Extractive-Abstractive of Text Summarization Stacking Model

Kuan-Lin Liu
New York University
kll482@nyu.edu

Yu-Lin Shen
New York University
yls247@nyu.edu

Yihong Wang
New York University
yw3408@nyu.edu

Yuanrui Huang
New York University
yh2910@nyu.edu

Abstract

Text summarization is a widely used technique in the information explosion world. However, one of the summarization approach, abstractive, encounter the readability issue from long text and the decreasing from repetition rate of the outcome are in the dilemma situation for improvement. In this work, we introduce a two-stage integrating concept, extractive stage, and abstractive stage, utilizing the output of extractive model as the input of the abstractive model, and implement different modern summarization techniques as multiple combinations. Finally, we measure the performance with the CNN/Daily Mail dataset by using the ROUGE score and repetition rate of the sentence.

1 Introduction

During the growth of the internet, the need for condensing information was raised, and summarization (Radev et al., 2002) technique has become more critical. In general, two types of summarization, extractive and abstractive, are distinguished. The former, extractive summarization, picks pieces of text from the original document into the summary (He et al., 2012), which tends to extract more accurate information but lose the concision. The latter, abstractive summarization, has more human-like results by generating new words and creating new sentences (Nallapati et al., 2016). Yet, an abstractive summary suffers from the problems of misinformation and inefficiency.

Recently, multiples methodologies are focusing on summarization and proposed different possibilities solutions: Among the extractive model, Narayan et al. (2018) reaches a high ROUGE score Thakkar et al. (2010) by using a graph-based structure to find connections between sentences from the original text. In the abstractive approaches, Pointer-Generator (See et al., 2017), improved from (Nallapati et al., 2016), provides the copy vocabulary, and incorporates coverage mechanism to handle the repetition.

Enlightened by integrated model (Wang et al., 2017), which inserts the output of the graph-based extractive model (Page et al., 1998), as the input of Bidirectional RNN abstractive model (Schuster and Paliwal, 1997), we propose an Extractive-Abstractive model to generate more readable abstractive summarization with BertSumTR and pointer generator methods.[1]

2 Related Work

Text summarization is generally divided into extractive summarization and abstractive summarization based on the output type.

2.1 BERT on Extractive Summarization

Since the appearance of BERT, it is widely viewed as a state-of-art tool to acquire the language's representation. BERT-based extractive summarization model has achieved great performance in Miller (2019). We thus propose BERT as a reliable tool in the Text summarization because it can acquire the representation of sentences' semantic relationships between sentences.

2.2 Abstractive Summarization

Andhale and Bewoor (2016) broadly classify abstractive summarization techniques into two groups: structured-based approach and semantic-based approach. Gupta and Gupta (2018) further propose another type as deep learning and neural network-based approach.Nallapati et al. (2016) constructs a hierarchical attention sequence-to-sequence model with two bi-directional RNN to leverage the importance on the word level and the sentence level.

[1] https://github.com/ElectronicTomato/bert_abstractive_summerization_model

2.3 Extractive + Abstractive

Last few years, some studies worked on designing the two-phase model. Wang et al. (2017) name their model extractive and abstractive long text summarization (EA-LTS), which adapts PageRank and RNN encoder-decoder. Zhang et al. (2018) introduce similar architecture with a coverage mechanism in the sequence-to-sequence model. Hsu et al. (2018) even propose an end-to-end training by defining a novice loss function, inconsistency loss, to minimize the loss function of both the extractor and abstractor at the same time.

3 Architecture

3.1 Extractive Phase

Traditional extractive model (Mihalcea and Tarau, 2004a) usually treats sentences as data points input and compares the similarity (cosine) within data points to choose the most representative sentences. However, methods only consider similarities like TextRank as in Wang et al. (2017) cannot yield summaries with high ROUGE score since this method is unsupervised. The first two sentences with the highest TextRank score will tend to have a close similarity, like two dots in a cluster closest to the Centroid. Selecting either will be representative enough, but selecting both could be redundant.

In this work, we use a BERT with enhanced modification on summarization called BERTSUM proposed by Liu and Lapata (2019). As shown in Figure 1, this method inserts external [CLS] in each sentence's beginning to acquire individual sentences and use interval segment embeddings to distinguish sentences in passages. With this mechanism, not only can high-layer classifier manipulate sentence-level representation, but also disable BERT pass precise information regarding the sentence position to the higher layer. BERTSUM has outperformed other models in CNN/DM dataset in terms of ROUGE Score.

