Generate Concise Content: Text Summarization with BERT

Xinyu Lu
University of Michigan
luxinyu@umich.edu

Yeeun Jang
University of Michigan
yejang@umich.edu

1 Introduction

Text summarization in Natural Language Processing is the task to generate a fluent and concise summary of long texts. Nowadays, there is an exploding amount of text data. The tool for generating concise content and reducing the size of the text has high practical importance in the big data era (Huang et al., 2020). A good summary contains the important information without losing the overall meaning of the original text document. Text summarization is a comprehensive task, which requires both language understanding and text generation.

Text summarization mainly falls into two paradigms: extractive summarization (Dorr et al., 2003; Mihalcea and Tarau, 2004; Nallapati et al., 2016) and abstractive summarization (Jing and McKeown, 2000; Rush et al., 2015; See et al., 2017). Extractive approaches identify and concatenate the most representative sentences in a document, where the sentences are picked based on scoring (Wong et al., 2008; Liu, 2019). While abstractive approaches rephrase and rewrite the original text to generate a new shorter text (Ganesan et al., 2010). Although the ROUGE(Lin, 2004) scores have significantly improved with the development of deep learning techniques, there are still large gaps between manual text summarization and automatic text summarization because computers lack human knowledge and understanding of the texts.

Pre-trained language models have improved the performance of a variety of NLP tasks. Liu and Lapata (2019) explored the potential of Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) for text summarization under both extractive and abstractive categories and proposed a document-level encoder based on BERT.

In this project, we aim to push the limit of text summarization with BERT by two approaches. In the first approach, we examine the effect of data preprocessing on the quality of summary. We apply coreference resolution to the original dataset before feeding the text into the model for summarization (Liu and Lapata, 2019). We evaluate the approach on the CNN/DailyMail dataset (Hermann et al., 2015). In the second approach, we try to improve the performance through a new model architecture. We combine BERT (Devlin et al., 2018) with pointer generator layer (See et al., 2017). Originally, See et al. (2017) uses Sequence-to-Sequence as an encoder, and LSTM as decoder. We replace both encoder and decoder with more advanced structures. Specifically, we replace the Sequence-to-Sequence encoder with BERT and LSTM decoder with Transformer (Vaswani et al., 2017b) decoder. We use CNN/DailyMail dataset for training.

We find out that applying coreference resolution helps extractive summarization while there is no significant improvement on the abstractive summarization with BERT. Replacing the referring nominal expressions enables the model to extract the most important information from the documents. We also find out that our suggestion of replacing the encoder of pointer-generator with BERT and decoder with Transformer decoder did not make an improvement in performance.

Code is available here[1].

2 Related Work

2.1 Extractive Summarization

Extractive summarization identifies and concatenates the most representative sentences in a document. Early works (Neto et al., 2002; Mihalcea and Tarau, 2004) manually define rules by using expertise knowledge, which are statistical meth-

[1] https://github.com/SandyLuXY/text-summarization-with-bert

ods. While recent neural architectures consider extractive summarization as a word or sentence classification problem (Nallapati et al., 2016; Xu and Durrett, 2019). An encoder learns sentence representations and a classifier selects the most important sentences.

2.2 Abstractive Summarization

Early efforts (Jing and McKeown, 2000) create abstractive summarization by editing and merging extracted pieces into a whole sentence, which is based on cut-paste. Recent neural methods (Rush et al., 2015) view abstractive summarization as a sequence-to-sequence task. An encoder maps the tokens in a document to their corresponding representations and a decoder generates every token in the summary. Hybrid method combines extractive and abstractive summarization, which is a two-step method including content selection and text generation (Huang et al., 2020).

2.3 Pre-trained Language Models

The emergence and evolution of pre-trained language models (Peters et al., 2018; Devlin et al., 2018; Zhang et al., 2019) have led to impressive achievements in a wide range of NLP tasks. The pre-trained language models learn representations by training under certain language modeling objectives. They are trained on gigantic corpus, which enables the pre-trained language models to contain rich information of words.

