NLP Applications

Project Report

EXTRACTIVE TEXT SUMMARIZATION

Submitted To:

Dr. Manish Shrivastava
LTRC ,IIIT Hyderabad

Submitted By:

Ishan Tyagi (2018201017)
Aman Krishna (2018201070)
Rajat Yadav (2018201077)

Contents

0.1 Important Links

- **Paper1** : Fine-tune BERT for Extractive Summarization.
  https://arxiv.org/pdf/1903.10318v2.pdf
- **Paper2** : Text Summarization with Pretrained Encoders.
  https://arxiv.org/pdf/1908.08345v2.pdf

Our Implementation
https://github.com/rajatyadav1994/Text_Summarization

Dataset: CNN/DailyMail
https://drive.google.com/open?id=1x0d61LP9UAN389YN00z0Pv-7jQgirVg6

Presentation
https://docs.google.com/presentation/d/1KR8ijv8c99Ez5xrjbOHprZ_ewpGyPOYW96UGXOLBQ4M/edit

Training video link
https://www.youtube.com/watch?v=AHpJm3Ndpco&feature=youtu.be

Testing video link
https://www.youtube.com/watch?v=xcYWCaNiE8Q

0.2 Abstract

Single-document summarization is the task of automatically generating a shorter version of a document while retaining its most important information. The task has received much attention in the natural language processing community due to its potential for various information access applications. Examples include tools which digest textual content (e.g., news, social media, reviews), answer questions, or provide recommendations.

The task is often divided into two paradigms, abstractive summarization and extractive summarization. In abstractive summarization, target summaries contains words or phrases that were not in the original text and usually require various text rewriting operations to generate, while extractive approaches form summaries by copying and concatenating the most important spans (usually sentences) in a document. In this project, we focus on extractive summarization using state-of-the-art model BERT.

Bidirectional Encoder Representations from Transformers(BERT) a pre-trained Transformer model, has achieved ground-breaking performance on multiple NLP tasks. Here we showcase how BERT can be usefully applied in extractive text summarization. Our extractive model uses BERT to generate sentence level embeddings which serve as input into other model. We are fine-tuning (training BERT weights during training) with a model on top to achieve state-of-the art result for extractive summarization. We have use 5 different models to showcase the performance:

- Baseline Model: Transformer
- BERT + Classifier (FFNN)
- BERT + Transformer
- BERT + RNN
- BERT Large

0.3 Introduction

Text Summarization

- Text summarization refers to the technique of shortening long pieces of text while focusing on the sections that convey useful information, and without losing the overall meaning
- The intention is to create a coherent and fluent summary having only the main points outlined in the document
- Text summarization in NLP is treated as a supervised machine learning problem.
- There are basically 2 types of approaches to summarize the text in NLP:
  1.Extraction-based summarization
  2.Abstraction-based summarization

  The **extractive text summarization** technique involves pulling keyphrases from the source document and combining them to make a summary. The extraction is made according to the defined metric without making any changes to the texts

  The **abstractive text summarization** algorithms create new phrases and sentences that relay the most useful information from the original text- just like humans do. When abstraction is applied for text summarization in deep learning problems, it can overcome the grammar inconsistencies of the extractive method
- Although many neural models have been proposed for extractive summarization recently the improvement on automatic metrics like ROUGE has reached a bottleneck due to the complexity of the task
- In our approach we use BERT. With its pre-training on a huge dataset and the powerful architecture for learning complex features, we can further boost the performance of extractive summarization

BERT

- **BERT**(Bidirectional Encoder Representations from Transformers) is a language model and consists of a pre-trained Transformer Encoder stack.
- It takes in word tokens as input and produces word embedding as well as sentence embedding as output

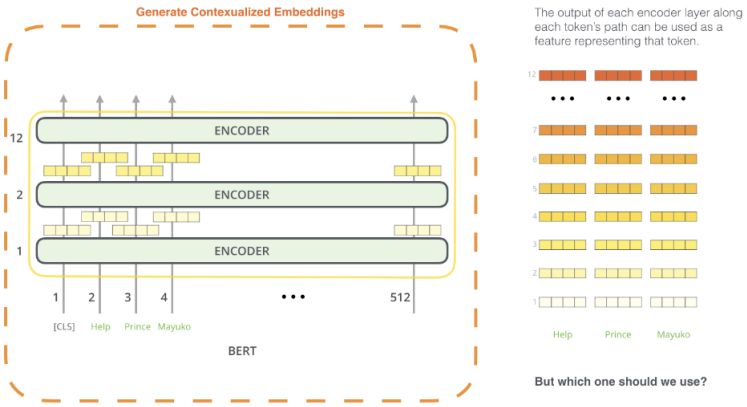
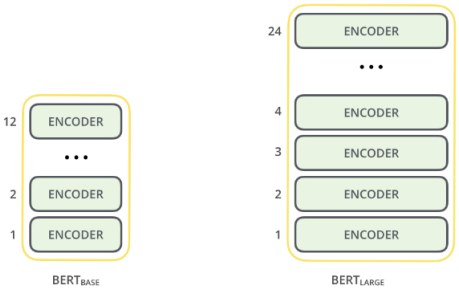


Figure 1: BERT Input Output

There are two implmentations of BERT model:
1.**BERT BASE** - Comparable in size to the OpenAI Transformer in order to compare performance.
2.**BERT LARGE** - A ridiculously huge model which achieved the state of the art results reported in the BERT paper.



