SSEA: News Summarization System with Extractive and Abstractive methods

Chi-luen Feng, Yu-chieh Huang
National Taiwan University

Abstract

The technology of machine learning has widely existed in this society, and there are major breakthroughs in the Computer Vision or Speech field. In the field of Text Summarization, there are two different types of practices, namely Extractive and Abstractive.

In this article, we experiment with 4 different Summarization models, including 2 Extractive and 2 Abstractive methods, and compare the performance(ROGUE[1]) brought by different Summarization methods , and analyze their speed. In addition to the research in this article, we also attach the code for your use*.

Moreover, we combined these models with a web crawler to capture CNN news on the Internet in real time, providing users with a more convenient and fast way to absorb the ever-changing knowledge, the product architecture is shown below.

1. Introduction

In modern society, people's time is gradually becoming fragmented, but the amount of information is increasing exponentially every day. Therefore, how to grasp the fragment time to absorb knowledge is an important topic.

We hope to use automated web crawling technology to crawl the latest news on the world in real time, so that users can save a lot of time on tedious information retrieval behavior. In addition, we use Text Summarization to condense these various articles into a few sentences, so that users can get the latest information in a very short time.

We used 2 Extractive methods and 2 Abstractive methods. In the Extractive part, we use **BERT Embedding + Kmeans**[2] technology and **TextRank**[3] technology, both of which transfer the different Sentences of the article to the high-dimensional Embedding Space, and then give each Sentence of the article a weight by grouping or sorting. At the end, scores are given by outputting sentences with higher weights. In the Abstractive part, we tried the traditional **Graph Based** method and the **Learning Based** method to generate abstractive sentences which might not appear in the original article. With Graph structure, We can split each sentence into words, and then recombine these words into suitable summarized sentences. On the other hand, using machine learning, we can train a sequence to sequence model to generate abstractive sentences.

---

*https://github.com/chiluen/DSP_2020_Final

Through the above methods, we hope that users can determine the desired Summarization model by themselves, and get a Summarization that is more suitable for them. In this way, unstructured knowledge on the Internet is transformed into structured knowledge.

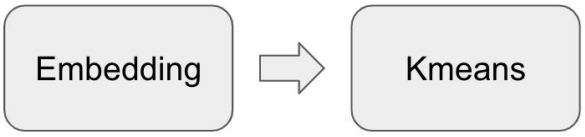
2. Method

This part will introduce the Summarization method we use, including Extractive summarization and Abstractive summarization.

2.1.1 Extractive summarization-BERT_Kmeans

This method is to first separate each sentence from article, then use the Pretrained model (such as BERT[4], ALBERT[5], etc.) to convert a whole sentence into Sentence Embedding, and then use the Kmeans algorithm to group the Sentence Embedding in a high-dimensional space. At the end, according to the number of Centroids, find each point corresponding to the shortest distance of Centroids, and combine these representative points into a summarization.

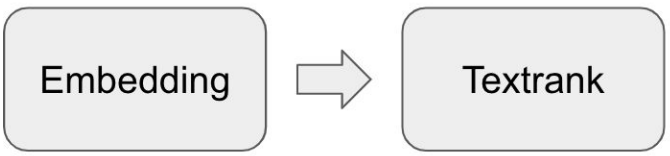
The characteristic of this model is that through models such as BERT, contextualized representations of sentences can be generated. Even if it is the same sentence, different representations can be generated through the semantic meaning of the context in different articles. In Summarization, this effect allows the Kmeans algorithm to perform classification more effectively. In the Kmeans part, we can control the length of the final output Summarization by setting the number of Centroids.



2.1.2 Extractive summarization-Textrank

Like BERT_kmeans, Textrank also divides the article into Sentences at the beginning, and embedding through the Pretrained model.

After embedding, Sentence Embeddings use the Textrank algorithm to calculate the similarity between each Sentence. The similarity is calculated as the number of words in the intersection of the two sentences divided by the total number of words in two sentences. After getting the weight of each Node and Edge, the Sentences with the highest similarity to other sentences in the article are selected as Summarization.



