Extractive Text Summarization

Arjun Sanchala
ICT Department
School of Engineering and Applied Science
Ahmedabad, India.
arjun.sanchala@gmail.com

***Abstract*— There is large amount of textual material in the world and it is growing every day by day. So, there is a great need to summarize much of this text data to shorter and focused which capture the meaningful details, so we can go thru it and understand whole the document. The goal is to create automatic summaries as good as those written by humans. I use some of the popular algorithms to summarize a news article.**

Keywords— Extractive Text summarization, TextRank, LexRank, Cosine similarity, Sentence score, Tokenization.

I. INTRODUCTION

With Drastic growth of the internet, people generating large amount of data which is in form of Image, Text etc. This resulting to research in area of Text summarization.

Automatic Text Summarization is a task to create summaries which hold key information which is given by original document. It will be shorter from original text and focused towards the original topic of document. It is very challenging task, because when we human wants to create summary, we read whole the article and understand and after that we write the summary which highlighting the main points. But computers don't have capability to understand, they can be trained, but they can't understand. Second, we can't say which summary is right and which summary is not, since everyone has different minds. So, if we give task to write summary to random humans, maybe everyone writes different summaries. It makes Automatic text summarization a very difficult task.

II. AUTOMATIC TEXT SUMMARIZER

A. Use

We are directly or indirectly uses summaries in our day to day life. Here is some example where we are using summaries: Headlines, Outline of student's notebook, Movie reviews, Biography, Weather bulletins.

B. Scope

This project is under Natural Language Processing field. In this project I am going to use some of the algorithms and some famous libraries of python to make extractive text summarizer and also explore some in-built algorithms like TextRank which is based on GOOGLE's PageRank algorithm and LexRank algorithm.

We can make text summarizer in two fields, one is to use Artificial intelligence algorithms like CNNs, Sequence to sequence model and LSTMs. And other way is to use Natural Language Processing algorithms which uses mathematical expressions and calculate importance of every sentences in that document.

C. Methods of summarization

There are two methods(approaches) of Text summarization. *Extractive* and *Abstractive text summarization. Extractive text summarization* involves the selection of phrases and sentences from the source document to make up the new summary. It will use sentences from original text and important sentences are appear in summary. *Abstractive text summarization* involves generating entirely new phrases and sentences to capture the meaning of the source document. This is a more challenging approach, but is also the approach ultimately used by humans.

III. EXTRACTIVE TEXT SUMMARIZATION

A. Introduction

I was working on Extractive text summarization. For that, I was using Python language. In Python, I used certain libraries like NLTK (natural language processing library), urllib (for grab the data from news portals), genism (for generate gold summary), wordcloud (for data visualization).

B. Flow chart

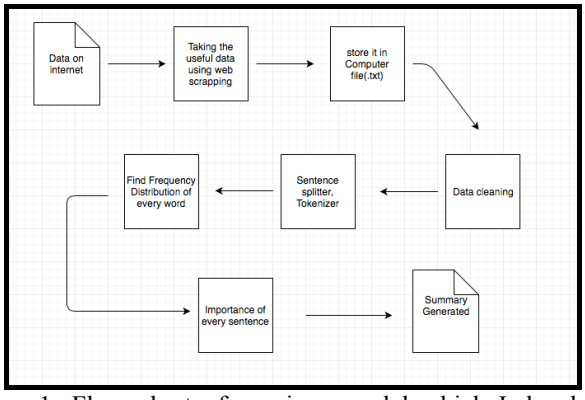

Figure 1: Flow chart of previous model which I developed erlier.

C. *Implementation*

I was taking top news from TOI website and store it in file by using beautifulsoup library. Then program gives you short news (summarize it). I also use the WordCloud, that gives you words that is used frequently in article. This is visualization technique and will helpful in summarization.

