Abstractive and Extractive Summarization with BERTSum and T5

Luke Schanne
lschanne@berkeley.edu

Kritesh Shrestha
kkshrestha@berkeley.edu

Abstract

Summaries are used in an attempt to convey an article's message and meaning at a glance. In this paper, we use BERTSum and T5 models to establish baseline performances for extractive and abstractive summarization tasks on the the *CNN/DailyMail* dataset (See et al., 2017). We create a hybrid model that feeds extractive summaries from the BERTSum model into a T5 model to produce abstractive summaries. Our new model has better performance on the abstractive summarization task than the baseline T5 model, but the ROUGE scores from the BERTSum extractive summaries remain the best.

1 Introduction

Text summarization attempts to take an input document and express the most important information in a more succinct output. Generally, there are two main types of text summarization: extractive and abstractive. Extractive summarization extracts the output summary using exact words and phrases from the input document. Abstractive summarization uses generative techniques to produce the output summary using potentially new words and phrases, as a real human might. Historically, the field has been dominated by extractive techniques due to easier implementation and better performance. Recently, there have been many advances in abstractive techniques that have made them fairly competitive.

We want to explore some state-of-the-art extractive and abstractive techniques to compare their performance on the *CNN/DailyMail* dataset (See et al., 2017). Then, we will try combining extractive and abstractive techniques to improve performance. To assess performance, we use Recall-Oriented Understudy for Gisting Evaluation (ROUGE) scores, which are the standard for automatic evaluation metrics in summarization. Because it is difficult to fully capture all of the criteria for a good summarization with an automatic evaluation metric, performance assessment should be accompanied by manual human evaluation of output summaries. However, given our limited resources, we have limited our evaluation to the automated ROUGE scores.

2 Background

Today, many state-of-the-art solutions to text summarization implement the pre-trained BERT transformer to perform encoding (Liu and Lapata, 2019; Bae et al., 2019; Zhong et al., 2019; Zhang et al., 2019).

One such solution is the BERTSum (Liu, 2019), which performs extractive summarization by starting with word embedding via BERT. The input documents are modified to place [CLS] and [SEP] tokens before and after each sentence so that the BERT encodings can be used to distinguish full sentences, which are the typical unit of extraction in extractive summarization (Liu, 2019). After generating encodings, the BERTSum model used several methods for building the summarization (Liu, 2019). The best-performing method used a 2-layer transformer with multi-head attention to determine the sentences for extraction based on the encodings from BERTSum (Liu, 2019). This method achieved ROUGE-1 43.25, ROUGE-2 20.24, and ROUGE-L 39.63 on the *CNN/DailyMail* dataset (Liu, 2019; See et al., 2017).

In an attempt to improve upon our extractive BERTsum performance introduce T5 as our abstractive model. The T5 model (Raffel et al., 2019) is a ”Text-to-Text Transfer Transformer” which can be used for many NLP tasks due to its focus on transfer learning. For our task of abstractive summarization, T5 has also achieved strong results comparable to models that are tuned for summarization. As T5 is a transfer learning model, all tasks require the model text with the task as an input. For our case, our inputs would require *"summarize:"* in front of the input text (Raffel et al., 2019). The T5 model starts with input sequence of tokens which is mapped to an embedding sequence that is then passed to an encoder. Within the encoder there is a self-attention layer and a small feed-forward network. The Encoder is the normalized and gets its weights and dropout parameters added to it which feeds to the decoder. The decoder works similar to the encoder with an additional standard attention mechanism after each self attention layer (Raffel et al., 2019). In the study conducted by Raffel et. al., the authors were able to achieve ROUGE-1-F/ ROUGE-2-F/ ROUGE-l-F scores of 43.52/21.55/ 40.69 on the *CNN/DailyMail dataset* which we intend to replicate (Raffel et al., 2019).

We are also very interested in hybrid approaches that combine extractive and abstractive methods. An approach by (Pilault et al., 2020) first uses a hierarchical seq2seq sentence pointer to extract key sentences from the input document before feeding those sentences into an a transformer language model (TLM) to produce a final abstractive summary. This approach did fairly well in ROUGE-1 and ROUGE-2 scores compared to other abstractive and hybrid techniques, but did not do as well in ROUGE-L scores (Pilault et al., 2020).

