Informative Sections and Relevant Words for the Generation of NLP Article Abstracts

Tianjiao Li and Yves Lepage
IPS, Waseda University
ltj1994@fuji.waseda.jp, yves.lepage@waseda.jp

Abstract

In this paper, we show that, in the field of NLP, abstracts of articles make only use of the text found inside the article. We introduce a multi-channel summarization method which makes use of only some sections to generate a summary. We show that the introduction and the conclusion are the most useful sections to generate accurate abstracts. We also show that reducing the content to informative N-grams suffices to retain the necessary information to generate accurate abstracts.

1 Introduction

There are two approaches to automatic summarization: extractive and abstractive [2]. Extractive summarization only copies words and sentences from the source text. It cannot create new words. Summarization techniques on scientific articles are mostly extractive approaches. The most advanced techniques use extra information such as citation summaries or citation networks. In contrast, abstractive summarization reads the whole source text and generates a summary which may contain words not found in the source text [6]. If machine learning techniques are used, these new words are picked from the vocabulary learned during the training phase.

Original extractive methods usually calculate a score to reflect how important a word or a sentence is. They use words and sentences which have highest score in their output [8]. However, sentences selected in this way may be not relevant in content. They may also not be in the right order in structure. Work has been done to try to solve this problem, including work on rhetorical status and structure [7, 13], and citation networks [9, 10]. Such previous work concluded that, in order to improve the performance of extractive summarization, extra information such as rhetorical status or citation networks is needed.

On the contrary to such previous studies, we show that, to generate a summary for a scientific article, all we need is the content of the article itself and that no extra information is needed. Therefore, we choose abstractive method to generate summaries. Abstractive methods are usually based on recurrent neural networks (RNN) [1, 12]. RNNs are a kind of neural networks equipped with a memory about the history of the input. They can generate words based on this history. Such NNs are suitable for the generation of abstractive summaries. OpenNMT-py [4] is an open-source tool originally designed for neural machine translation. It has been extended to text summarization and other tasks. It uses RNNs as its basic framework. However, being a mock-up tool, it has a heavy practical restriction on the length of the input. We confirmed this during our experimental manipulations. This practical constraint is a challenge which goes into the direction of our claim that only the content of a scientific article is sufficient for the generation of its abstract. It goes even further in constraining us to propose some light pre-processing to make the article content even shorter, without having an impact on the result of summarization. In other words, the pre-processing to shorten the content of the article should keep the informative content that may appear in the summary.

To train the neural network, a reasonable amount of data is necessary. But there is no such data set available on scientific articles. So we built one by ourselves.

In this paper, we introduce a multi-channel abstractive approach to generate summaries of scientific articles only using their sections. We show that to generate a summary, all we need is the article itself. Moreover, we hope we can generate abstract-like summaries, so that once an article is written, its abstract can be generated automatically.

2 Background

2.1 Recurrent Neural Network

Recurrent neural networks (RNN) use hidden units to store the information from the current input and previous hidden units, so that they have a memory of inputs of all the time. It is suitable to be used on summary generation. One of their drawbacks is that their performance drop on longer inputs [3].

In the encoder-decoder model, the encoder part reads the source sentence token by token and uses hidden states to store the information. The final output of the encoder part is passed to the decoder part which generates the target sentence token by token. All tokens previously generated influence the generation of the next token. As a result of an output token being conditioned on the source

