Get the Key Points: Summarization by Grouping Sentences

Bohan Zhang
University of Michigan
zbohan@umich.edu

Yujian Liu
University of Michigan
yujianl@umich.edu

1 Problem Description

Text summarization task aims to automatically generate a summary for a given document. The input is a long document and corresponding summary usually consists of few sentences. There are mainly two methods of summarization: extraction and abstraction. The extractive method directly selects salient sentences from the document as the summary whereas the abstractive method rewrites and compresses the document into a shorter one as the summary. These two methods can also be combined in an extract-then-abstract way where salient sentences in the source article are extracted first and then rewritten to the summary.

2 Related Work

Extractive and abstractive methods have both been popular in text summarization task. They handle the task in two different ways. Extractive methods form the summary by extracting salient content (Kupiec et al., 1995), and abstractive methods generate the summary by rewriting like humans (See et al., 2017). Recently, some works try to combine these two methods in an extract-then-abstract way to get advantages of both (Sharma et al., 2019; Chen and Bansal, 2018; Dong et al., 2018; Xiao et al., 2020).

In this project, we also follow the extract-then-abstract framework. However, most previous work extract one sentence at each decoding time step and rewrite them independently. Consequently, they tend to ignore that sources sentences can be fused together to generate a summary, which results in a less coherent summary. We attempt to tackle this problem from two ends: In extraction part, we enable the extractor to extract multiple sentences at one decoding step. Then in abstraction part, we group the multiple extracted sentences at one decoding time together and rewrite and compress them to a summary sentence. A recent work(Lebanoff et al., 2019) proposes a sentence fusion based summarization. However, they have to inefficiently predict over all sentence pairs in the source article and rank them in a order that is not necessary to be the coherent order of the source article. Also, they are not able to fuse more than two sentences. Our model won't require examination over all source sentence pairs but just enable decoding multiple sentences at one time sequentially.

3 Methodology

We will introduce the architecture of our model in this section. We also propose a new method to extract multiple matched original sentences of each summary sentence to train the extractor.

3.1 Data Pre-processing

We use the CNN/Daily Mail news dataset(Hermann et al., 2015). It includes 287226 training, 13368 validation and 11490 test pairs. We start by manually labeling 50 documents to verify the many-to-one relation between original document sentences and summary sentences. For each summary sentence, we label the associated sentences in original document. We found 26.7% summary sentences are fused from multiple original sentences and 70% summaries contain at least one fused sentence. This demonstrates that sentence fusion is a common phenomenon in human written summaries, and it is important for the model to be able to fuse multiple original sentences into one summary sentence.

We then build a key-word-driven label construction method. For each summary, we extract its keywords using tf-idf (Ramos et al., 2003) based approach. We randomly sample 10,000 summaries as a training corpus and train a tf-idf transformation model in scikit-learn's tf-idf implementation(Pedregosa et al., 2011). Then we use the trained tf-idf model to produce a tf-idf for the

	Avg # of selected sentences	Recall
Greedy	3.22(±0.76)	61.04%
Gold	4.34(±1.42)	–
Ours	5.14(±1.65)	77.92%

Table 1: Statistics of constructed dataset.

summary. The words present in the summary are ranked in decreasing order of tf-idf score, and top 12 words are selected as keywords. Then for each sentence in a summary, we iteratively pick the most similar original sentence (based on ROUGE-L score) that covers some keywords of the summary sentence until all keywords are covered or 3 original sentences have been selected. This is different from most label construction methods used in previous extract-then-abstract work. Usually, they just select one most similar sentence from the original document. We may select multiple sentences in the original document to cover all keywords in the summary sentence.

We apply this algorithm on CNN/Daily Mail dataset to get extraction labels. Some statistics of the labeled dataset are shown in table 1. As a result, we see that our algorithm achieves a higher recall on labeled 50 documents compared to the greedy algorithm that only extracts the most similar original sentence, while extracting 0.8 more sentences per summary than gold labels.

3.2 Extractor

The extractor extracts salient sentences and groups related ones together. We use hierarchical neural models to learn sentence encoding, and exploit a decoder network to select sentences.

3.2.1 Hierarchical Sentence Encoding

We first use BERT (Devlin et al., 2019) to encode each original sentence following (Xiao et al., 2020) because that encoding provides sentence-level context. We take encoding of each token from BERT and use a soft attention mechanism to combine them into sentence encoding. We then pass each sentence encoding to a Bi-LSTM to capture global document information. The ourput is our final sentence encoding h.

3.2.2 Sentence Selection

To extract a sequence of original sentences, we train another LSTM network as Pointer Network (Vinyals et al., 2015b). Specifically, at each decoding step, we calculate a context vector c_t following glimpse operation (Vinyals et al., 2015a):

$$a_j^t = v_g^T tanh(W_{g1}h_j + W_{g2}z_t) \tag{1a}$$

$$\alpha^t = softmax(a^t) \tag{1b}$$

$$c_t = \sum_j \alpha_j^t W_{g1}h_j \tag{1c}$$

where z_t is the hidden state of LSTM, W_{g1}, W_{g2}, v_g are learnable weights. We then calculate the extraction score of each original sentence by:

$$u_j^t = v_p^T tanh(W_{p1}h_j + W_{p2}c_t) \tag{2}$$

We extract original sentences according to their extraction score, and we take encoding of previously selected sentences as the input to next step LSTM.