[b]0.9

Figure 2: BERT Base vs BERT Large

0.4 Literature Survey

In the present scenario of fast growing usage of the Internet, the task of searching for the user has become very tedious . Among these results sorting out the most relevant is another difficult task.
Text Summarization came into being in order to reduce these difficulties for the user. It is the process of reducing the huge amount of data which is available on Internet or on any other source of information like newspapers, books etc., into summarized documents, in order to create the summary of the document that retains the most important points and the overall meaning. The process of text summarization is further divided into extractive text summarization and abstractive text summarization. In Extractive summarization, the summary is generated by using most important sentences from the document exactly as they appear in it and then concatenating them in shorter form.

Different methods have been used over the years to solve the problem of Extractive summarization:

- **Neural Networks**: Before BERT or Transformer RNN were one of the best way to capture context among the words and with it's extension the sentence. RNN's have the capability to retain information across sequence of words which allow it to capture good sentence level embedding like the ones mentioned here:
  `https://arxiv.org/pdf/1611.04230.pdf`

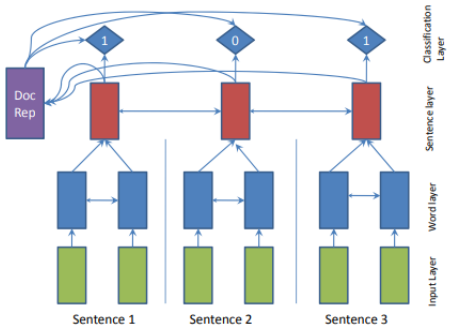


Figure 3: Extractive Summarization using RNNs

- **CNNs**: With the advent of CNNs in the field of Computer Vision and their ground breaking performance over some specific tasks made some NLP researcher to look into CNN for solving the problem of extractive summarization. The words of the sentence can be viewed as vectors and sentence being a combination of vectors can be seen as an image with concatenation of it's constituent words and can be use with a CNN as done here:
  `https://www.aclweb.org/anthology/N18-1158.pdf`

Figure 4: Extractive Summarization using CNNs

- **Transformers**: With the advent of transformers came the rise of attention based context embedding. The transformer were fed sentence embedding (using an RNN etc.) or were used to form sentence embeddings of their own to perform the task of extractive summarization. The advantage of transformers was their ability to train in parallel and more importantly capturing long term dependencies using the attention mechanisms which were missed by LSTMs or other networks

- **BERT**: BERT gave researchers tremendous power. It was the true case of learning transfer. They can fine-tune or use the complete BERT architecture as it is for a variety of task thanks the way BERT was trained. BERT was trained as a bi-directional language model implemented using the Transformer architecture. It allowed it to capture rich contextual information. Moreover, the researcher of BERT built into it the ability to capture information of a segment (sentence, paragraph etc) using the special CLS symbol which makes transfer learning all the more easier. Here is a paper utilizing BERT which achieved state-of-the-art result in extractive summarization task:
  `https://arxiv.org/pdf/1903.10318.pdf`