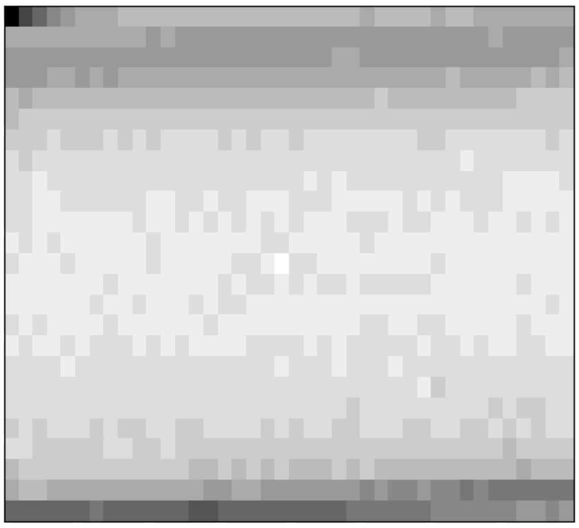

Figure 1: Number of times a token in an article appears in the abstract

and the previously generated target part, sentences generated in this way are more grammatical and more fluent.

2.2 ACL-ARC data set

Collections of articles built for extractive summarization are too small in size for training a neural network. Therefore, we built a data set using the ACL-ARC corpus[1]. ACL-ARC is a corpus of scholarly publications about computational linguistics. It consists of 22,878 articles. We extracted the text from each article and broke it down into four parts: abstract, introduction, conclusion and body (i.e., the rest). So far, we have already extracted texts from 15,000 articles.

2.3 ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [5] is a metric to evaluate the performance of summarization. The scores range from 0 to 100. The higher the score, the better the performance. Rouge computes a recall, a precision and an F-score against a gold standard. The unit to compute the scores may be unigrams (ROUGE-1) or bigrams (ROUGE-2). The ROUGE-L score uses the length of the longest common subsequence as the numerator to calculate the recall and precision scores.

3 Which sections are informative for an abstract?

We use the open-source tool OpenNMT-py for our experiments. The basic mechanism of OpenNMT-py is attention based recurrent neural network. As mentioned above, the performance of RNNs drops when the input

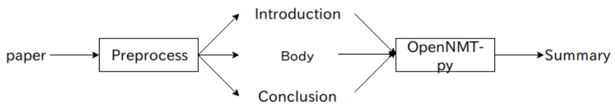

Figure 2: Multi-channel method

length increases. In practice, OpenNMT-py does not allow to use the whole body of an article as an input. It limits the input to 400 tokens. To remedy this constraint, we propose to use only some sections of an article. It is like a filter through which only some piece of information is passed. We call each piece of information a channel. In this way a whole article will be divided into several channels.

The next question is which parts of an article, i.e., which channel, is most effective for our goal of summarizing an article and getting an abstract-like summary. Usually, an abstract is structured in the following way: it introduces the background of a topic, states the existing problems, proposes a new approach, mentions the main results of experiments and stresses the improvements made.

A natural hypothesis is that all these points are described in details mainly in the introduction and the conclusion sections. To confirm this hypothesis, we perform the following experiment on our ACL-ARC data set: for a given article, we replace each token in the paper, from the first token of the introduction to the last token of the conclusion, with the number of times it appears in the abstract. We average all articles by scaling them to the same length and plot the picture obtained in Figure 1. In this figure, the pixels are the averaged positions of tokens. The darker they are, the higher number of times they appear in the abstract. It is obvious that the top and bottom areas of Figure 1, which correspond to the introduction and conclusion sections, are much darker.

However, the body of the article is longer than the introduction and the conclusion. It may be the case that the words in the abstract also come from the body of the article. We calculate the ROUGE scores of different parts of articles using their abstracts as gold standards. These scores are shown in Table 1a. Introduction, body and conclusion all exhibit a high ROUGE-1 recall score, with the body part getting a higher score. It means that most of the words in an abstract are actually found in the body of the article. In other words, the abstract is already almost entirely in the body of an article.