BERT (Devlin et al., 2018) is trained with masked language modeling and next sentence tasks on English Wikipedia (2,500M words) and BooksCorpus (800M words) (Zhu et al., 2015). It combines word and sentences representations in a single Transformer (Vaswani et al., 2017a) and has been used for sentence-level and paragraph-level natural language understanding problems.

2.4 Summarization Encoder Based on BERT

Liu and Lapata (2019) proposed a document-level encoder BERTSUM for text summarization. Figure 1 shows the comparison of the architecture of BERT and BERTSUM. In the original BERT, [CLS] symbol is inserted at the start of the input, whose output represents the aggregation information. [SEP] symbol is inserted at the end of the a sentence for separation. There are three different embeddings assigned to a token. Token embeddings represent the meaning, segmentation embeddings distinguish the inputs in a given pair for sentence-pair classification, and position embeddings contain the information of position of the token (Devlin et al., 2018).

While for BERTSUM, external [CLS] tokens are inserted at the beginning of each sentence to collect the information of each sentence. The red and green interval segmentation embeddings are used to differentiate sentences. Document representations are learned in a hierarchical way that lower Transformer layers learn the representations of adjacent sentences and higher Transformer layers learn the representations of multiple sentences (Liu, 2019).

2.5 Coreference Resolution

Coreference resolution is the problem of figuring out which noun phrases refer to each real-world entity mentioned in the text (Ng and Cardie, 2002). To resolve the coreferences, we replace the pronouns with noun phrases. Coreference resolution can be applied to many NLP tasks, for example text understanding, machine translation, information extraction and sentiment analysis. It helps obtain unambiguous sentences and makes it easier for computers to understand sentences (Beheshti et al., 2017).

Among the various tools to deal with coreference resolution, SpaCy (Honnibal and Montani, 2017) provides fast coreference resolution module based on spaCy parser and the neural net scoring model introduced in the paper *Deep Reinforcement Learning for Mention-Ranking Coreference Models* (Clark and Manning, 2016).

2.6 Pointer Generator

Although sequence-to-sequence models have provided a new approach for abstractive text summarization, the approach had problems: low quality of factual detail reproduction, and inclination of repeating themselves (See et al., 2017). Therefore, an abstractive summarization method leveraging the advantage of extractive summarization was proposed. The model uses a hybrid pointer-generator network that can copy words from the source text via pointing. The model employs sequence-to-sequence for encoder and LSTM for decoder. This enables accurate reproduction of information while retaining the ability to generate new words.