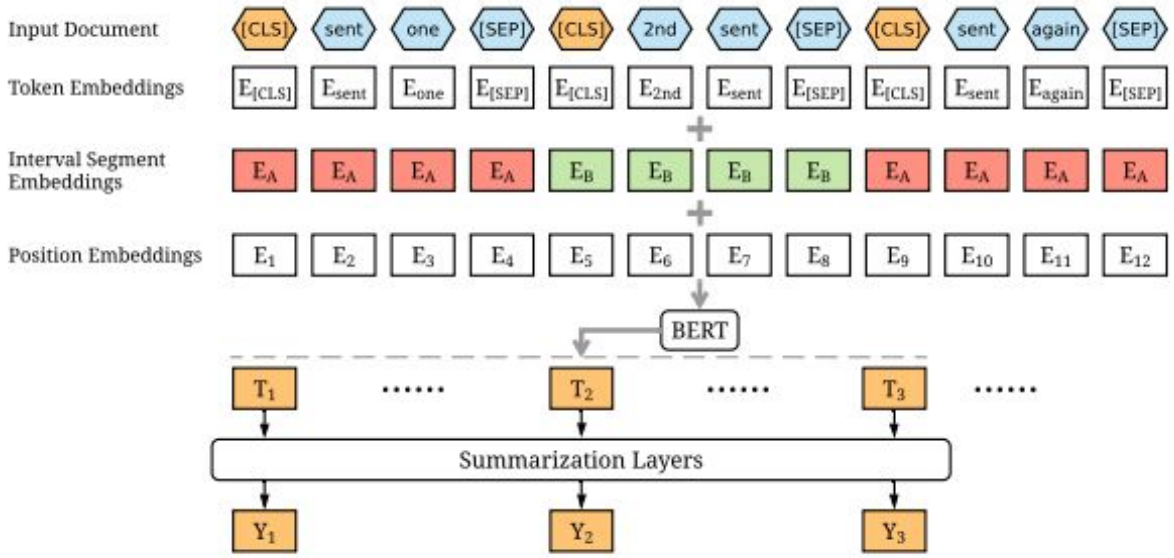
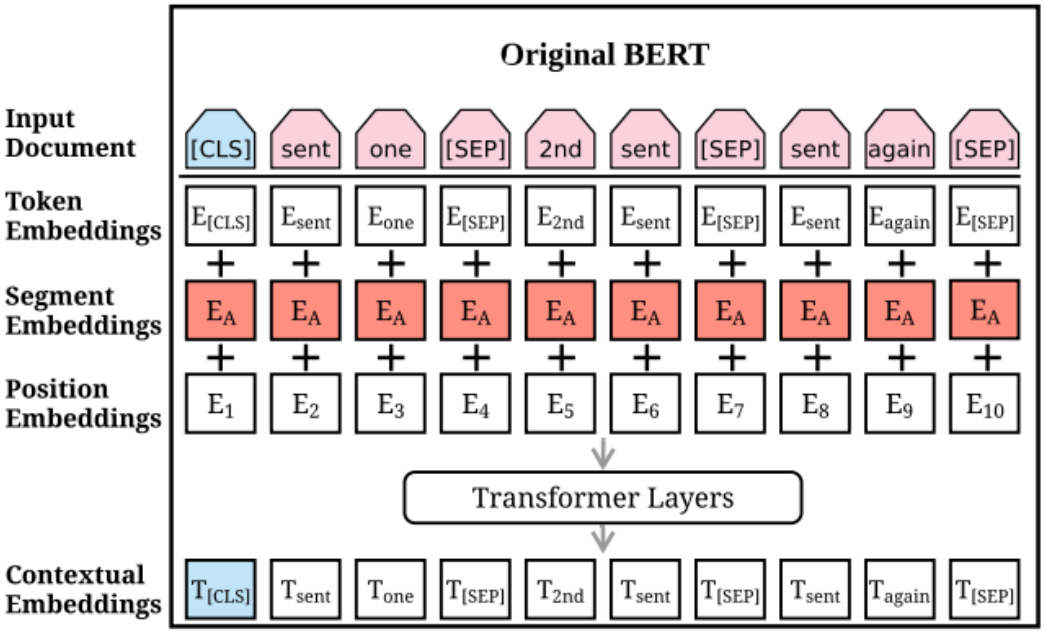


Figure 5: Extractive Summarization using BERT

0.5 Research Methods

Basic Approach

Let d denote a document containing several sentences $[sent_1, sent_2, ..., sent_m]$, where $sent_i$ is the i^{th} sentence in the document. Extractive summarization can be defined as the task of assigning a label $y_i \in 0,1$ to each $sent_i$, indicating whether the sentence should be included in the summary. It is assumed that summary sentences represent the most important content of the document.