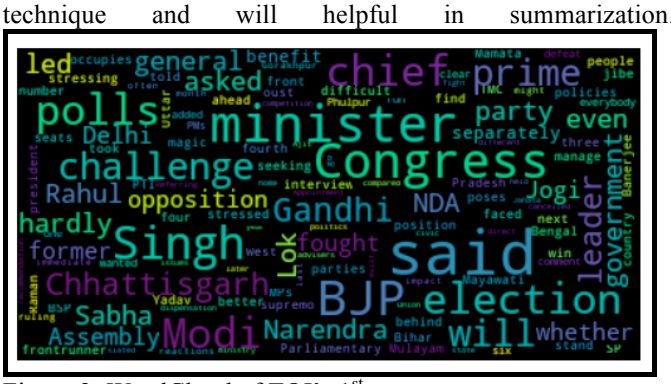

Figure 2: WordCloud of TOI's 1st news.

We can clearly visualize some of words which are frequently being used in that article. After that, I tokenize the article, which is process to break article into sentences. Next step is to clean the data. So I used stemming technique which replace words which have same meaning (i.e. 'work', 'worked', 'working' replace by 'work'). Then I removed some symbols and newlines. After that, I found the most frequent words from that article and based upon that I select sentences which uses that words. This was the basic summarizer which I build earlier and idea behind that is, if a word is frequent, then that word is important and sentences which use those words are also important. so, in summary, you can find those sentences.

This type of summarizer work well in some the article. I gave input of 25,197 words and it gave me 7636 words of summary.

The main problem with this summarizer is, it didn't have any algorithm to work on, it directly gave you summary based on frequency of words. So I work on some of the algorithm to attach in my code.

IV. Implementation of Various Algorithm

In this paper I am using two known algorithms which is very useful in summarization, TextRank and LexRank. For evaluation of this two algorithm, I am using summary which is generated by GENSIM library and then we compare with our summary generated by these algorithms.

A. *TextRank algorithm*

The TextRank algorithm is based on GOOGLE's PageRank algorithm, which is Graph based algorithm. Here, the vertices of the graph are sentences, and the edge weights between sentences are how similar the sentences are.

I am using a paragraph and at the end of the code you will be given scored sentences based upon how much they are similar to the original text. And for that we have to follow certain steps which I mention below:

Tokenize the paragraph into sentences, Tokenize sentences to bag of words, convert it to a graph (words as vertices and edges as weights), use PageRank algorithm to score the sentences.

we are defining a different relation, which determines a connection between two sentences if there is a "similarity" relation between them, where "similarity" is measured as a function of their content overlap. This relationship gives idea to user that those sentences share some common information regarding that article. So we create a link between them and give the weight based upon their similarity.

TF-IDF - A better Strategy

Rather than just counting, we can use the TF-IDF score of a word to rank it's importance.

The tfidf score of a word, w, is:

$$tf(w) * idf(w)$$

Where tf(w) = (Number of times the word appears in a document) / (Total number of words in the document)

And where idf(w) = log(Number of documents / Number of documents that contain word w).

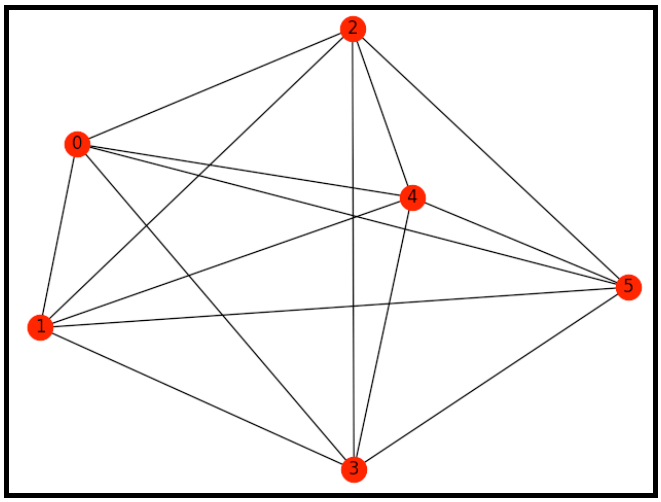

Figure 3: Sentences as vertex and edges as similarity.