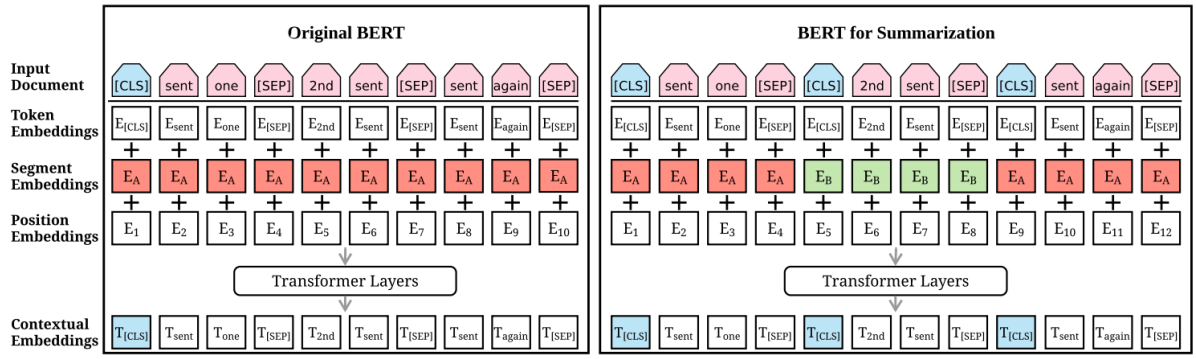

Figure 1: Model Architecture of original BERT and BERTSUM

3 Approaches

3.1 Coreference Resolution + Fine-tune BERT

For this approach, we first preprocessed the data by applying the coreference resolution to the input text. We took advantage of the coreference resolution module NeuralCoref 4.0[2] provided by spaCy, which is a coreference resolution tool with neural networks. For example, if the original text is *Deepika has a dog. She loves him*, then the processed text after coreference resolution will be *Deepika has a dog. Deepika loves a dog*. The pronouns *she* and *him* are replaced with the real entity *Deepika* and *a dog*. We fine-tuned the abstractive BERTSUM model using the processed data and generate the summaries. We evaluated the performance of our fine-tuned model under ROUGE-1, ROUGE-2 and ROUGE-L metrics. See the details of these metrics in Section 4.

In addition, we would like to compare extractive summarization with abstractive summarization. We also fine-tuned the extractive BERTSUM model and compared the results with the abstractive method.

Model We used the abstractive BERTSUM model for fine-tuning. Additionally, we used the extractive BERTSUM model for the additional comparison. For the abstractive BERTSUM model, it is a encoder-decoder framework in See et al. (2017). The pre-trained BERTSUM model is used as the encoder, and a six-layer Transformer with random initializations is used as the decoder. In particular, two Adam optimizers with different warmup steps and learning rates are applied for the encoder and decoder. For the extractive BERTSUM model, it is to assign a binary label to each sentence in a document, which indicates whether the summary contains the sentence. In order to capture the features of document level, there are multiple stacked inter-sentence Transformer layers on the output of BERT (Liu and Lapata, 2019).

Dataset The dataset we used for this approach was the CNN/DailyMail dataset (Hermann et al., 2015). It consists of news articles and associated highlights of brief overviews, which is a widely-used dataset in the text summarization task. Abstractive summary bullets are generated in CNN and Daily Mail websites from news stories, and stories as the corresponding passages from which the system is expected to answer the question. There are 286,817 training pairs, 13,368 validation pairs and 11,487 test pairs in the corpus.

3.2 BERT + Pointer-Generator

We will take the model proposed in *Get To The Point: Summarization with Pointer-Generator Networks* (See et al., 2017) as the baseline model. It adds pointer-generator layers on sequence-to-sequence (seq2seq)(Sutskever et al., 2014) encoder and LSTM(Hochreiter and Schmidhuber, 1997) decoder. We employ the pointer generator layer, which contains a generation probability, from our baseline model and replace the encoder with BERT and decoder with that of Transformers. We evaluate the approaches using ROUGE-1, ROUGE-2 and ROUGE-L metrics.

Context vector h_t^* is calculated by a_t as follows:

$$h_t^* = \Sigma_i a_i^t h_i$$

where a_t is a sum of multi-head-attention of encoder and decoder, and h_t is averaged along the dimension of the encoder output. The generation

[2] https://github.com/huggingface/neuralcoref

probability $p_{gen} \in [0, 1]$ is calculated using context vector h_t^*, decoder input x_t, and decoder hidden state s_t. Vectors w_h, w_s, w_x, and scalar b_{ptr} are learned through training process.

$$p_{gen} = \sigma(w_h^T h_t^* + w_s^T s_t + w_x^T x_t + b_{ptr})$$

p_{gen} serves as a switch to choose whether to generate words from the vocabulary P_{vocab} or to copy words from the input sequence from a_t. With calculated p_{gen}, the model generates a new word from the vocabulary distribution P_{vocab} :

$$P_{vocab} = softmax(V(V'[s_t, h_t^*] + b) + b')$$

where V, V' and b' are learnable parameters. P_{vocab} is a probability distribution over all words in the vocabulary, and also can be explained as the softmax of the transformer decoder output. The structure of the model is represented in Figure 2.

