Text Summarization Based on Text Rank

Name: Weisen Cheng  
Department: *Computer Science Department*   
Organization: *The University of Texas at Dallas*Address: Richardson, Tx NetID: wxc170730  
StudentID: 2021373412

Name: Di Jin  
Department: *Computer Science Department*   
Organization: *The University of Texas at Dallas*Address: Richardson, Tx NetID: dxj170930  
StudentID: 2021377200

Name: Xiaoyu Zhang  
Department: *Computer Science Department*   
Organization: *The University of Texas at Dallas*Address: Richardson, Tx NetID: xxz173130  
StudentID:

*Abstract*—we will introduce a graph-based ranking model for text processing- Text Rank. It resorts the sentences according to ability to conclude the article. By using popular pre-trained word embedding vectors and PageRank, the most wanted sentences are extracted.

Keywords—text rank, word vector, undirected graph, page rank

# Introduction

With growing digital life and ever growing. Many people don't have the time to go over whole context. Text summarization is one of the most challenging  problems in the field of Natural Language Processing. It starts from 1960's with methods like title words in text, location of words, count of words appear. Many studies shows great algorithm to this challenging topic. Text summarization can be divided into two categories extractive summarization and abstractive summarization. Extractive summarization extracts several parts, such as phrases and sentences, from a piece of text and combine them as sentences to summarize the text.  Abstractive summarization use advanced NLP method to generate entire new sentence which may not be in the original text. In this paper, we use Text Rank to extract important sentence from text.

Unsupervised extraction removes the need for training data. It deals the problems in a different way. Instead of trying to learn explicit features that summarize the text, the Text Rank algorithm find structure of the text to determine words that are important in the article. Text Rank can be used on any arbitrary text, and we combined it with pre-trained word embedding vectors to improve accuracy.

Text Rank algorithm find structure of the text to determine words that are important in the article. Text Rank can be used on any arbitrary text, and we combined it with pre-trained word embedding vectors to improve accuracy. To understand Text Rank, we should start with graph. Edges are created based on word concurrence and weight is cosine similarity. Vertices are sentences. All the vertices are connected since it should calculate similarity for each sentences with all.

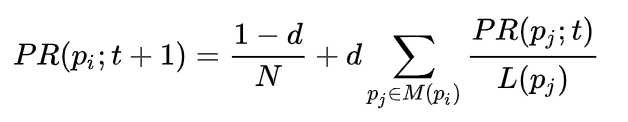
# Pagerank

PageRank is an algorithm used by Google Search to rank web pages in their search engine results. It works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites[1].

Formally, we can compute PageRank in and iterative way. At t=0, an initial probability distribution is assumed, we used PR(pi;0)=10 in our experiment, which means page i at time 0 has the weight 10.

In each iteration, we recompute the rank of each page according to the connectivity between the pages. The formula is shown in Figure 1, where d is the damping factor. The damping factor is usually set 0.85,which will remain the same in out experiment.

First, confirm that you have the correct template for your paper size. This template has been tailored for output on the A4 paper size. If you are using US letter-sized paper, please close this file and download the Microsoft Word, Letter file.



1. the formula used to compute pagerank.

# Textrank

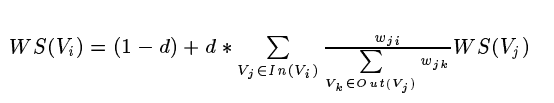
## Undirected graph

The PageRank algorithm is applied on directed graphs traditionally. In our experiment, we need to apply it on undirected graphs since we need to compute the similarity between every two sentences. In which the in-degree and out-degree of a vertex would be identical.

## Weighted graph

In the context of Web surfing, it is unusual for a page to include multiple or partial links to another page, and hence the original PageRank definition for graph-based ranking is assuming unweighted graphs[2].

However, in out experiment. The similarity between the sentences, which could be considered as the edge between the vertices, has a value. Therefore, it would be essential to take the weight of an edge into account. Consequently, we will introduce a new formula which is used in our experiment which will add the weight of the edges to count the rank of every vertex. It will be similar to the formula discussed above in Figure 2.