4 Which words are relevant for an abstract?

Compared to recall scores, ROUGE-1 precision scores are much lower. This means that all parts contain more words which are not in the abstract, than which are. These words are irrelevant to generate the abstract. Neural networks will learn which words are more important and

[1]http://acl-arc.comp.nus.edu.sg/

Table 1: ROUGE scores of different sections of scientific papers and summaries generated from them. (a) and (b) are for the input sections, raw or after pre-processing. (c) is for generated summaries. I, C and B stand for introduction, conclusion and body, respectively. Boldfaced numbers show best ROUGE scores for each unit (unigram (ROUGE-1), bigram (ROUGE-2) or longest common subsequence (ROUGE-L)) and each measure (recall (R), precision (P) and F-score (F)).

(a) Raw sections.

	# of tokens	ROUGE-1			ROUGE-2			ROUGE-L		
	avg. ± std.dev.	R	P	F	R	P	F	R	P	F
I	471 ± 233	74.38	19.77	29.67	31.52	8.04	12.22	56.59	15.03	22.54
B	2656 ± 1363	**82.43**	6.22	10.43	**34.36**	2.17	3.81	**67.18**	4.96	8.32
C	232 ± 164	56.96	**34.04**	**38.71**	19.46	**12.14**	**13.59**	43.55	**26.52**	**29.90**

(b) Sections after pre-processing. Stop words and punctuation marks have been removed from introduction and conclusion. (I′, C′) Body is reduced to the more informative N-grams only. (B″)

	# of tokens	ROUGE-1			ROUGE-2			ROUGE-L		
	avg. ± std.dev.	R	P	F	R	P	F	R	P	F
I′	295 ± 146	**65.26**	17.12	25.83	**26.24**	6.79	10.29	**40.74**	10.66	16.07
B″	31 ± 24	10.20	**38.88**	14.95	1.43	5.40	2.08	7.86	**31.37**	11.62
C′	139 ± 99	46.52	28.97	**32.45**	16.07	**10.71**	**11.72**	28.12	18.15	**20.03**

(c) Summaries generated from different combinations of sections after pre-processing.

	ROUGE-1			ROUGE-2			ROUGE-L		
	R	P	F	R	P	F	R	P	F
Summary of I′	40.38	44.77	40.50	10.82	12.17	10.84	31.22	35.10	31.46
Summary of B″	28.59	31.03	28.56	5.43	5.87	5.42	22.72	24.87	22.79
Summary of C′	39.22	40.10	38.22	9.80	10.03	9.55	31.00	32.01	30.34
Summary of I′+B″	41.56	43.98	41.10	11.43	12.06	11.27	32.25	34.53	32.05
Summary of B″+C′	38.76	41.37	38.43	9.92	10.53	9.78	30.55	32.92	30.39
Summary of I′+C′	43.44	**45.83**	**42.83**	**12.73**	**13.37**	**12.51**	33.54	**35.83**	**33.26**
Summary of I′+B″+C′	**43.51**	45.33	42.71	12.72	13.29	12.48	**33.58**	35.48	33.16

which are irrelevant given enough training data. However, experimentally, RNNs perform poorly on longer inputs and practically OpenNMT-py imposes a limit of 400 tokens in length. Table 1a shows that the average length of each part is well over this limit. We propose a pre-processing to reduce length.

For the introduction and conclusion channels, we remove all stop words and punctuation marks. These tokens are not significant as they can be reproduced by the RNN model because they are remembered from the abstracts themselves. Table 1b shows that, after removing stop words and punctuation marks, the ROUGE scores of the introduction and the conclusion channels decrease. This means that information is now missing.

The body channel scores in Table 1a are taken for all the text which extends from end of the introduction to the beginning of the conclusion. The ROUGE-1 recall score is higher in average than for the introduction of the conclusion. This means that these sections contain more words which appear in the abstract. To solve the length problem (average of 2,656 tokens), simply removing stop words and punctuation marks is not enough. Instead, we perform keyword extraction using the Rake tool[2] [11]. The length of the entire body is reduced to a very small number of tokens (average 31). ROUGE recall scores decrease but precision scores increase, which was expected.