Predicted words $P(w)$ is determined as following :

$$P(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \Sigma_{i:w_i=w} a_i^t$$

4 Evaluation

To evaluate and compare the results we got from different models, we used Rouge-1, Rouge-2, and Rouge-L scores. ROUGE (Recall-Oriented Understudy for Gisting Evaluation) (Lin, 2004) score is one of the most commonly used automatic evaluation metrics for text summarization and machine translation. The metrics compare summary or translation generated by model against a set of reference sentences. ROUGE-1 refers to the overlap of unigram between the system and reference sentences. ROUGE-2 refers to the overlap of bigrams between the system and reference sentences. ROUGE-L is a Longest Common Subsequence (LCS) based statistics. Longest common subsequence problem considers sentence level structure similarity and identifies longest co-occurring in sequence n-grams automatically.

4.1 Coreference Resolution + Fine-tune BERT

Setup For the abstractive summarization with BERTSUM, separated optimizers are used for the BERTSUM encoder and Transformer decoder. The parameter setting is $\beta_1 = 0.9, \beta_2 = 0.999$. β_1 and β_2 are the exponential decay rates for the first moment estimates and second moment estimates respectively. The warmup steps and learning rates for the encoder are

$$lr_E = \tilde{lr}_E \cdot min(step^{-0.5}, step \cdot warmup_E^{-1.5})$$
$$\tilde{lr}_E = 2e^{-3}$$
$$warmup_E = 20000$$

The warmup steps and learning rates for the decoder are

$$lr_D = \tilde{lr}_D \cdot min(step^{-0.5}, step \cdot warmup_D^{-1.5})$$
$$\tilde{lr}_D = 0.1$$
$$warmup_D = 10000$$

Other hyperparameters are

$$dropout = 0.2$$
$$batchsize = 140$$

For the extractive summarization, the loss is the binary entropy loss. The optimizier is the Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.999$, and

$$lr = \tilde{lr} \cdot min(step^{-0.5}, step \cdot warmup^{-1.5})$$
$$\tilde{lr} = 2e^{-3}$$
$$warmup = 10000$$

Other hyperparameters are

$$dropout = 0.1$$
$$batchsize = 3000$$

Training Results For the training of abstractive summarization, we report the accuracy, model perplexity and cross entropy loss. The accuracy is defined as the percentage of the correctly predicted words, which is calculated as the number of correct words divided by the total number of words. The model perplexity is used to decide the best combination of the learning rates of the encoder and decoder in abstractive summarization.

Figure 3-5 show the accuracy, model perplexity and cross entropy loss during the training of abstractive summarization. The training accuracy is quite vibrating but increases steadily. After around 450 training steps, there is no obvious trend of increasing. The finalized accuracy score is around 70%. The model perplexity is huge at the beginning, which is over 20000. It is on a fast decreasing trend and the final model perplexity is less than 3. The cross entropy loss decreases quickly at the beginning of the training and the decreasing speed