However, to enable individual sentence manipulation, BERTSUM breaks the coherence of passage. Therefore, we proposed integrating TextRank into the BERTSUM model to ensure our model will also consider the correlation of sentences while choosing the sentence optimal for the ROUGE score. Figure 1 shows that TextRank score vectors are added to the embeddings before inputting into Transformer layers.

3.2 Abstractive Phase

In the second stage, we take the output from the first stage as the input to our abstractive models and fine-tune Pointer-Generator (See et al., 2017) on the coverage. Pointer-Generator is an extension of a standard sequence-to-sequence attention model (Nallapati et al., 2016) in which the common shortcoming of the architecture is when the decoder puts the value on the highest probability of the next token, out-of-vocabulary (OOV) rate would be large, and word repetition would be unavoidable. Those two problems definitely interrupt human understanding when reading the summary and weaken our initial motivation for generating a more readable summary. However, the two mechanisms, P_{gen} and coverage, in See et al. (2017) deal with both the repetition and OOV problem. We will talk more about them in the next section.

4 Methodology

4.1 BERTSUM

BERTSUM is a BERT optimized for summarization task. BERTSUM inserts externel [CLS] in the beginning of each sentence to acquire individual sentence. Also, interval segment embeddings are used in BERTSUM.Intervel segement assign segment embeddings of $sent_i$ with E_A or E_B depending on parity of i. For instence, a document $[sent_1, sent_2, sent_3, sent_4, sent_5]$ would be assigned with $[E_A, E_B, E_A, E_B, E_A]$.

4.2 TextRank

We use TextRank by Mihalcea and Tarau (2004b) to measure the importance of sentence for the document level. Denote $G = (V, E)$ as an undirected graph with the set of vertices V and set of edges E. For a given vertex V_i, $In(v_i)$ is the set of vertices where vertex V_i points to(predecessors), and $Out(V_i)$ is the set of vertices where vertex V_i points to (successors). Besides, w_{ij} indicates the weight of the edge between the node V_i and V_j. Weight $w_{i,j}$ is calculated using the cosine similarity between the two sentences S_i and S_j. Specifically, we use the token embedding of the BERT for each sentence to calculate the cosine similarity. The score of the vertex V_i is defined as follows:

$$S(V_i) = (1-d) + d \times \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} S(V_j) \quad (1)$$

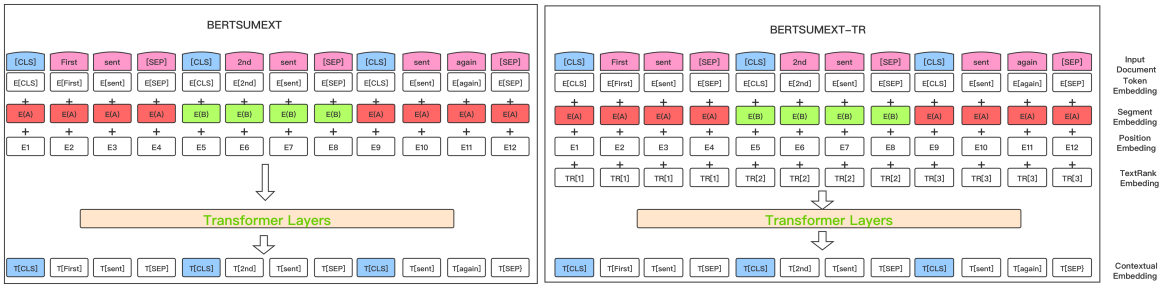

Figure 1: Structure of BERTSUMEXT(left) and BERTSUMEXT-TR(right). EXT stands for Extractive. The sequence on top is input document. BERTSUMEXT applies multiple [CLS] insertion and interval segmentation embeddings, while BERTSUMEXT-TR adds TextRank Embedding on the basis of BERTSUMEXT

where d is a damping factor that can be set between 0 and 1. The suggested factor d value is 0.85 (Brin and Page, 1998), and this is the value we are also using in our implementation.