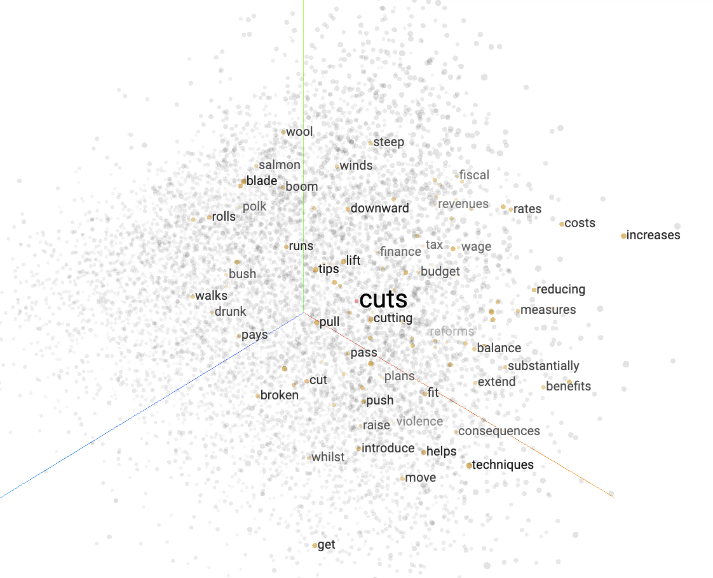
1. the formula used to compute pagerank with weighted edges.

## Word Vector

Word vectors are simply vectors of numbers that represent the meaning of a word. Word vectors represent words as multidimensional continuous floating point numbers where semantically similar words are mapped to proximate points in geometric space.

In simpler terms, a word vector is a row of real valued numbers (as opposed to dummy numbers) where each point captures a dimension of the word’s meaning and where semantically similar words have similar vectors. This means that words such as cuts and cutting should have similar word vectors to the word cut(because of the similarity of their meanings), whereas the word get should be quite distant[3].

The word vectors of cuts, cut and cutting in three dimension are shown in Figure 3. In the graph, it is clearly that the word cuts which we highlighted is in the middle of the space. The word cutting is just beneath the word cuts. The word cut is on the bottom-left of the word cuts. However, the word get is a long distance away from these three words we mentioned above, on the bottom side of the space.



1. the word vectors of cuts, cut and cutting shown in the three-dimention.

## Keyword Extraction

The graph we mentioned above should be built from the document with text unit. In this case, people may want to add all the words into the text unit, however, this is wasting time. The essential words which are the skeleton of a sentence: nouns, verbs, adjectives are enough for key extraction. Researchers has proved that nouns and adjectives are the two most influential parts of a sentence. Therefore, we need to remove some redundant words before we setting the sentence as a text unit. After that, we can use it to build an undirected graph. The nodes represent the sentences and the edges will represent the relationship between two sentences with weight. Once the graph is made, we can use the optimized page rank algorithm on the graph until convergence.

## TextRank

Text Rank is a graph-based ranking algorithm for NLP. PageRank is an essential part of it. In the key phrase extraction part, it will builds a graph based on the text units as vertices and some measure of semantic or lexical similarity between the text units as edges. Unlike the PageRank runs on an directed graph, the edges in text rank are typically undirected and have weight which reflect the similarity. Once the graph is constructed, it is used to form a stochastic matrix, combined with a damping factor (as in the "random surfer model"), and the ranking over vertices is obtained by iteratively calculate the rank using the formula discussed above until converged.

The vertices should correspond to what we want to rank. In this experiment, we want to extract a few sentences from a large amount of text, which means the vertices should be different sentences. The edges are created based on sentence correspondence in this application using the cosine similarity between the vectors of the sentences. The vectors of the sentences is calculated by sum the word vectors of all the words in the sentences.

# Implementation

In this experiments, there are 4 parameters: data set, vector model, number of iterations, selection ratio.

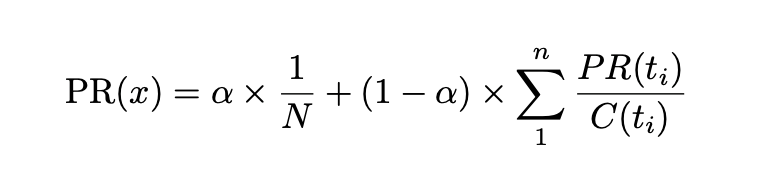
The data set is a collection of CNN news downloaded from Kaggle dataset. Figure 4 shows the composition of the dataset. There are 9 columns in this dataset and we will take advantages of the title and content columns in this experiment.



1. The composition of the dataset.

The vector model used in the experiment is the GloVe word embeddings. GloVe Word Embeddings are vector representation of words. We can use it to create vectors for words and combine them to get the vectors of sentences. This pretrained model helps minimize the word training process as well as offering more accurate word vectors. In this experiment, we use the glove.6B.100d model which will transfer a word to a 1\*100 vector.