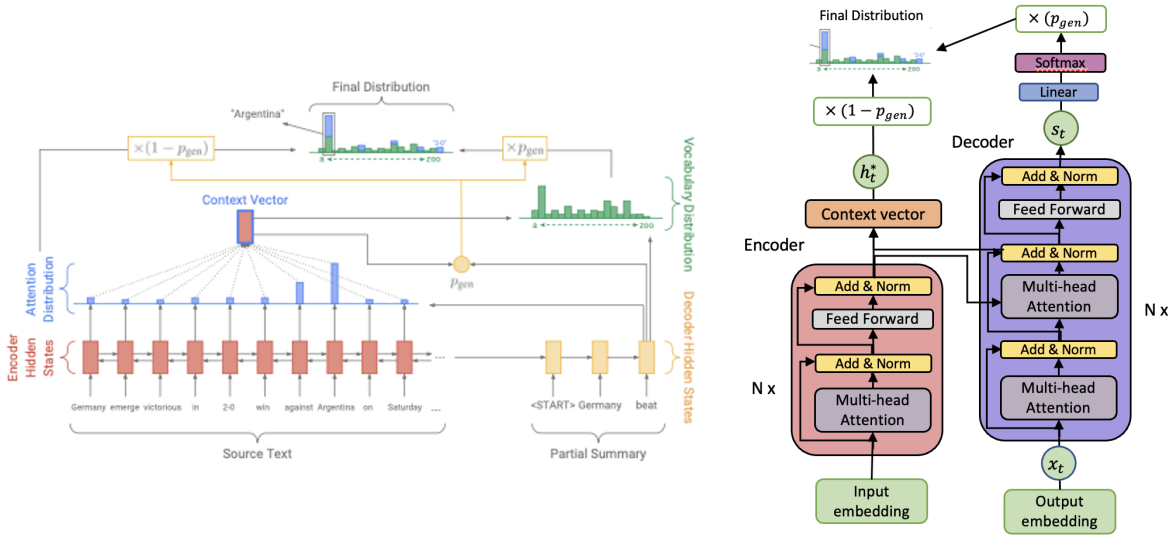

Figure 2: Structure of Pointer Generator and BERT Pointer Generator

lowers down as the training proceeds. After training for 500 steps, the cross entropy loss is steadily around 0.8. For the training of extractive summarization, we report the cross entropy loss.

Figure 6 shows the cross entropy loss during the training of extractive summarization. Different from the cross entropy loss in the abstractive summarization, the cross entropy loss under extractive setting first drops very fast, then slowly decreases for around 100 training steps. After training for about 150 steps, it decreasse faster for approximately 100 steps, and then on a slow decreasing trend again. The final converged cross entropy loss is very close to zero, which is around 0.02.

Evaluation Results For the evaluation, we report the ROUGE-1, ROUGE-2 and ROUGE-L scores. Table 1 shows the ROUGE scores for evaluating abstractive summarization and extractive summarization, and also the baseline results from Liu and Lapata (2019). Compared with the baseline abstractive BERTSUM, our approach slightly improves the ROUGE-2 score by 0.02, and the ROUGE-L score by 0.03, which the ROUGE-1 score decreases by 0.02. Compared with baseline extractive BERTSUM, our approach makes a reletively greater improvement, where the ROUGE-1 score increases by 0.63, the ROUGE-2 score increases by 0.22, and the ROUGE-L score increases by 0.18.