3.2.3 Training Objective

We train the extractor using mean squared error between predicted extraction score and ground truth extraction score. We obtain ground truth extraction score using previously constructed dataset. Specifically, original sentences have extraction scores 2; 2, 1.5; 2, 1.5, 1 if a summary sentence maps to one, two, or three original sentences respectively. For all unselected sentences, they have score 0. We distinguish scores for multiple selected sentences because we want the most similar sentence to be extracted more likely. At inference time, we first extract the sentence with highest extraction score, and we further select sentences if they have score higher than threshold 0.3.

We also tried other training objectives such as binary cross entropy loss and KL divergence, but we found mean squared error converges faster.

3.3 Abstractor

For the abstractor, we use the traditional encoder-aligned-decoder model(Bahdanau et al., 2014) with attention mechanism(See et al., 2017). Due to the limitation of computational resources, we tuned the abstractor from the pre-trained weights provided by (Chen and Bansal, 2018). For the input, if a sentence is just extracted singly by the extractor then the input to the encoder of the abstractor is just the sentence. If multiple sentences are grouped together by the extractor then the input to the encoder is the concatenation of them. The training loss of the abstractor is the cross-entropy loss of the decoder language model at each generation step. Since one source sentence may be extracted multiple times by the extractor which is then feed into

	ROUGE-1	ROUGE-2	ROUGE-L
Lead-5+abstractor	45.46	19.17	41.98
Extractor+abstractor	36.14	13.74	32.02
System-greedy-selection+abstractor	46.77	25.08	45.26
System-multiple-selection+abstractor	49.82	26.30	46.84

Table 2: Final results

F1-score	0.28
% of fused summary sentences	62%

Table 3: Performance of extractor.

the abstractor to rewrite, we add a post-processing procedure to avoid duplicate sentence output of the abstractor. We will remove an output sentence if it has the recall of ROUGE-L higher than 0.7 with any its preceding output sentence.

4 Experiments and Results

4.1 Dataset and Evaluation Metrics

We only use 1/3 of the training data of CNN/Daliy Mail dataset due to the limitation of the computational resources. We evaluate standard ROUGE-1,2 and L on full length F1 on full test data.

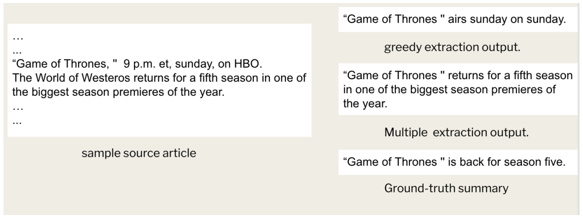
4.2 Extractor Performance

We evaluate extractor on constructed dataset. The results are shown in table 3. We notice that the performance is not good. Particularly, we suffer from the overfitting problem during training, where validation accuracy starts dropping after a few epochs while training loss keeps decreasing. This overfitting exists regardless of how much dropout or regularization we add to the model. We think the overfitting might come from the data. Due to limitation of computation resources, we only train the extractor on 3,000 data, which is about 1% of the total training data. Besides, our constructed dataset is not accurate enough and still contains noise.

4.3 Final Results

The final results are shown in table 2. The baseline, shown in the first row, is feeding the lead-5 sentences of the source article into the abstractor. It achieved 41.98 R-L scores. As we don't have a strong extractor at this moment, our full model achieved 32.02 R-L scores.

We have two additional experiments: one is system-greedy-selection. We feed the constructed labels by most previous work where only the


…
…
“Game of Thrones, " 9 p.m. et, sunday, on HBO.
The World of Westeros returns for a fifth season in one of the biggest season premieres of the year.
…
…

“Game of Thrones " airs sunday on sunday.

“Game of Thrones " returns for a fifth season in one of the biggest season premieres of the year.

“Game of Thrones " is back for season five.

Figure 1: Sample Output

sentence with the highest ROUGE-L recall in the source article with the corresponding reference sentence is selected. The other is system-multiple-selection. We feed multiple selected sentences which are the outputs of our label construction system to the abstractor. The results show that multiple-selection outperforms the greedy-selection by 1.58 rouge scores. Even We don't have the expected extractor performance at this moment, we think this shows the potential of our hypothesis: Salient sentences should be grouped together to boost the performance in the summarization task.

A sample output is shown in figure 1. In this example, we can see the ground truth summary fuses two source article sentences. For greedy-selection, it only rewrites the first source sentence and the output is ungrammatical. For multiple-selection, the abstractor can learn to fuse the two sentences together. It covers the key entities like the subject, verb and number.

5 Conclusion

In conclusion, we designed an extract-then-abstract model where the extractor can extract and group multiple sentences one time and the abstractor generate summaries by fusing the extracted sentences. We don't get a strong extractor but our results show the potential of improving performance by extracting and fusing multiple sentences. In future work, if possible, we may expect to train the extractor with more data and consider high-level abstraction to remove the post-processing procedure.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. Cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. Banditsum: Extractive summarization as a contextual bandit. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3739–3748.

K. Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, W. Kay, Mustafa Suleyman, and P. Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '95, page 68–73, New York, NY, USA. Association for Computing Machinery.

Logan Lebanoff, Kaiqiang Song, Franck Dernoncourt, Doo Soon Kim, Seokhwan Kim, Walter Chang, and Fei Liu. 2019. Scoring sentence singletons and pairs for abstractive summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2175–2189, Florence, Italy. Association for Computational Linguistics.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. volume 242.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Eva Sharma, Luyang Huang, Zhe Hu, and Lu Wang. 2019. An entity-driven framework for abstractive summarization. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2015a. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015b. Pointer networks. In *Advances in Neural Information Processing Systems*, volume 28, pages 2692–2700. Curran Associates, Inc.

Liqiang Xiao, Lu Wang, Hao He, and Yaohui Jin. 2020. Copy or rewrite: Hybrid summarization with hierarchical reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.