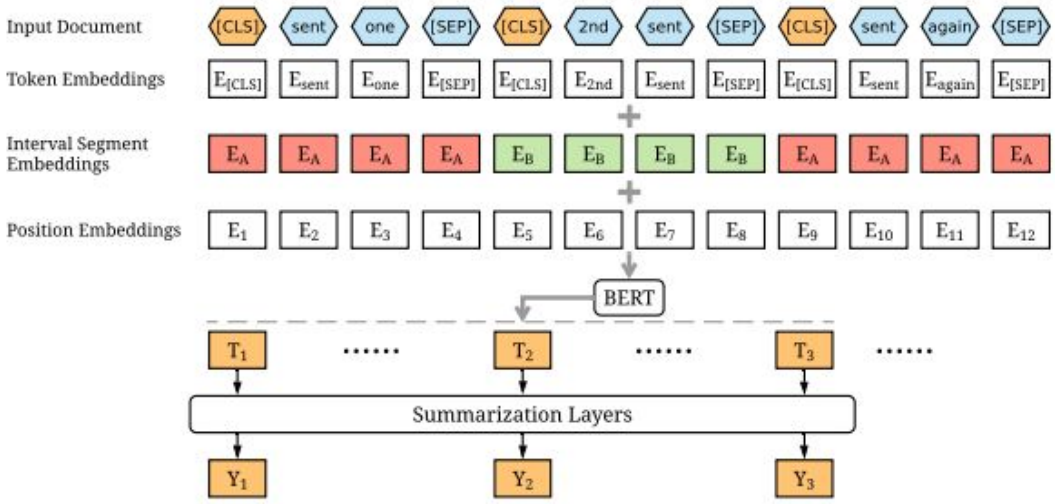


(a) BERT Model

- [CLS]: Special symbol inserted at the start of segment
- [SEP]: Special symbol inserted at the end of segment
- E_x: Token embedding for word X
- E_A & E_B: Embedding for even and odd sentences
- E_i: Positional Embeddings
- T_i: Output Embeddings of BERT for input token i
- T_{CLS}: Output Embeddings of BERT for the segment

Extractive Summarization with BERT

To use BERT for extractive summarization, we require it to output the representation for each sentence. However, since BERT is trained as a masked-language model, the output vectors are grounded to tokens instead of sentences. Meanwhile, although BERT has segmentation embeddings for indicating different sentences, it only has two labels (sentence A or sentence B), instead of multiple sentences as in extractive summarization. Therefore, we modify the input sequence and embeddings of BERT to make it possible for extracting summaries. We insert CLS symbol at the start of each sentence to capture the contextualized sentence embedding of the document.



(a) BERT Model for Summarization

- [CLS] **& [SEP]**: Special symbol inserted at the start and end of the each sentence
- E_x: Token embedding for word X
- E_A & E_B: Embedding for even and odd sentences
- E_i: Positional Embeddings
- T_i: Output Embeddings of BERT
- T_{CLS}: Output Embeddings of BERT for the segment
- Y_i: Output for each sentence i.e. whether to keep or reject(like binary classification)

Fine Tuning BERT

Our **extractive model** combines sentence embedding output by BERT with a different model. Both BERT and the additional model are trained together.We have used three different model in combination with BERT

Case I: Simple Classifier
Simple Classifier adds a FFNN on the BERT sentence embedding and uses a sigmoid function for prediction. Here sigmoid function: σ

$$\hat{Y}_i = \sigma(W_o T_i + b_o)$$

Case II: Inter-sentence Transformer
Instead of a simple sigmoid classifier, Inter-sentence Transformer applies more Transformer layers on the sentence representations extracting document-level features focusing on summarization tasks from the BERT outputs:

$$\hat{h}^l = LN(h^{l-1} + MHAtt(h^{l-1})$$
$$h^l = LN(\hat{h}^l + FFN(\hat{h}^l))$$

- **LN:** Layer normalization operation
- **MHAtt:** Multi-head attention operation
- **Superscript l:** The depth of the stacked layer
- Final output layer is still the Sigmoid layer
- h_0 = PosEmb(T), where PosEmb is the function of adding positional embeddings (indicating the position of each sentence) to T
- **T** : Sentence vectors output by BERT

Case III: BERT + LSTM

We apply an LSTM layer over the BERT outputs to learn summarization-specific features. At time step i, the input to the LSTM layer is the BERT output T_i , and the output is calculated as:

$$\mathrm{F}_i, I_i, O_i, G_i = LN_h(W_h h_{i-1} + LN_x(W_x T_i))$$
$$C_i = \sigma(F_i) \odot C_{i-1} + \sigma(I_i) \odot \tanh(G_{i-1})$$
$$h_i = \sigma(O_t) \odot \tanh(LN_c(C_t))$$

- F_i, I_i, O_i: Forget gates,Input gates,Output gates.
- G_i:Hidden vector.
- C_i:Memory vector.
- h_i:Output vector.
- LN_h, LN_x, LN_c:Different Layer Normalization operations.

0.6 Findings and Analysis

Metric for Evaluation:
We used ROUGE-F1 score as our metrix evaluation. ROUGE-F1 score can be calculated as follows

$$\text{ROUGE-F1 Scores} = \frac{2PR}{(P+R)}$$

P= ROUGE-PRECISION
R= ROUGHE-RECALL

ROUGE-N: measures unigram, bigram, trigram and n -gram.
ROUGE-L: measures the longest matching sequence of words using LCS.

