Extractive Text Analysis Using TextRank

Mehar Singh

I. TextRank Explanation

A. PageRank

The PageRank [1] algorithm was originally introduced to rank web pages for Google's search engine. It is, in essence, vertex voting.

The PageRank algorithm assumes that important pages have incoming edges from other important pages. The algorithm is as follows:

$$P_r(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{P_r(V_j)}{|Out(V_j)|}$$

$In(v)$ are all the vertices that v has an incoming edge from. $Out(v)$ are all the vertices that v has an outgoing edge to.

The importance of a page v is therefore the sum of importance of each page that leads into v divided by the total number of pages it leads to. Therefore, the impact of a web page having an incoming edge from an important web page with *many* outgoing edges would be less significant than the impact of an incoming edge from an important web page with *few* outgoing edges.

The damping factor is considered so that new or small web pages do not receive insignificant importance.

Arbitrary values can be assigned as each vertices scores initially. After many iterations of PageRank, the scores reach a convergence.

B. TextRank

TextRank [2] is a variation of the PageRank algorithm. It assumes that important text units (whether they be individual tokens or sentences) that should be included in a summary of a larger document are connected to other important text units.

An edge between text units shows some form of relation between them. They can represent large semantic similarity, syntactic similarity, or other metrics.

Conventionally, the TextRank algorithm is written as follows:

$$S(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

C. TextRank-Based, Semantical Text Summarization

A semantical text summarization [3] using weighted edges was recently proposed, where the weights of the edges represent the semantic similarity between two text units.

The algorithm is thus modified to:

$$W(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{W_{ji}}{\sum_{V_k \in Out(v_j)} W_{jk}} W(V_j)$$

where W_{ij} represents the similarity between text units V_i and V_j. Similarity functions (such as cosine similarity) are used on embeddings from word2vec and doc2vec models to estimate the similarity between tokens and sentences respectively.

To extract keywords, summaries (important sentences) are first extracted. These are grouped and used as the input for keyword extraction.

II. Implementation Details

Code for the complete implementation can be found here[1]. It takes some inspiration and code from a different TextRank implementation[2].

A. Semantical Embedding Models

Due to the (1) lack of training documents available to train a word2vec and doc2vec models and (2) memory limitations, spaCy's en_core_web_md English model was used which is pretrained and smaller than other advanced models such as Google's word2vec.

B. Filters

Filtering is done to only evaluate nouns or proper nouns as keywords. Further, unknown words are ignored.

C. Edge Creation and Weights

The graphs for summarization are undirected and fully-connected with edge weights given by the semantic similarity between sentences.

The graphs for keywords contains edges between tokens if they can be grouped by a window. As such, these edges are also undirected. The default window size is 1. The weight of the edges is the semantic similarity between words.

D. Constants

Iterations will occur until a convergence with a 0.0001 threshold or, otherwise, a maximum of 100 iterations.

As stated above, the default window size is 1.

The default size of summarization is $p = 0.2$ of the original input text. The default number of keywords provided is $N = 5$.

III. Results

There was difficulty using a gold standard dataset to compare keywords and summarizations. As such, the results and takeaways are observational.

[1]https://github.com/mehardsingh/extractive-summarization

[2]https://github.com/summanlp/textrank/blob/master/summa/pagerank_weighted.py

A. Summarization

For brevity, sample summarizations are found in the shared repository under the examples directory[3].

B. Keyword Extraction

Comparison of a few keyword extractions from semantic TextRank implementation versus generic TextRank can be found in **Table 1**.

C. Overall

Through experimentation, it *seems* as if semantic TextRank does better at conveying the meaning and knowledge of text than the generic TextRank, but only slightly.

The summaries are coherent with great flow between sentences, possibly due to the sentences having strong semantic similarity to one another.