After the pre-processing presented above, an article is reduced to three channels: introduction, body and conclusion. Their lengths are acceptable for OpenNMT-py, even when concatenated. If too long, OpenNMT-py uses only the first 400 tokens.

We randomly divide the ACL-ARC articles into a training set (14,000 articles), a validation set (500) and a test set (500). Different combinations of the three channels are tested. The results are shown in Table 1c. The length of the summaries generated by different combinations of channels range from 110 tokens to 120 tokens, which matches well the average length of the golden standard average length (113 tokens).

[2]`https://github.com/zelandiya/RAKE-tutorial`

5 Analysis

Among summaries generated using only one channel, the introduction channel has the highest ROUGE score. It is easy to understand from Table 1b that this channel has the highest recall score. This means that it has the largest number of important words, and that it is the most informative one. For summaries generated using a combination of two channels, introduction and conclusion (I'+C') is the most informative one. Table 1b also shows that the length of I'+B''+$C'$is slightly longer than 400 in average. OpenNMT-py will drop the last tokens after 400. Compared to I'+C', in I'+B''+C', some tokens in conclusion will be replaced by tokens from the body, but body is not as informative as conclusion. This explains why summaries generated using I'+B''+C'end up with a lower ROUGE than summaries generated only from I'+C'.

6 Conclusion

In this paper, we showed that, at least on the collection of papers from our data set, ACL-ARC, most of the words in the abstract appear in the article. The conclusion is that, to generate the abstract of a scientific paper automatically, the task consists of removing irrelevant words.

In order to solve the input length problem for a tool like OpenNMT-py, we proposed a multi-channel method, where the channels are the introduction, the body and the conclusion of an article, on which a light pre-processing has been performed.

We performed experiments on a set of articles from the natural language processing domain: ACL-ARC. We showed that generating an abstract from the introduction and the conclusion only is better than relying on the entire article. The introduction and the conclusion are the most informative parts of the article.

7 Acknowledgement

This work was supported by JSPS KAKENHI Grant Number JP18K11446.

References

[1] Sumit Chopra, Michael Auli, and Alexander M Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, 2016.

[2] Dipanjan Das and André F. T. Martins. A survey on automatic text summarization. *Literature Survey for the Language and Statistics II course at CMU*, 4:192–195, 2007.

[3] Trieu H. Trinh, Andrew M. Dai, Lng Thng, and Quoc V. Le. Learning longer-term dependencies in RNNs with auxiliary losses. In *International Conference in Machine Learning*, February 2018.

[4] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017. doi: 10.18653/v1/P17-4012.

[5] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In Stan Szpakowicz Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[6] Elena Lloret and Manuel Palomar. Text summarization in progress: a literature overview. *Artificial intelligence review*, 37(1):1–41, January 2012.

[7] Kenji Ono, Kazuo Sumita, and Seiji Miike. Abstract generation based on rhetorical structure extraction. In *Proceedings of the 15th conference on Computational linguistics-Volume 1*, pages 344–348. Association for Computational Linguistics, 1994.

[8] H P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):157–165, 05 1958.

[9] Vahed Qazvinian and Dragomir R Radev. Scientific paper summarization using citation summary networks. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 689–696. Association for Computational Linguistics, 2008.

[10] Vahed Qazvinian, Dragomir R. Radev, Saif M. Mohammad, Bonnie Dorr, David Zajic, Michael Whidby, and Taesun Moon. Generating extractive summaries of scientific paradigms. *Journal of Artificial Intelligence Research*, 46(1):165–201, 2013.

[11] S. Rose, D. Engel, N. Cramer, and W. Cowley. *Text Mining: Theory and Applications*, chapter Automatic Keyword Extraction from Individual Documents, pages 1–20. John Wiley & Sons., 2010.

[12] Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024, 2011.

[13] Simone Teufel and Marc Moens. Summarizing scientific articles - experiments with relevance and rhetorical status. *Computational Liguistics*, 28(4): 409–446, 2002.