2.2.1 Abstractive summarization-Graph Based

This method[6] aims to get the longer but redundant sentences in an article, that is to say the main points for an article. And there are Four major steps. First, we have to attach

part-of-speech tags (POS) to all words in the article. Second, we generate the graph representation of sentences. Third, with the graph, we can find promising paths to get candidate summary sentences and then calculate scores for those candidates. Finally, after sorting all candidates by their scores, we can determine the number of summaries that we want to see. Following is an example to further elaborate those steps.

Ex: 2 sentences about “run very fast”:

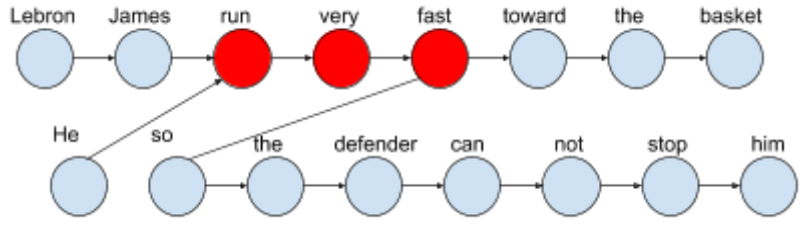
Step1. POS tagging

POS can be used to check if the word can be a valid starting node or ending node. Moreover, we can use it to check if a sentence is valid or not.

1. Lebron James **run very fast** toward the basket => Lebron/NN James/NNS run/VBP very/RB fast/RB toward/IN the/DT basket/NN
2. he **run very fast** so the defender can not stop him => he/PRP run/VB very/RB fast/RB so/IN the/DT defender/NN can/MD not/RB stop/VB him/PRP

Step2. graph representation

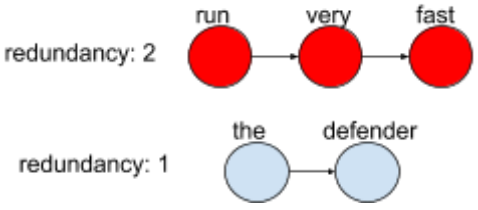
Path redundancy shared by two sentences naturally captured by graph nodes.



Graph representation

Step3. promising paths with score

Since the “run very fast” is a redundant sentence path. the redundancy must be high.



Due to the fact that, the final score = redundancy * length. We finally might get one of sentence paths like “Lebron James runs very fast so the defender can not stop him” with score = 1 * 9 + 2 * 3 = 15

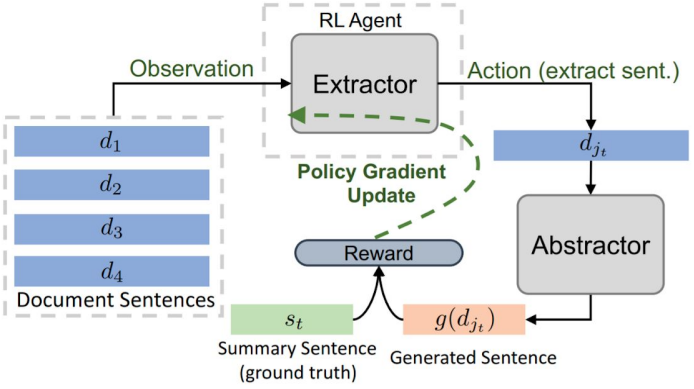
Step4. ranking the sentences

After ranking, we might get the optimal summary, “Lebron James runs very fast so the defender can not stop him” which is a longer but redundant sentence.

2.2.2 Abstractive summarization-Learning Based

Machine learning lets us easily extract features of sentences for easy classification, which can be used for an extractor to determine which sentence is important. On the other hand. a sequence-to-sequence model allows us to generate new sentences which did not appear in the original article. As a result, The model[7] combined both an extractor and an abstractor can be used for abstractive summarization. In this model, we have two steps to complete the overall training process.