Most of the keywords in **Table 1** are shared between the generic TextRank and the semantic version. Only for the keywords on citation [6] does there appear to be a significant difference. The generic TextRank seems to prioritize frequent words such as *year*, which isn't as related to the central theme of *technology* as some of the words the semantic version picked (*research, system, technical*). These words were potentially picked up by the semantic TextRank due to to actual meaning of the words and sentences they are part of.

Further testing is necessary to determine how significant the difference is between the two summarization models. In [3], the performance difference on the English language showed slight improvement.

IV. Questions

A. Can you summarize the state of the art of text summarization you learned in no more than one page?

Yes, please refer to the first page of this paper.

B. Are you able to reproduce the results? Observe, analyze and interpret the results.

Yes, please refer to the second page of this paper. Future testing can also be done with gold standard datasets such as the ones mentioned in [3], but there was difficult in using them.

C. Can you demo how your extension work? Can you make it more useful?

I am still trying to get the Chrome extension to work, but I've been unsuccessful so far. However, a demo of my existing solution (through the command line) is provided[4].

There are a few ideas to make this project more useful:

1) Clustering can resolve the issue in which there are multiple topics in a single document. By clustering sentences before summarization, a summary for each topic within the document can be created.
2) Using a dependency graph to form edges between tokens for keyword extraction can potentially be used to make the keywords even more useful. I tried this idea but the keywords were much worse, potentially because some words such as the root had no incoming edges. Perhaps treating the dependency graph as undirected can produce better results.
3) User input can also be received to know the context of the information the user is looking for. This can be especially important if a document contains many topics.

D. What have you learning from the exercise above?

I learned a lot about why TextRank and PageRank work. In previous classes, I was introduced PageRank using linear algebra without as solid of a foundation regarding how the algorithm functions.

I also learned about many NLP tools such as word2vec, doc2vec, spaCy, and genSim.

Lastly, I gained invaluable experience researching, reading papers, and implementing novel concepts without much prior information.

E. Do you think the text summarization results were satisfactory for practical usage? What can be improved?

Yes, I believe the text summarization results were satisfactory since they did a solid job represented the knowledge of the text. It was also simple to use which makes it practical for frequent usage. Of course, the summarization results could improve with some of the ideas mentioned in question C.

F. What new applications can be built using text summarization? Propose a cool application you can build that does not exist today.

One new application of text summarization is on school notes. With the prevalence on digital notetaking, students often rely on keywords to search for specific notes on their devices. However, this approach often fails if students enter synonyms or other words that do not exactly match the text in their notes verbatim. Implementing a semantic searcher that takes in a phrase and returns documents whose content matches that phrase on a knowledge level could prove to be extremely useful. A specific version of this could have the inputs be questions (such as *What is the difference between protons and neutrons?*) and the outputs be documents that can answer such questions.

[3]https://github.com/mehardsingh/extractive-summarization/tree/master/examples

[4]https://uofi.box.com/s/7j8nxntkgji8adhnr6hm9b6ptltl4sqd

V. Table 1

Citation	Semantic	Generic
[4]	chocolate, milk, cacao, flavor, cocoa	chocolate, milk, flavor, cacao, cocoa
[5]	croissant, cup, bakery, style, sweet	croissant, bakery, city, pastry, crescent
[6]	founder, technology, research, system, chemical	technology, self, year, life, carbon

References

[1] S. Brin and L. Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer networks and ISDN Systems*, 30(1-7)

[2] Mihalcea, R. (2004). Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 170-173

[3] Samizadeh, Mina. ”Graph-based Semantical Extractive Text Analysis.” arXiv preprint arXiv:2212.09701 (2022)

[4] https://tcho.com/blogs/news/dark-chocolate-vs-milk-chocolate-what-s-the-difference

[5] https://altohartley.com/a-brief-history-of-the-croissant/#: :text=Croissants%20are%20a%20style%20of,in%20various%20shapes%20and%20sizes

[6] https://www.weforum.org/agenda/2021/10/top-emerging-technologies-10-years/