a puppeteer [SEP]"
tokenized_text = tokenizer.tokenize(text)
print(tokenized_text)
```

```
['[CLS]', 'who', 'was', 'ji', '##m', 'he', '##nson', '?', '[SEP]', 'ji',
'##m', 'he', '##nson', 'was', 'a', 'puppet', '##eer', '[SEP]']
```

Figure 4: An example of how BertTokenizer would tokenize a sentence.

Current Model and Results

Currently, this project will continue with training models with all extractor/encoder pairs from Kedzie using Glove embeddings and BERT embeddings on 5 epochs. Furthermore, one BERT+Kedzie model example with 50 epochs will be run as a reference point to see how much higher performing the summaries are when trained with more epochs. However, we can only do this with one model, since the process is extremely time and memory intensive. The BERT+Kedzie models are expected to out perform Kedzie models with Glove embedddings

Conclusion and Planned Experiments

Extractive summarization with BERT is a unique problem that provides a lot of opportunities for future research. Furthermore, the maximum number of 5 epochs serves as a current limitation on the efficacy of the model. A projected solution would be to add multiple GPU processing so that the time and memory constrictions can be addressed in cases of maximum epoch sizes of 50. Also, it would be interesting to test BERT embeddings with reinforcement learning models with extractive summarization, since reinforcement learning is another recent novel approach to summarization.

Sources Cited

1. Chris Kedzie, Kathleen McKeown, Hal Daume III. 29 Oct 2018. Content Selection in Deep Learning Models of Summarization.
2. Pranay. 2017. Text Summarization in Python: Extractive vs. Abstractive techniques revisited
3. Yang Liu. 25 Mar 2019. Fine-tune BERT for Extractive Summarization

Acknowledgement

I would like to thank Professor Dragomir Radev and Alexander Fabbri for their guidance on this project.