There was also a very similar two-stage hybrid model produced by feeding BERTSum extractive summaries into a 6-layer transformer and decoder stage to produce abstractive summaries (Liu and Lapata, 2019). Our own hybrid model approach mainly differs in the second stage, where we use a T5 abstractive summarization model. Their BERTSumExtAbs model achieved lower ROUGE scores on the *CNN/DailyMail* dataset (See et al., 2017) than their BERTSum extractive model but achieved higher scores than other abstractive models (Liu and Lapata, 2019).

3 Methods

We use the *CNN/DailyMail* dataset (See et al., 2017) to train and test our models. The dataset has a total of 311,971 input documents and target summaries. The input documents are new articles and the target summaries are joined highlights. We use the train, validation, and test split as provided by TensorFlow, with 11,490 documents allocated to the test set, 287,113 documents allocated to the training set, and 13,368 documents allocated to the validation set.

BERTSum First, we implement the BERTSum + Transformer (Liu, 2019) and T5 (Raffel et al., 2019) models on the *CNN/DailyMail* dataset (See et al., 2017) to verify that we achieve results close to those published with the original models. Then, we use the extractive summaries from the BERTSum + Transformer (Liu, 2019) as inputs to train a new version of the T5 (Raffel et al., 2019) model. We compare performance of the layered hybrid approach to the baseline models. With this setup we are hoping to generate a headline that extracts key words from the article then removes unnecessary words while keeping the headline legible.

For our implementation, we use the BERTSum extractive summarizer from the PreSumm github repository (Liu and Lapata, 2019). We ran for 50,000 epochs with a learning rate of .002, a dropout rate of 0.1, a batch size of 500, a max input length of 512, and a gradient accumulation count of 2. We also forked the original implementation and made some small changes to make it compatible with Pytorch 1.9.1 so that we could run the code using GPUs with current Nvidia drivers. From testing on the validation set, the model after 45,000 training epochs is selected as the best.

T5 Our T5's implementation is based on the transformers library from Huggingface. We use the T5tokenizer to prepare inputs for our model and T5forConditionalGeneration transformer (t5-base) model to train and generate our summaries. Before we fed data to our model we processed it through a dataloader to be able to pass our data in batches in training and validation steps. Before adding the dataloader our models would crash with memory errors due to our data set's size being to large to load to memory at once. Our T5 model ran for 1 epoch on training and testing with a learning rate of .0001, batch size of 2, max input length of 512, min length of 150 and target length of 150. Our epoch size, learning rate and batch size stay consistent in our T5 portion since our instance would only run those values.

We use an optimizer with this model which tunes the weights of the network in the training portion. Specifically we used the Adaptive Moment Estimation (Adam) optimizer as it is a replacement optimization algorithm for stochastic gradient descent for training deep learning models. We chose the Adam optimizer as it's performance with neural networks was favorable with little memory requirements (Kingma and Ba, 2017). In our validation step we set our max length was set to 150, with a target length of 30, number of beams for beam-search was set to 2, and a repetition penalty of 2.5 was added to reduce word repetition. Our initial max length is based on the average source article length we decide to decrease the max length in our validation step to encourage shorter summaries.The model ran on pytorch to take advantage of GPUs which helps reduce time to finish running the model.

Hybrid Our ”hybrid” model is a two-stage model, with the first stage being the exact BERTSum extractive model that we use for a baseline. Stage two uses extractive summaries from stage one as inputs to a T5 model that produce abstractive summaries. Our ”initial hybrid” model had the T5 model in stage two using the same parameters as the baseline T5 model, but the model weights are different since the inputs are different. Due to time and resource constraints we spent a short amount of time tuning our baseline T5 model to achieve the better results. We tuned T5 during our hybrid step which provided marginally better results. We reduce the max length to 30, since our inputs were the out puts from our BERTSum model. The min length was reduced to 5. We also added a length penalty of 0.5 to encourage the model to produce shorter sequences.

Ensemble Our final, ”ensemble” model is built on all three of the prior models. We look at the ROUGE scores on the validation set as a function on the number of tokens in the input document. We use the tokenizer for the T5 model to determine the number of tokens in the input document. We then select one of our three models to generate the output summary based on the ROUGE score performance of the models on documents with a similar number of tokens in the validation set. Unfortunately, the BERTSum model has consistently better ROUGE scores at every token length, so our ensemble model degenerates into the BERTSum model.