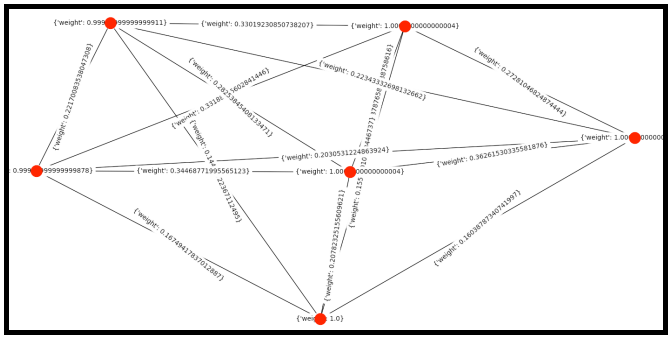

Figure 4: Weighted graph.

After getting the graph, we can apply PageRank algorithm to our matrix. It will return the importance of each sentence in given paragraph.

```
scores & sentence
(0.18512419075420891, '"Narendra Modi is one prime minister

(0.17840612141279158, 'The Congress led by Rahul Gandhi pos

(0.16662817891956777, '"It will hardly hardly manage to get

(0.16396057970024244, '"Jogi will exist (in state politics)

(0.16280210332013673, '"There are no challenges in front of

(0.14307882589305221, '"Delhi cannot be run like this," he
```

Figure 5: Scored sentences

After the ranking algorithm is run on the graph, sentences are sorted in reversed order of their score, and the top ranked sentences are selected for the summary.

Now, for cross check how our model performs, I am using GENSIM library which is widely used library for text summaries. They gave use function for directly generate the summaries of given text.

Below, you can see some of the sentences extracted by GENSIM library. So we can compare those to our model.

```
Summary created by TextRank1 function :

"Narendra Modi is one prime minister from the BJP while there are six PMs from there (the
  opposition)," he said.Asked to comment on the BJP's defeat in the Phulpur and Gorakhpur
  by-elections last month, Singh said "immediate reactions" often had an impact on by-polls and
  they might go against a ruling dispensation.But civic and Assembly polls were fought on
  different issues and could not be compared with general elections, he said.On Chhattisgarh's
  Assembly elections, slated to be held later this year, the chief minister said the BJP's
  direct competition was with the Congress.Referring to Ajit Jogi's Chhattisgarh Janata
  Congress, Singh said whether the former chief minister fought from the Congress or separately,
  the BJP would benefit.
The Congress led by Rahul Gandhi poses no challenge to Prime Minister Narendra Modi and will
  find it difficult to win even 40 or 50 seats in the 2019 Parliamentary polls, Chhattisgarh
  chief minister Raman Singh has said.Modi has his "magic" and the people of the country stand
  by him, Singh said, stressing that the BJP would do even better in the next general elections
  than it did in 2014 because of the NDA government's policies.


Summary created by GENSIM :

The Congress led by Rahul Gandhi poses no challenge to Prime Minister Narendra Modi and will
  find it difficult to win even 40 or 50 seats in the 2019 Parliamentary polls,Chhattisgarh
  chief minister Raman Singh has said.
Modi has his "magic" and the people of the country stand by him, Singh said, stressing that the
  BJP would do even better in the next general elections than it did in 2014 because of the NDA
  government's policies.
Referring to Ajit Jogi's Chhattisgarh Janata Congress, Singh said whether the former chief
  minister fought from the Congress or separately, the BJP would benefit."

Process finished with exit code 0
```

Figure 6: Comparision of TextRank summary vs GENSIM summary

B. LexRank algorithm

LexRank is an unsupervised graph based approach similar to TextRank. LexRank uses IDF-modified Cosine as the similarity measure between two sentences. This similarity is used as weight of the graph edge between two sentences. LexRank also incorporates an intelligent post-processing step which makes sure that top sentences chosen for the summary are not too similar to each other.

$$\text{idf-modified-cosine}(x, y) = \frac{\sum_{w \in x,y} \text{tf}_{w,x}\,\text{tf}_{w,y}(\text{idf}_w)^2}{\sqrt{\sum_{x_i \in x}(\text{tf}_{x_i,x}\,\text{idf}_{x_i})^2} \times \sqrt{\sum_{y_i \in y}(\text{tf}_{y_i,y}\,\text{idf}_{y_i})^2}}$$