Overall model:



Step1. train Extractor and Abstractor respectively.

(a) Extractor training

We can treat this training problem as a classification task. Using ROUGE with similarity, we can obtain pseudo labels to represent a positive or negative sentence like the following formula.

$$j_t = \text{argmax}_i(\text{ROUGE-L}_{recall}(d_i, s_t))$$

(b) Abstractor training

Make ground truths and Generated Sentences as training pairs and then use the training method for seq2seq directly.

Step2. Reinforce Learning (RL) to Optimize the whole model

Making Extractor as a RL agent to form a Markov Decision Process (MDP), we now can apply gradient descent to the whole model by defining a reward as following formula.

$$r(t+1) = \text{ROUGE-L}_{F_1}(g(d_{j_t}), s_t)$$

3. Experiment and Discussion

At this Section, we will use the common evaluation metrics of Text summarization: ROUGE-1, ROUGE-2, ROUGE-L to test the model, and compare the speed of these models and our observation on the model output.

For testing dataset, we use cnn/dailymail dataset. The dataset is provided by TensorFlow DataSet. On the other hand, the Learning Based model of Abstractive summarization was trained on cnn/dailymail training dataset.

3.1 Extractive summarization

For extractive summarization, because the quality of kmeans or textrank algorithm is largely based on whether the Embedding process is done well, we can check the final performance of the model by changing different Embedding models.

In the experimental setting, For BERT_kmeans, the Embedding models we choose are: DistilBERT[8], BERT, ALBERT. As for Textrank, the selected Embedding models are: Universal Sentence Encoder[9], BERT, where BERT approach is to add up the Embedding of each word as a Sentence Embedding. In addition, our output sentence only selects the most meaningful sentence in the entire article. The table below is the result of our experiment:

Model	ROUGE-1	ROUGE-2	ROUGE-L	Time(s)
BERT_Kmeans - DistilBERT	0.2639	**0.1312**	0.217	2266
BERT_Kmeans - BERT	0.2564	0.1253	0.2098	3862
BERT_Kmeans - ALBERT	0.2608	0.1284	0.2137	3726
Textrank - Universal Sentence Encoder	0.2824	0.1240	0.2390	232*
Textrank - BERT	**0.2843**	0.1131	**0.2391**	111164**

*The time is so short because we set a large converge tolerance, so it converges very quickly.
**Almost the all inference time is spend on textrank algorithm

We can find that in BERT_Kmeans, no matter which Pretrained model is used for Embedding, the time is not much different. In addition, the ROUGE scores are also very similar. The reason may be that regardless of the result of Embedding, Kmeans will select similar summarization if the output summarization is one sentence, leading to this result.

However, there is a big difference in Textrank. First of all, the parameter settings of our two models on textrank are the same. The only difference lies in the difference in the Embedding model. From the results, Universal sentence embedding can make Textrank converge very quickly, and the use of BERT to embed each token and then add it will make Textrank spend a lot of iterations to converge, which will greatly increase the time. The possible reason for this result is because we use padding to make the length of each sentence the same. Such a method may cause the BERT Summation method to add too many padding token embeddings, making each sentence embedding too long, making it difficult for Textrank to converge.

In addition, in the process of the experimental results, we found that because Extractive Summation directly concatenates the sentences with the highest scores, there will be a sense of discontinuity when reading these summarizations. Therefore, in the future, we can research how to modify the sentences obtained by Extractive summarization to make summarization more fluently. Below are examples of translation.

Part of the article:

"Ever noticed how plane seats appear to be getting smaller and smaller? With increasing numbers of people taking to the skies, some experts are questioning if having such packed out planes is putting passengers at risk. They say that the shrinking space on aeroplanes is not only uncomfortable - it's putting our health and safety in danger. ..... Gulf Air economy seats have between 29 and 32 inches, Air Asia offers 29 inches and Spirit Airlines offers just 28 inches. British Airways has a seat pitch of 31 inches, while easyJet has 29 inches, Thomson's short haul seat pitch is 28 inches, and Virgin Atlantic's is 30-31."