4.3 Pointer-Generator

After capturing the extractive summary from the first stage, we employ Pointer-Generator (See et al., 2017) [2]as the abstractive layer. The main concept of the Pointer-Generator is extracting the part of the original text as the input to complement the unknown phase. To solve the OOV problem, P_{gen} is defined as the probability to extract the word from the vocabulary or the input:

$$P_{gen} = \sigma(w_{h*}^T h *_t + w_s^T s_t + w_x^T x_t + b_{ptr}) \tag{2}$$

σ is the sigmoid function, $h*_t$ is the context vector, which is the attention after the encoder and decoder, s_t is the decoder state, x_t is the input of the decoder and $w_{h*}^T, w_s^T, w_x^T, b_{ptr}$ representing as vectors are learnable parameters. We will then obtain the following probability distribution $P(w)$:

$$P(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i=w} a_i^t \tag{3}$$

Moreover, Pointer-Generator utilized the coverage mechanism to decrease the repetition in the experiment by adding the coverage loss as the form:

$$loss_t = -logP(w_t^*) + \lambda \sum_i min(a_i^t, c_i^t) \tag{4}$$

5 Experiment

All extractive models are implemented with BERT-based Encoders and linear classifiers to predict the sentence score(the last layer is a sigmoid layer). For the baseline model, we use original BERT, and for BERTSUM, we use the same model setting in Liu and Lapata (2019). Our BERTSUM-TR model setting is almost the same with BERTSUM but with a TextRank Embedding added in the encoder layer.

5.1 Summarization Datasets

We implement our work on the CNN/Daily Mail news highlight (Hermann et al., 2015), which has been widely used in the previous text-summarization work.

Table 1: Extractive model ROUGE Score

Length	ROUGE-1	ROUGE-2	ROUGE-L
Baseline	40.90	18.02	37.17
BERTSUM	42.03	18.92	38.82
BERTSUM-TR	**42.37**	**19.39**	**38.90**

The F-measure of different ROUGE-N scores

5.2 Result

5.2.1 Extractive model results

We evaluate our summarization quality with ROUGE score proposed by Lin and Och (2004). Table 1 shows the different ROUGE scores with different models, and our BERTSUM-TR model reaches the highest ROUGE score.

We also investigate our model by analyzing the selected sentences' distribution. We set the output selected sentence number as three. From Figure 2, we can conclude that oracle selection is smoothly distributed through the passage, while the baseline model's selection is more central to the first three sentences. BERTSUM's output is the most decentralized one as the mechanism of BERTSUM limits sentence position information passing to higher layers. BERTSUM-TR's output is most similar to Oracle's (Notice only Oracle and BERTSUM-TR

[2] The repository of Pointer-Generator refers to: https://github.com/abisee/pointer-generator

have picked more sentences in the fourth position than the third position).

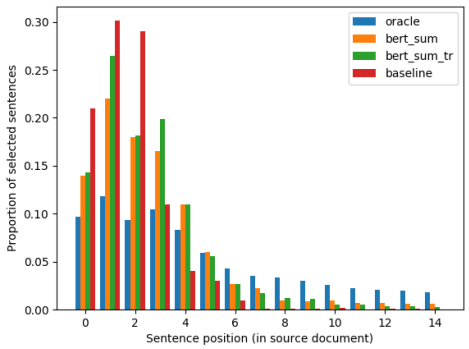

Figure 2: Proportion of extracted sentences corresponding position in original passage

5.2.2 BERTSUM-TR + Pointer-Generator

First of all, we preprocess the output obtained from the first phrase and transform the data into the required binary format for Pointer-Generator. Second, we use the pre-trained model provided by (See et al., 2017) and fine-tune on the decoder by changing the weights of coverage, which could potentially reduce the repetition. Three different coverage weights, 0.2, 0.5, and 1, which are suggested in (See et al., 2017) are selected in this experiment. We also try no coverage to see whether the coverage mechanism does help. After getting the summary from Pointer-Generator, we compute the repetition rate and the ROUGE score introduced by Lin and Och (2004)[3]. The ROUGE score result with the coverage weight of 1 is show in Table 2. The repetition rate P_{rep} is computed as the following:

$$P_{rep} = 1 - \left(\frac{f_{unique}(x)}{f_{total}(x)}\right) \quad (5)$$

where $f_{unique}(x)$ is the number of the unique 1-gram tokens and $f_{total}(x)$ is the number of the total 1-gram tokens.

From Figure 3, we find that the ROUGE-1 scores do not change too much with the different choices on numbers of sentences obtained from BERTSUM-TR. Also, in Figure 4, the repetition rate only decreases with the 10-sentence BERTSUM-TR outputs.

6 Conclusion

By adding TextRank embedding to BERTSUM-TR, we enable our extractive model to consider the relevance of the sentences. When the model encounter sentences with a similar ROUGE score, BERTSUM-TR will tend to choose the sentence with higher TextRank, i.e., the sentence more relative to the whole passage. Therefore, our extractive model can produce outputs that reach a higher ROUGE score in the test dataset.