The number of iterations is used for the PageRank process. The formula used for PageRank could be found in Figure 5. In this formula, the page rank of a vertex is calculated using its neighbor vertices. However, we can only calculate the page ranks one by one. We need to do several iterations until the result converge. Of course more iterations will take much more time.



1. the formula used as pagerank formula in our experiment.

Finally, the selection ratio is used to decide the number sentences need to be extracted. We need to use a ratio instead of a constant number in case the number of sentences in the content is too large. In this case, choosing several sentences does not make sense obviously, besides, the accuracy will decrease dramatically.

The cluster used in the experiment is an AWS cluster with the setting Ganglia 3.7.2, Spark 2.4.4, Zeppelin 0.8.2 and 6 instances.

## Data Preprocessing

In the data processing part, we firstly select the rows we need (content and title). Then tokenized the content into sentences with the specific RegEx utilizing Spark RegexTokenizer. After that we removed the stop words using Spark StopWordsRemover. At last we changed all the words into the lowercase.

## Vector Extraction

We used the GloVe Word Embeddings mentions in III.C to transfer the words in a sentence to word vectors, and sum the word vectors in the same sentence and get the sentences vectors. Then calculate the cosine similarity between all the sentence pairs.

## Text rank calculation

Next we use the formula mentioned above to calculate the PageRank of all the sentences and set the number of iterations. It will return a sorted array with the index of the sentence and its page rank.

## Text Generation

Using the text rank calculated above, we can extract the specific sentence in the sentences array and write it back to the file.

# Evaluation and analysis

## Evaluation

The results are evaluated using the title column. By calculating the cosine similarity between the title and every sentence in the content, we can get the weight of the edges in the graph. Then use the optimized page rank algorithm mentioned above. We can get the top k similar sentence in the paragraph, which are the ideal result we should achieve in our experiment.

Then we use the selection ratio to compare the first N idea result sentences and the first N sentences in our experiment result. We can get a accuracy of one paragraph in a row. Then we calculated the total accuracy value from 0 to 1.

## Analysis

The results are evaluated using accuracy, time consuming and selection ratio. For comparison purpose, we tried different combinations of iterations and ratio. The result is shown in Table 1. The table shows us that the more iterations we trained, the higher accuracy we get. However, more iterations will make the time consuming increases exponentially. When the number of iterations comes to 1, we can get only 49% accuracy. When the number of iterations rise to 10, we can get 69% accuracy. When the iteration number increased 50. The results are nearly the same. Thus, the result of our data set will converge at 10 iterations.

Notice that the text rank algorithms ignore the connectivity of the paragraph, it will only extract the most important sentences. Besides, text rank algorithm will generate a ranked list of sentences in the paragraph, it is easy for us the decide if we need to generate a long summary.

1. relationship between iteration number, slection ratio and accuracy

|  |  |  |
| --- | --- | --- |
| iteration | Selection ratio | Accuracy |
| 1 | 0.2 | 0.29972148 |
| 1 | 0.3 | 0.3945073 |
| 1 | 0.4 | 0.484363802 |
| 5 | 0.2 | 0.426578 |
| 5 | 0.3 | 0.573658 |
| 5 | 0.4 | 0.612054 |
| 10 | 0.2 | 0.5759627 |
| 10 | 0.3 | 0.6927219 |
| 10 | 0.4 | 0.7765254 |
| 50 | 0.2 | 0.5759627 |
| 50 | 0.3 | 0.6927219 |
| 50 | 0.4 | 0.7765254 |

# Conclusion and Future work

In this project, we introduced text rank which is a graph based ranking model for text summarization. We also introduced out implementation and experiment on text summarization. In the future, we will explore the seq2seq model and the RNN LSTM model. We will do some experiment on them and compare with the text rank algorithm.

##### References

1. [Facts about Google and Competition](https://web.archive.org/web/20111104131332/https:/www.google.com/competition/howgooglesearchworks.html). Archived from [the original](https://www.google.com/competition/howgooglesearchworks.html) on 4 November 2011. Retrieved 12 July 2014.
2. TextRank: Bringing Order into Texts. 2004.  Rada Mihalcea and Paul Tarau [Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing](https://www.aclweb.org/anthology/volumes/W04-32/)(404-411).
3. Introduction to Word Vectors. 2018 <https://medium.com/@jayeshbahire/introduction-to-word-vectors-ea1d4e4b84bf>