4 Results and Discussion

BERTSum With the implementation of BERTSum (Liu and Lapata, 2019), we achieve ROUGE-F(1/2/l) scores of 42.31/19.51/38.70 and ROUGE-R(1/2/l) scores of 53.75/24.80/49.13. The authors achieved very similar scores of 43.23/20.24/39.63 for ROUGE-F(1/2/l) on the same *CNN/DailMail* dataset (Liu and Lapata, 2019), so we have confidence in our implementation of the model as a strong baseline for extractive summarization.

T5 Our initial T5 implementation achieved ROUGE-F(1/2/l) of 22.86/2.70/14.62 which is far from the scores presented by Raffel et. al. 43.52/21.55/ 40.69 (Raffel et al., 2019). However with limited tuning we were able to increase our scores by changing our max length and target length from the default. Our baseline T5 model achieved ROUGE-F(1/2/l) of 32.15/10.51/20 which is a great improvement over the initial implementation.

Hybrid Our initial hybrid model achieves ROUGE-F(1/2/l) scores of 28.53/7.00/21.39 and ROUGE-R(1/2/l) scores of 29.10/7.32/22.02, consistently outperforming our baseline T5 abstractive model. However, it does not perform better than the baseline BERTSum extractive model. We further tune the T5 portion to achieve marginally better results by reducing max length, min length, and adding a length penalty. With these changes, our hybrid model achieves ROUGE-F(1/2/l) scores of 27.51/9.36/21.56 and ROUGE-R(1/2/l) scores of 31.04/10.43/30.92.

Stacking the two models seems to improve the abstractive summaries produced by our hybrid model as compared to the baseline T5 model. However, our only metric is the automated ROUGE scores, and both stages of the hybrid model are trained against the same pre-generated summaries. Since we did not have the resources for manual human evaluation of our summaries, it is possible that the hybrid model is over performing on ROUGE scores at the cost of readability or sensibility. The BERTSumExtAbs two-stage model from (Liu and Lapata, 2019) also had lower ROUGE scores on the same dataset as compared to their BERTSum extractive model, but their reported ROUGE-F(1/2/l) scores of 42.13/19.60/39.18 were much higher than our own (Liu and Lapata, 2019). These results suggest that the 6-layer transformer second stage produces better results than the T5 second stage. However, we did not have the opportunity to try many different hyper-parameters, so there are likely many performance improvements possible for our T5 stage (Liu and Lapata, 2019).

Ensemble The ensemble method looks at the ROUGE scores of the previous three models as they vary with the number of tokens in the input document. Our hope was to take an input document, calculate the number of tokens in the document, and pick an output summary from one of the three models based on sum token number cutoff points that we could determine in our analysis of ROUGE scores on the validation dataset. However, at every token length, the BERTSum extractive model outperforms the other two models, so the ensemble model degenerates into a copy of the BERTSum extractive model. Therefore, it achieves the same ROUGE scores.

	ROUGE-F			ROUGE-R		
	R-1	R-2	R-l	R-1	R-2	R-l
Baseline BERTSum	42.31	19.51	38.70	53.75	24.80	49.13
Baseline T5	22.86	2.70	14.62	17.63	2.09	11.30
T5	32.14	10.15	20.00	57.28	19.18	36.00
Initial Hybrid	28.53	7.00	21.39	29.10	7.32	22.02
Hybrid	27.52	9.36	21.56	30.92	10.27	24.24

Table 1: R-1 is the uni-gram overlap. R-2 is the bi-gram overlap. R-l is the longest common sub-sequence. ROUGE-F scores measure the f1-scores, defined as $\frac{2*precision*recall}{precision+recall}$, of the n-grams. ROUGE-R scores measure the recall of the n-grams.

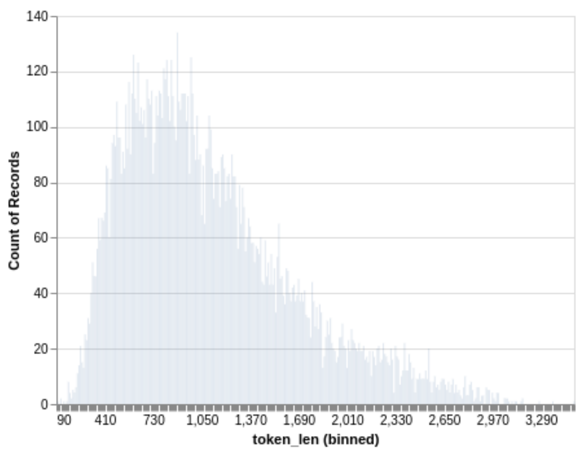

Figure 1: Distribution of the number of tokens per document.