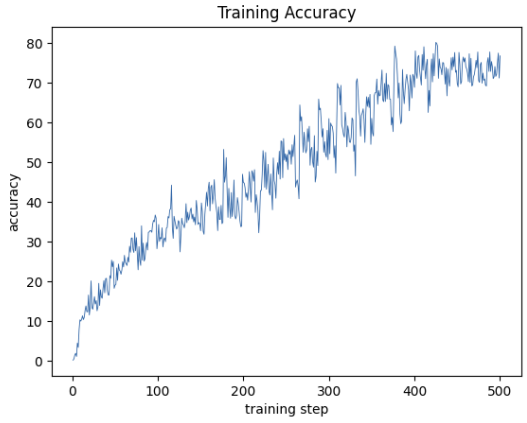

Figure 3: Training Accuracy of Abstractive BERTSUM

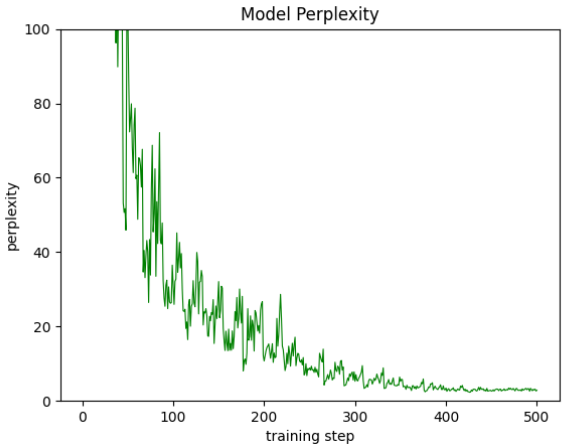

Figure 4: Model Perplexity of Training Abstractive BERTSUM

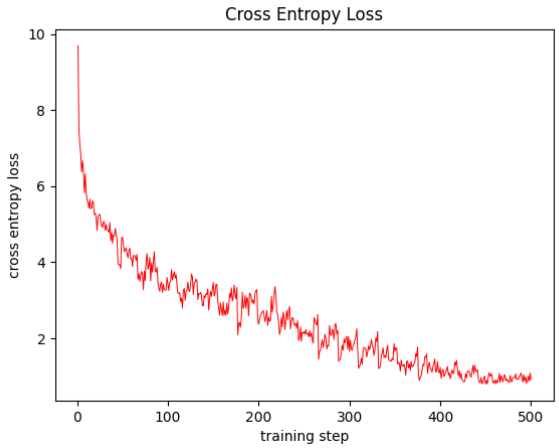

Figure 5: Cross Entropy Loss of Training Abstractive BERTSUM

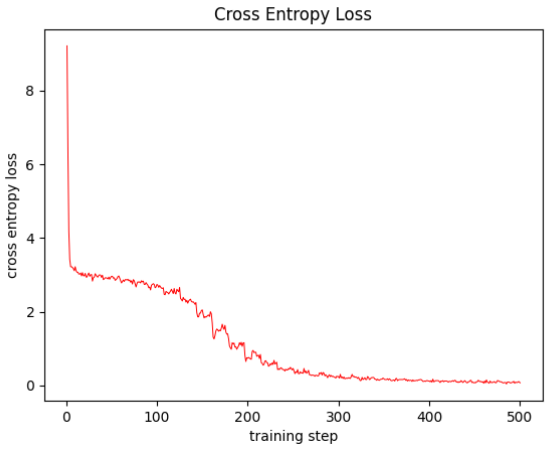

Figure 6: Cross Entropy Loss of Training Extractive BERTSUM

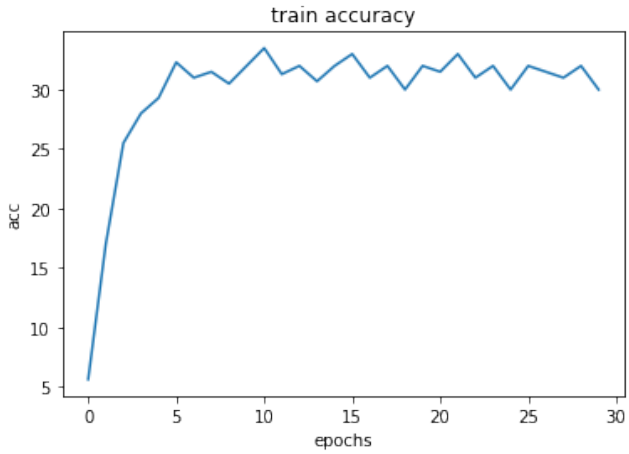

Figure 7: Training accuracy of BERT-Pointer-Generator

4.2 BERT + Pointer-Generator

Setup For our second approach, we divide train, validation, and test data by document id for each article as the CNN/DailyMail dataset we used is consisted of pairs of news article and corresponding headline.

For combined pointer generator model, we used 8 heads for the transformer's multi-headed-attention, and 6 layer transformer. We set vocab size as 50000, dropout rate as 0.1, and trained the model for 30 epochs. Due to limited computing resources, we set batch size as 4, and hidden dimension size as 768. We used pre-trained bert-base-uncased tokenizer and model provided by huggingface, and fine-tuned the pre-trained BERT model during training process. For the loss function, we use cross-entropy loss.

Training Results For the training step, we report the accuracy and cross entropy loss. Figure 7 shows the accuracy and cross entropy loss during the training of BERT-pointer-generator. We stopped the training after epoch 30 because the training accuracy did not improve after epoch 30. The training accuracy have been improved on the first 10 epochs but started to fluctuate without a tendency of increasing afterwards. The best training accuracy is around 33%. Figure 8 shows the cross entropy loss during the training process of BERT-pointer-generator. The loss was decreasing on the first 10 epochs, but it stopped decreasing afterwards, and fluctuated until the final epoch.