Comparison after 1000 training steps
Due to lack of proper hardware we manage to train our Bert-LARGE + Transformer model for only 1000 steps. To highlight the model performance we trained every model i.e. Baseline, Bert-BASE + Classifier, Bert-BASE + RNN, Bert-BASE + Transformer to show a comparative study between the models performance. Here are the screen shots of the outputs

```
---------------------------------------------
1 ROUGE-1 Average_R: 0.42789 (95%-conf.int. 0.42552 - 0.43057)
1 ROUGE-1 Average_P: 0.31754 (95%-conf.int. 0.31532 - 0.31977)
1 ROUGE-1 Average_F: 0.35087 (95%-conf.int. 0.34893 - 0.35286)
---------------------------------------------
1 ROUGE-2 Average_R: 0.15961 (95%-conf.int. 0.15748 - 0.16186)
1 ROUGE-2 Average_P: 0.11793 (95%-conf.int. 0.11619 - 0.11971)
1 ROUGE-2 Average_F: 0.13036 (95%-conf.int. 0.12860 - 0.13212)
---------------------------------------------
1 ROUGE-L Average_R: 0.38556 (95%-conf.int. 0.38323 - 0.38807)
1 ROUGE-L Average_P: 0.28662 (95%-conf.int. 0.28450 - 0.28872)
1 ROUGE-L Average_F: 0.31644 (95%-conf.int. 0.31451 - 0.31844)

[2020-04-29 23:58:15,812 INFO] Rouges at step 1000
>> ROUGE-F(1/2/3/l): 35.09/13.04/31.64
```

Figure 8: Baseline's result after 1000 iterations

```
----------------------------------------------
1 ROUGE-1 Average_R: 0.46619 (95%-conf.int. 0.46358 - 0.46871)
1 ROUGE-1 Average_P: 0.33326 (95%-conf.int. 0.33103 - 0.33566)
1 ROUGE-1 Average_F: 0.37503 (95%-conf.int. 0.37289 - 0.37709)
----------------------------------------------
1 ROUGE-2 Average_R: 0.19120 (95%-conf.int. 0.18879 - 0.19367)
1 ROUGE-2 Average_P: 0.13740 (95%-conf.int. 0.13549 - 0.13941)
1 ROUGE-2 Average_F: 0.15405 (95%-conf.int. 0.15204 - 0.15604)
----------------------------------------------
1 ROUGE-L Average_R: 0.42317 (95%-conf.int. 0.42066 - 0.42563)
1 ROUGE-L Average_P: 0.30298 (95%-conf.int. 0.30089 - 0.30523)
1 ROUGE-L Average_F: 0.34071 (95%-conf.int. 0.33863 - 0.34272)

[2020-04-30 00:16:14,643 INFO] Rouges at step 1000
>> ROUGE-F(1/2/3/l): 37.50/15.40/34.07
```

Figure 9: Bert-BASE + Classifier after 1000 iterations

```
1 ROUGE-1 Average_R: 0.46325 (95%-conf.int. 0.46067 - 0.46580)
1 ROUGE-1 Average_P: 0.33211 (95%-conf.int. 0.32986 - 0.33449)
1 ROUGE-1 Average_F: 0.37314 (95%-conf.int. 0.37115 - 0.37512)
----------------------------------------------
1 ROUGE-2 Average_R: 0.18954 (95%-conf.int. 0.18725 - 0.19204)
1 ROUGE-2 Average_P: 0.13655 (95%-conf.int. 0.13463 - 0.13850)
1 ROUGE-2 Average_F: 0.15291 (95%-conf.int. 0.15101 - 0.15490)
----------------------------------------------
1 ROUGE-L Average_R: 0.42031 (95%-conf.int. 0.41788 - 0.42291)
1 ROUGE-L Average_P: 0.30182 (95%-conf.int. 0.29973 - 0.30407)
1 ROUGE-L Average_F: 0.33886 (95%-conf.int. 0.33694 - 0.34089)

[2020-04-30 01:02:36,863 INFO] Rouges at step 1000
>> ROUGE-F(1/2/3/l): 37.31/15.29/33.89
```