As for the abstractive model, Pointer-Generator has significantly reduced the repetition rate with the 10-sentence input and generated human-readable summarization. However, future work may include exploring the difference between the input sentence's length. We would like to study further the reason why only 3-sentence input could lead to around 33% of the ROUGE score.

[3] https://github.com/google-research/google-research/tree/master/rouge

Table 2: ROUGE-N score and repetition rate of BertSumTR + Pointer-Generator

Length	ROUGE-1	ROUGE-2	Avg Rep
N	32.11	12.20	**22.80**
3	33.16	**13.49**	24.59
6	33.11	13.33	24.49
10	**33.18**	13.08	22.97

The F-measure of different ROUGE-N scores in percentage are shown in the table. Length denotes the length of the sentence obtained from BERTSUM-TR in which N represents the original test data set. Avg Rep is the average repetition rate among all input to Pointer-Generator.

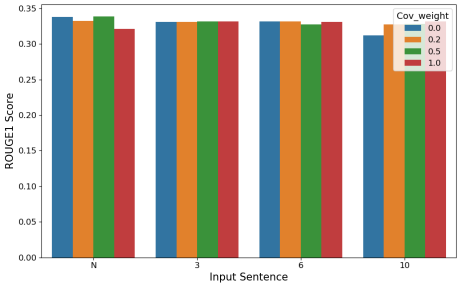

Figure 3: ROUGE scores of all coverage weights

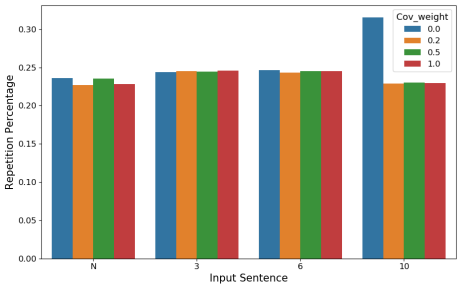

Figure 4: Repetition rate of all coverage weights

7 Contribution Statement

Each member corporate fully and equally.

8 Collaboration Statement

All members agree to collaborate with the spirit of cooperation, equality and sharing. Any conflict will be resolved by a remote meeting. We reserve the right to redistribute points unevenly between team members in case of very uneven contributions.

References

N. Andhale and L. A. Bewoor. 2016. An overview of text summarization techniques. In *2016 International Conference on Computing Communication Control and automation (ICCUBEA)*, pages 1–7.

S. Brin and L. Page. 1998. The anatomy of a large-scale hypertextual web search engine. In *Seventh International World-Wide Web Conference (WWW 1998)*.

Som Gupta and S.K Gupta. 2018. Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications*, 121.

Zhanying He, Chun Chen, Jiajun Bu, Can Wang, Lijun Zhang, Deng Cai, and Xiaofei He. 2012. Document summarization based on data reconstruction. In *AAAI*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 132–141, Melbourne, Australia. Association for Computational Linguistics.

Chin-Yew Lin and FJ Och. 2004. Looking for a few good metrics: Rouge and its evaluation. In *Ntcir Workshop*.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *EMNLP/IJCNLP*.

Rada Mihalcea and Paul Tarau. 2004a. Textrank: Bringing order into text. In *EMNLP*.

Rada Mihalcea and Paul Tarau. 2004b. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

Derek Miller. 2019. Leveraging BERT for extractive text summarization on lectures. *CoRR*, abs/1906.04165.

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.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *ArXiv*, abs/1808.08745.

L. Page, S. Brin, R. Motwani, and T. Winograd. 1998. The pagerank citation ranking: Bringing order to the web. In *Proceedings of the 7th International World Wide Web Conference*, pages 161–172, Brisbane, Australia.

Dragomir R. Radev, Eduard H. Hovy, and Kathleen McKeown. 2002. Introduction to the special issue on summarization. *Computational Linguistics*, 28:399–408.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.*, 45(11):2673–2681.

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

K. S. Thakkar, R. V. Dharaskar, and M. B. Chandak. 2010. Graph-based algorithms for text summarization. In *2010 3rd International Conference on Emerging Trends in Engineering and Technology*, pages 516–519.

S. Wang, X. Zhao, B. Li, B. Ge, and D. Tang. 2017. Integrating extractive and abstractive models for long text summarization. In *2017 IEEE International Congress on Big Data (BigData Congress)*, pages 305–312.

Yong Zhang, Erdan Chen, and Weidong Xiao. 2018. Extractive-abstractive summarization with pointer and coverage mechanism. pages 69–74.