In Figure 1, we show the histogram of the number of tokens per input document in the validation set. We see that the majority of the input documents have less than 1,500 tokens. In Figure 2, we show the ROUGE scores of the three models as a function of the number of tokens in the input document. We see that the BERTSum extractive model achieves higher ROUGE scores than the other two models at every token length. However, the ROUGE-F scores become very close for the three models as the number of tokens increases. Specifically, the BERTSum does not perform as well as the number of tokens increases, while the two abstractive models have a more consistent performance across document length. These results may suggest extractive summarization tasks are better suited to smaller documents.

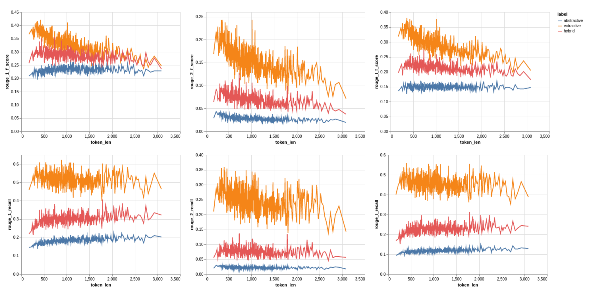

Figure 2: ROUGE scores of models as a function of the number of input tokens in the document. The top row, from left to right, is the ROUGE-F 1, 2, and l scores. The bottom row, from left to right, is the ROUGE-R 1, 2, and l scores. The orange lines are the BERTSum extractive model scores. The red lines are the hybrid model scores. The blue lines are the T5 abstractive model scores.

Limitations Unfortunately, much of our methodology is limited by resource constraints. Our limited budget did not allow for large GPUs, which meant that it took much longer to train the models. Additionally, the t5-large model did not fit into the GPU memory during training even with a batch size of 1, so the models using T5 were limited to t5-base or t5-small. Even using the t5-base model, we were only able to fit two concurrent samples into the GPU memory during training, so training even a single epoch of the T5 model took upwards of ten or twelve days. Given the long training and limited time constraints for the project, we were only able to run one or two iterations of the T5 abstractive model and the hybrid model. This limitation meant that we had very little room to experiment with hyper-parameters. With all of these considerations in mind, it is very likely that we left a lot of performance improvements on the table with the T5 abstractive model, the hybrid model, and the ensemble model.

Examples In addition to our analysis, looking at examples of our output summaries from BERTSum and Hybrid model consistently verifies BERTSum's superior performance. Even though there are instances where the Hybrid model's summaries closely match BERTSum, it never seem to be an improvement BERTSum's output.

BERTSum: *premier league outfit west ham are closing in on the signature of jamaican starlet deshane beckford.*

Hybrid: *jamaican starlet deshane beckford is close in on* *the signature of derjasian international player dejano signed from."*

BERTSum: *hamburg have appointed bruno labbadia as their new coach on a month contract.*

Hybrid: *bruno labbadia has been appointed as hamburg new coach on month contract deal until end of the season and will take over from mat.*

BERTSum: *danny rose has attracted interest from manchester city after a strong season with tottenham.*

Hybrid: *danny rose has attracted interest from manchester city after a strong season with tottenham hotspur and spurs this summer.*

5 Conclusion

Of the models we tested, the BERTSum extractive model has the best ROUGE score performance across the board. By combining the BERTSum and T5 models into a two-stage hybrid model, we produce abstractive summaries with better ROUGE scores than the baseline T5 abstractive model. Because the BERTSum extractive model has better ROUGE scores for documents of all lengths, using an ensemble model that chooses the model based on document length degenerates into a copy of the BERTSum model. However, the ROUGE score performance of the BERTSum extractive model becomes much closer to the ROUGE score performance of the hybrid models as document length increases. Due to limited GPU capacity and time, our abstractive and hybrid models are probably under-performing.

References

Sanghwan Bae, Taeuk Kim, Jihoon Kim, and Sang goo Lee. 2019. Summary level training of sentence rewriting for abstractive summarization.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Yang Liu. 2019. Fine-tune bert for extractive summarization.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders.

Jonathan Pilault, Raymond Li, Sandeep Subramanian, and Chris Pal. 2020. On extractive and abstractive neural document summarization with transformer language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9308–9319, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. *CoRR*, abs/1704.04368.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization.

Ming Zhong, Pengfei Liu, Danqing Wang, Xipeng Qiu, and Xuan-Jing Huang. 2019. Searching for effective neural extractive summarization: What works and whats next. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 1049–1058.