Figure 10: Bert-BASE + RNN after 1000 iterations

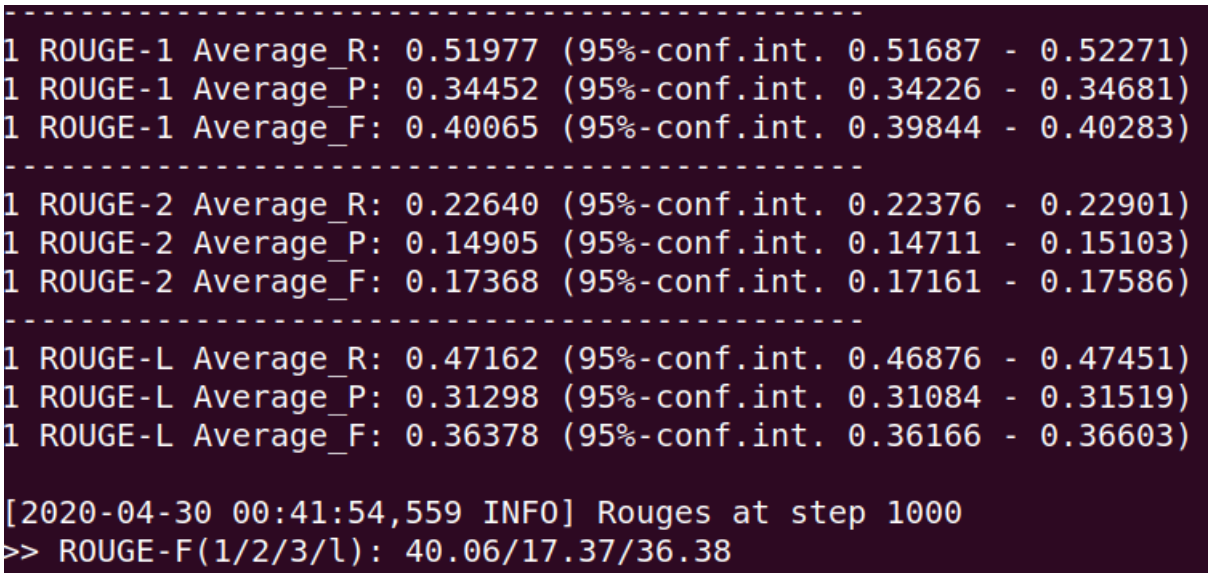


Figure 11: Bert-BASE + Transformer after 1000 iterations

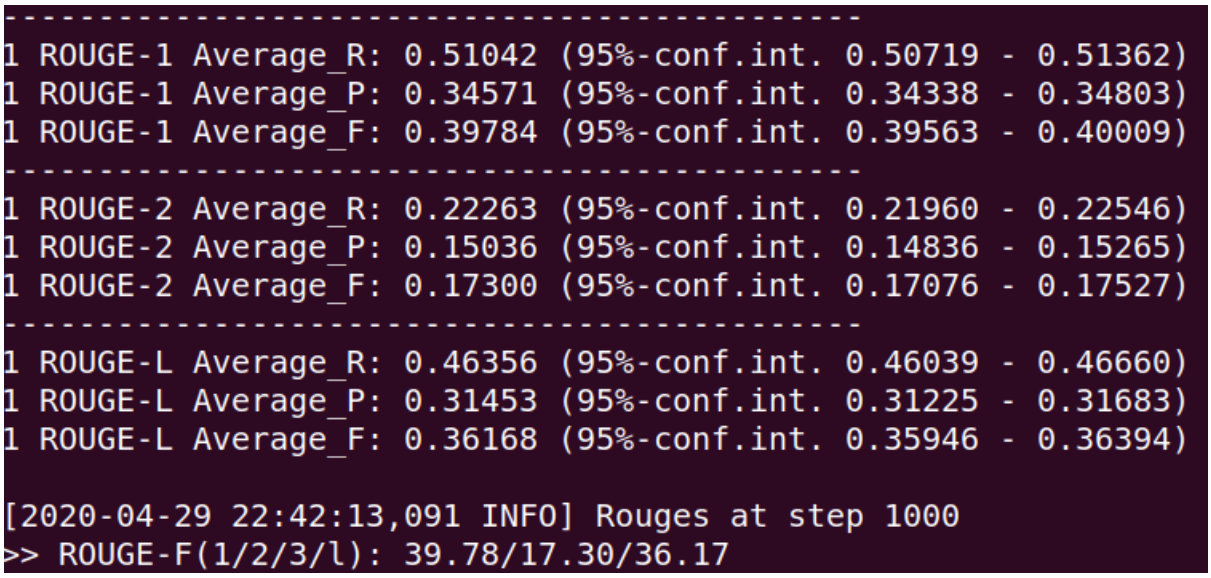


Figure 12: Bert-LARGE + Transformer after 1000 iterations

Results after 1000 steps	ROUGE-1	ROUGE-2	ROUGE-L
Baseline	35.09	13.04	31.63
BERT_BASE + Classifier	37.50	15.40	34.07
BERT_BASE + RNN	37.31	15.29	33.89
BERT_BASE + Transformer	40.06	17.37	36.38
BERT_LARGE + Transformer	39.78	17.30	36.17

Figure 13: Comparison between models after 1000 steps

Comparison after 10000 training steps

We trained all the models except Bert-LARGE till 10000 steps. Here are the screen shots of the outputs

```
1 ROUGE-1 Average_R: 0.50374 (95%-conf.int. 0.50085 - 0.50649)
1 ROUGE-1 Average_P: 0.34283 (95%-conf.int. 0.34057 - 0.34521)
1 ROUGE-1 Average_F: 0.39431 (95%-conf.int. 0.39212 - 0.39652)
---------------------------------------------
1 ROUGE-2 Average_R: 0.21813 (95%-conf.int. 0.21548 - 0.22076)
1 ROUGE-2 Average_P: 0.14875 (95%-conf.int. 0.14675 - 0.15086)
1 ROUGE-2 Average_F: 0.17077 (95%-conf.int. 0.16865 - 0.17294)
---------------------------------------------
1 ROUGE-L Average_R: 0.45797 (95%-conf.int. 0.45514 - 0.46066)
1 ROUGE-L Average_P: 0.31212 (95%-conf.int. 0.30987 - 0.31447)
1 ROUGE-L Average_F: 0.35878 (95%-conf.int. 0.35650 - 0.36097)

[2020-04-30 01:07:07,284 INFO] Rouges at step 10000
>> ROUGE-F(1/2/3/l): 39.43/17.08/35.88
```