LexRank give us the facility to set threshold value so that we can filter out sentences which has low cosine values. So, here is some of the sentences which are extracted by LexRank algorithm.

```
Cosine simmilarity :

0 0 1.00
0 1 0.97
0 2 0.94
0 3 0.87
0 4 0.97
0 5 0.95
1 0 0.97
1 1 1.00
1 2 0.93
1 3 0.88
1 4 0.97
1 5 0.94
2 0 0.94
2 1 0.93
2 2 1.00
2 3 0.85
2 4 0.94
2 5 0.89
3 0 0.87
3 1 0.88
3 2 0.85
3 3 1.00
3 4 0.85
3 5 0.83
4 0 0.97
4 1 0.97
4 2 0.94
4 3 0.85
4 4 1.00
4 5 0.97
5 0 0.95
5 1 0.94
5 2 0.89
5 3 0.83
5 4 0.97
5 5 1.00
```

Figure 7: Cosine similarity

After knowing cosine similarity, we can grab senteces which are most similar and predict that they are closer to the topic of our document.

```
LexRank summary :
"There are no challenges in front of the prime minister," he told PTI in an
  interview when asked about the challenges Modi faced ahead of the Lok Sabha
  polls.The Rahul Gandhi-led Congress is no challenge, he stressed.The party
  occupies the fourth position in Uttar Pradesh, is behind three or four
  parties in Bihar and is not a frontrunner in West Bengal, Singh said.


"Narendra Modi is one prime minister from the BJP while there are six PMs from
  there (the opposition)," he said.Asked to comment on the BJP's defeat in the
  Phulpur and Gorakhpur by-elections last month, Singh said "immediate
  reactions" often had an impact on by-polls and they might go against a ruling
  dispensation.But civic and Assembly polls were fought on different issues and
  could not be compared with general elections, he said.On Chhattisgarh's
  Assembly elections, slated to be held later this year, the chief minister
  said the BJP's direct competition was with the Congress.Referring to Ajit
  Jogi's Chhattisgarh Janata Congress, Singh said whether the former chief
  minister fought from the Congress or separately, the BJP would benefit.
```

Figure 8: LexRank summary

- Length of original document is: 2227 characters.
- Length of TextRank summary is: 1158 characters.
- Length of LexRank summary is: 1133 characters.
- Length of GENSIM's summary is: 621 characters.

V. CONCLUSION

As we see these two algorithms are graph-based algorithms, there are some other type of algorithms are available like frequency-based algorithm.

From above numerical information, we see that GENSIM gives us more short summary. that is because of our document has long sentences. So, when we tokenize it, it will separate in few sentences. So, we get long sentences in our summary but GENSIM gives those sentence but in half of the length. If we assume GENSIM's summary as "Gold summary", then we can say that TextRank perform well because all of the sentences are covered in TextRank's summary. There is somewhat different summary given by LaxRank algorithm.

VI. FUTUREWORK

As we know, thinking process of every human is different. So, if we would give same paragraph to different people, the summary generated by all is different from each other.so, we can't say which summary is perfect and that's why we can't measure how any model is performing. All we can do is; we minimize the length of summary so that reader can easily interpret the meaning of whole paragraph.

We can also build abstractive text summarizer but it is very difficult for computers to address same summary using different phrases unlike humans. Only way to build that type of text summarizer is, to use of Artificial intelligence.

REFERENCES

[1] A paper on Text Summarization techniques: https://arxiv.org/pdf/1707.02268.pdf

[2] A paper on TextRank: https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf

[3] I https://rare-technologies.com/text-summarization-in-python-extractive-vs-abstractive-techniques-revisited/

[4] https://en.wikipedia.org/wiki/Tf–idfR. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[5] https://rare-technologies.com/text-summarization-with-gensim

[6] online Natural language processing couse

[7] https://thetokenizer.com/2013/04/28/build-your-own-summary-tool/

[8] https://www.analyticsvidhya.com/blog/2017/10/essential-nlp-guide-data-scientists-top-10-nlp-tasks/