Evaluation Results On evaluation step, we also report ROUGE-1, ROUGE-2 and ROUGE-L scores, which are the metrics we also used for

Models	ROUGE-1	ROUGE-2	ROUGE-L
BERTSUMABS (baseline)	41.72	19.39	38.76
BERTSUMABS	41.70	19.41	38.79
BERTSUMEXT (baseline)	43.23	20.24	39.63
BERTSUMEXT	43.86	20.46	39.81

Table 1: Rouge Scores for the Coreference Resolution + Fine-tuning BERT Approach and Baseline Results

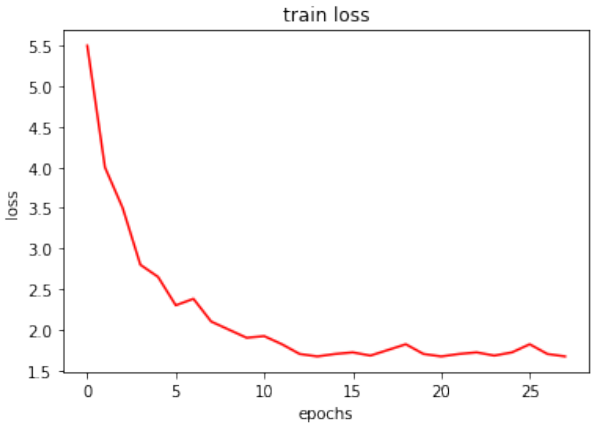

Figure 8: Cross Entropy Loss of Training BERT-Pointer-Generator

the coreference resolution and fine-tune BERT approach. Table 2 shows the ROUGE scores for evaluating BERT-Pointer-Generator model and the original pointer generator model proposed by (See et al., 2017). The ROUGE scores of BERT-Pointer-Generator model did not exceed the scores of original pointer generator model, achieving 11.7 in ROUGE-1, 7.33 in ROUGE-2, 10.64 in ROUGE-L.

5 Discussion

5.1 Coreference Resolution + Fine-tune BERT

After preprocessing the original CNN/DailyMail dataset with coreference resolution, our approach achieved better performance on extractive summarization with respect to ROUGE-1, ROUGE-2, and ROUGE-L scores, while the improvement of the abstractive summarization was very small and the ROUGE-1 score even became lower.

According to our results, we think that applying coreference resolution on the text doesn't have much help on the abstractive summarization. The slight fluctuation of ROUGE-1, ROUGE-2, and ROUGE-L scores compared with the baseline results may be caused by the randomly-initialized values of the Transformers in the architecture. The ROUGE-1 score even lowers down, which suggests that the preprocessed text didn't make the summarization better. The decrease of ROUGE-1 score means there is less overlap between the unigrams of the generated summary and the reference summary. After replacing all the pronouns with their real entities in the sentence, the generated summary is less likely to contain the pronouns or pronoun phrases, which is a possible reason to cause the difference in the unigram overlap.

For the extractive summarization, there is evidence that coreference resolution helps identifying the main topic of a document. Since the extractive summarization is basically a sentence-scoring method, it is important for the model to recognize the most important words and sentences. Replacing the pronouns and pronoun phrases with their entities makes the important terms repeat more often. The more frequent occurrence of the main entities enables the model to extract the most important information about the terms from the document.

However, compared with the state-of-art results in Zhang et al. (2019) and Zhong et al. (2020), our results are still not promising. We think the reason is that currently we just adopt a simple substitution approach where we replace all referring nominal expressions. For future work, we would like to further explore the impact of coreference resolution on the extractive summarization. Besides the BERTSUM model, we will also try the HiBERT model proposed in Zhang et al. (2019). Instead of the naive substitution approach of the coreference resolution, we will try incorporating coreference information into vector-space-based source representations. In this way, we expect the model will better capture the inter-sentence relationship and also better identify the most important information and entities in the document.