Figure 14: Baseline's result after 10000 iterations

```
1 ROUGE-1 Average_R: 0.52250 (95%-conf.int. 0.51977 - 0.52519)
1 ROUGE-1 Average_P: 0.36834 (95%-conf.int. 0.36607 - 0.37076)
1 ROUGE-1 Average_F: 0.41780 (95%-conf.int. 0.41568 - 0.41983)
---------------------------------------------
1 ROUGE-2 Average_R: 0.23876 (95%-conf.int. 0.23596 - 0.24143)
1 ROUGE-2 Average_P: 0.16880 (95%-conf.int. 0.16677 - 0.17077)
1 ROUGE-2 Average_F: 0.19097 (95%-conf.int. 0.18885 - 0.19302)
---------------------------------------------
1 ROUGE-L Average_R: 0.47785 (95%-conf.int. 0.47511 - 0.48044)
1 ROUGE-L Average_P: 0.33750 (95%-conf.int. 0.33527 - 0.33980)
1 ROUGE-L Average_F: 0.38251 (95%-conf.int. 0.38049 - 0.38462)

[2020-04-30 15:53:59,205 INFO] Rouges at step 10000
>> ROUGE-F(1/2/3/l): 41.78/19.10/38.25
```

Figure 15: Bert-BASE + Classifier after 10000 iterations

```
1 ROUGE-1 Average_R: 0.52049 (95%-conf.int. 0.51764 - 0.52326)
1 ROUGE-1 Average_P: 0.37017 (95%-conf.int. 0.36789 - 0.37252)
1 ROUGE-1 Average_F: 0.41820 (95%-conf.int. 0.41596 - 0.42030)
---------------------------------------------
1 ROUGE-2 Average_R: 0.23802 (95%-conf.int. 0.23536 - 0.24072)
1 ROUGE-2 Average_P: 0.16964 (95%-conf.int. 0.16756 - 0.17176)
1 ROUGE-2 Average_F: 0.19122 (95%-conf.int. 0.18905 - 0.19341)
---------------------------------------------
1 ROUGE-L Average_R: 0.47631 (95%-conf.int. 0.47356 - 0.47906)
1 ROUGE-L Average_P: 0.33932 (95%-conf.int. 0.33704 - 0.34160)
1 ROUGE-L Average_F: 0.38307 (95%-conf.int. 0.38093 - 0.38519)

[2020-04-30 16:44:58,557 INFO] Rouges at step 10000
>> ROUGE-F(1/2/3/l): 41.82/19.12/38.31
```

Figure 16: Bert-BASE + RNN after 10000 iterations

```
---------------------------------------------
1 ROUGE-1 Average_R: 0.52445 (95%-conf.int. 0.52185 - 0.52725)
1 ROUGE-1 Average_P: 0.36705 (95%-conf.int. 0.36476 - 0.36957)
1 ROUGE-1 Average_F: 0.41775 (95%-conf.int. 0.41558 - 0.41992)
---------------------------------------------
1 ROUGE-2 Average_R: 0.23888 (95%-conf.int. 0.23611 - 0.24151)
1 ROUGE-2 Average_P: 0.16756 (95%-conf.int. 0.16550 - 0.16971)
1 ROUGE-2 Average_F: 0.19030 (95%-conf.int. 0.18811 - 0.19255)
---------------------------------------------
1 ROUGE-L Average_R: 0.47947 (95%-conf.int. 0.47675 - 0.48222)
1 ROUGE-L Average_P: 0.33614 (95%-conf.int. 0.33395 - 0.33851)
1 ROUGE-L Average_F: 0.38230 (95%-conf.int. 0.38021 - 0.38447)

[2020-04-30 16:18:14,939 INFO] Rouges at step 10000
>> ROUGE-F(1/2/3/l): 41.77/19.03/38.23
```

Figure 17: Bert-BASE + Transformer after 10000 iterations

Results after 10000 steps	ROUGE-1	ROUGE-2	ROUGE-L
Baseline	39.43	17.08	35.88
BERT_BASE + Classifier	41.78	19.10	38.25
BERT_BASE + RNN	41.28	19.12	38.31
BERT_BASE + Transformer	41.77	19.03	38.23