BERT_Kmeans output:

" seatEver noticed how planes appear to be getting smaller and smaller? More than squabbling over the arm rest, shrinking space on planes putting our health and safety in danger? Many economy seats on United Airlines have 30 inches of room, while some airlines offer as little as 28 inches ."

Textrank output(Universal sentence encoder):

"Many economy seats on United Airlines have 30 inches of room, while some airlines offer as little as 28 inches . But these tests are conducted using planes with 31 inches between each row of seats, a standard which on some airlines has decreased, reported the Detroit News."

3.2 Abstractive summarization

Because abstractive methods generate new sentences by themselves, sometimes they produce strange sentences which might have grammarly problems or difficulty reading. As a result, the Graph Based method takes advantage of the POS to determine the result summaries are correct sentence of not according to the grammar. We compared the performance in this condition. On the other hand, due to the other problem that abstractive methods sometimes generate repeating words and phrases. Therefore, the Learning Based method applies beam-search according to the number of repeated N-grams to rerank sentence candidates. In this model, we compare two results; the first one does not use beam search, and the other one adopts beam search.

Model	ROUGE-1	ROUGE-2	ROUGE-L	Time(s)

Graph-based	0.2689	0.1096	0.2353	4522
Graph Based with sentence check	**0.2719**	0.1105	**0.2379**	**2798**
Learning Based with greedy search	0.2530	0.1441	0.2129	3295
Learning Based with beam search	0.2701	**0.1525**	0.2232	11076

From the above result table, we confirm that the Graph Based method with sentence check according to POS provides better performance and even having less computation time. This is because the subsequence on the invalid paths would also be dropped in the processing time. However, the performance was not improved much obviously, and I think the reason is that the sentence check is sometimes too strict which leads to lower recall. On the other hand, the beam search adopted in the Learning Based method makes the performance become a little bit better as well. In fact the summary sentences are generated from mutually exclusive sentences, which naturally avoids the repeating problem. As a result, there is a tradeoff between the computation time and performance because the time becomes much longer after using beam search.

* The input article is the same as Extractive method.
Graph Based method's output:

"Many economy seats on United Airlines have 30 inches , some airlines offer as little as 28 inches . Ever noticed how plane seats appear to be getting smaller and smaller ."

Learning Based method's output:

"the government is happy to set standards for animals flying on planes .
'It is time that the DOT and FAA take a stand for humane treatment of passengers .
They say the shrinking space on aeroplanes is not only uncomfortable .
'In Leocha , consumer representative says ."

4. Conclusion

In this article, we studied the two methods of Extractive and Abstractive summarization. Among them, the performance of Extractive summarization is good, but the incoherence between the sentences of the extraction still needs to be improved. Besides, using Textrank with Universal Sentence Encoder will get a fast speed and stable quality, so this Extractive summarization technology is highly recommended. As for Abstractive summarization, the performance is good as well. However, we have to care more about the output sentences' grammar, repeating problems. For ideal Extractive and Abstractive summarization, the former one has better coherence in each sentence, and the latter has better coherence in overall summary sentences. Though Abstractive summarization is more complicated than Extractive summarization, this approach is more like a natural summarization done by people.

In addition to research, we test the products we have built to enable users to quickly get the latest CNN news information. We believe that in this era of information explosion, this product can let people keep up with the world!

References

[1]ROUGE: A Package for Automatic Evaluation of Summaries

[2]Leveraging BERT for Extractive Text Summarization on Lectures

[3]TextRank: Bringing Order into Texts

[4]BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

[5]ALBERT: A Lite BERT for Self-supervised Learning of Language Representations

[6]Opinosis: A Graph Based Approach to Abstractive Summarization of Highly Redundant Opinions

[7]Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting

[8]DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter

[9]Universal Sentence Encoder