5.2 BERT + Pointer-Generator

The approach was to replace the encoder and decoder parts from the original pointer generator model suggested in (See et al., 2017), thereby improving the performance of the model. However

Models	ROUGE-1	ROUGE-2	ROUGE-L
Pointer-Generator (original)	36.44	15.66	33.42
BERT-Pointer-Generator	11.70	7.33	10.64

Table 2: Rouge Scores for Original Pointer Generator and BERT-Pointer-Generator

after lots of trials, it has been turned out that the approach was not effective. Unfortunately we were not be able to train the model successfully. Therefore, we analyzed the result and added possible reasons that might caused such a failure. Firstly, pre-trained model might not have been suitable in this case. As BERT is pre-trained in a different way from the way pointer generator is being trained, it might be a possible reason. Secondly, there exists a project that successfully combined BERT encoder and Transformer decoder with pointer-generator layer work using Chinese dataset. Therefore, there might be some problem due to difference between the languages, which we could not figure out even though we tried numerous sets of hyperparameters.

5.3 Overall Comparison

Since the accuracy and ROUGE scores for the second approach are not promising yet, we won't compare our two approaches. Also, since the first approach is about data preprocessing and the second approach is related to the model architecture, we can compare the results with their baseline models independently. Regarding the future work, we will try upgrade the accuracy of the second approach and then compare the increment or decrement of the ROUGE scores of our approaches to see whether they improve the text summarization problem with BERT.

6 Conclusion

In this project, we work on two approaches on text summarization with BERT. For the first approach, we apply coreference resolution to the original CNN/DailyMail dataset and then fine-tune the BERTSUM model for abstractive summarization and extractive summarization. We identify that applying coreference resolution improves the performance of extractive summarization while there is no significant effect on the abstractive summarization. We will explore the potential of coreference resolution on text summarization by incorporating the coreference information into the vector respresentations. For the second approach, we replace the encoder of pointer-generator with BERT and decoder with Transformer decoder. We use CNN/DailyMail dataset for training. We identify that our suggestion did not make an improvement in performance.

References

Seyed-Mehdi-Reza Beheshti, Boualem Benatallah, Srikumar Venugopal, Seung Hwan Ryu, Hamid Reza Motahari-Nezhad, and Wei Wang. 2017. A systematic review and comparative analysis of cross-document coreference resolution methods and tools. *Computing*, 99(4):313–349.

Kevin Clark and Christopher D Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. *arXiv preprint arXiv:1609.08667*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 Text Summarization Workshop*, pages 1–8.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: A graph based approach to abstractive summarization of highly redundant opinions.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28:1693–1701.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Dandan Huang, Leyang Cui, Sen Yang, Guangsheng Bao, Kun Wang, Jun Xie, and Yue Zhang. 2020. What have we achieved on text summarization? *arXiv preprint arXiv:2010.04529*.

Hongyan Jing and Kathleen R. McKeown. 2000. Cut and paste based text summarization. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Yang Liu. 2019. Fine-tune bert for extractive summarization. *arXiv preprint arXiv:1903.10318*.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Joel Larocca Neto, Alex A. Freitas, and Celso A. A. Kaestner. 2002. Automatic text summarization using a machine learning approach. In *Advances in Artificial Intelligence*, pages 205–215, Berlin, Heidelberg. Springer Berlin Heidelberg.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 104–111.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

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

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017a. Advances in neural information processing systems. *Proceedings of Machine Learning Research*, pages 5998–6008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017b. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Kam-Fai Wong, Mingli Wu, and Wenjie Li. 2008. Extractive summarization using supervised and semi-supervised learning. In *Proceedings of the 22nd international conference on computational linguistics (Coling 2008)*, pages 985–992.

Jiacheng Xu and Greg Durrett. 2019. Neural extractive text summarization with syntactic compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3292–3303, Hong Kong, China. Association for Computational Linguistics.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5059–5069, Florence, Italy. Association for Computational Linguistics.

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. Extractive summarization as text matching. *arXiv preprint arXiv:2004.08795*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.