Figure 18: Comparison between models after 10000 steps

Models	ROUGE-1	ROUGE-2	ROUGE-L
Transformer Baseline	40.9	18.02	37.17
BERTSUM+Classifier	43.23	20.22	39.60
BERTSUM+Transformer	43.25	20.24	39.63
BERTSUM+LSTM	43.22	20.17	39.59

Figure 19: Original Results obtained in the paper

Analysing the Results

After analysing the result we derived the following conclusions:

- After 1000 training steps models with transformer performed best which lead us to conclude that transformer capture more information in it's embedding using the BERT sentence embedding. This could also be due to the fact that all the models we under-trained and will still have high bias
- Bert-BASE + Transformer outperformed Bert-LARGE + Transformer after 1000 training steps. This could be due to the fact that Bert-LARGE has higher number of weights to be trained when compared to Bert-BASE
- All the other models except Bert + Transformer improved significantly after training 10000 steps and all the BERT models are quite close in their ROUGE scores which lead us to believe that the BERT output sentence embedding play a very large role in determining the performance of the whole model
- The ROUGE score of our model are below the performance achieved by the paper because the models were trained for 50000 steps with a batch size of 3000 while we trained them for 10000 steps with batch size of 1000. The Bert-LARGE model was trained by us just for 1000 steps with batch size of 100 while the paper trained it for 2000,000 steps with batch size of 3000

0.7 Implementation Details

- Library :We used PyTorch, OpenNMT and the bert-base-uncased version of BERT to implement the model. BERT and summarization layers are jointly fine-tuned.
- GPU :The paper used three GTX 1080ti for their experiment. We have used a single GPU (GTX 1070m: Laptop GPU).
- Solver :Adam with $\beta 1 = 0.9$, $\beta 2 = 0.999$ is used for fine-tuning.
- Batch Size :We reduced batch size from 3000 to 1000 for Bert-BASE and 100 for Bert-LARGE as it overflowed our GPU VRAM.
- Iterations :Paper suggested running the experiment for 50000 iterations. However, due to lack of similar resources (as mentioned in paper) we ran the experiment for 10000 iterations for Bert-BASE models and only for 1000 iterations for Bert-LARGE model. However, our close proximity with the results obtained in paper suggests we would achieve similar results by running it for the full duration.

0.8 Discussion

- **Bert+Transformer**:The BERT+Transformer model form the attention based sentence level context which has performed with best when it's the BERT-LARGE model paired with a transformer
- **BERT+RNN**:We form sentence level context just like RNN have with words however the long inter-sentence relationships cannot be preserved in a RNN in the same way as in transformer which uses attention based mechanism
- **Looking at the Results**:The results of BERT+simple Classifier, BERT+RNN , BERT+Transformer are close. One of the reason can be that the sentence embedding of BERT models are rich enough to produce good result. The BERT+Classifier directly use the BERT output while the other two add more sentence level context embeddings as suggested above however the difference in performance is quite low. This could suggest that using BERT models of higher layer could perform even better. The state-of-the-art result obtained for extractive summarization is currently held by BERT-LARGE + Transformer model (which we could only train for 1000 steps)

0.9 Limitations

- Our model was under trained when compared to the original model mentioned in the paper which leads to under-performance
- The cause of under training were the hardware at our disposal. To train a BERT-LARGE model to the same level as in the paper we would need at least 24 GBs of VRAM while we had 8 GBs at our disposal
- When commenting on the model itself we can say that heavy reliance on hardware can be seen as a limitation though easily met in a research institute
- Since all 3 models BERT + Classifier/RNN/Transformer achieved almost similar result. We can conclude that the overall model is heavily reliant on the BERT model which is a limitation as it could be said that the contribution of models placed on top of BERT is quite limited
- We are completely ignoring the word embeddings obtained by the BERT model and only using the sentence level embedding for our final evaluation

0.10 Future Scope

- Training the best Model BERT+Transformer over different datasets to better train the model
- Finding a way to incorporate the word embeddings output by the BERT model and using them to further improve the performance
- Trying out different length of input tokens and see their impact on the performance of the model
- Since all 3 models, BERT + Classifier/RNN/Transformer are in close proximity when it comes to performance, suggesting heavy reliance on BERT output embeddings, working on improving the BERT model (more layer, training on different datasets) can be an interesting way to approach the problem
- Coupling the BERT output embeddings with different models or a combination of models and checking their performance

0.11 References

- http://jalammar.github.io/illustrated-bert/
- http://jalammar.github.io/illustrated-transformer/
- https://pytorch.org/tutorials/
- https://rxnlp.com/how-rouge-works-for-evaluation-of-summarization-tasks/#.Xqvp7N9fgRY
- https://arxiv.org/pdf/1903.10318.pdf
- https://arxiv.org/pdf/1908.08345.pdf
- https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-r
- https://www.youtube.com/watch?v=iDulhoQ2pro