A COMPARISON OF AUTOMATIC EXTRACTIVE TEXT SUMMARIZATION TECHNIQUES

Alex Day[1] , Soo Kim[1]
[1]Clarion University of Pennsylvania, Computer Information Science Department
A.D.Day@eagle.clarion.edu, skim@clarion.edu

ABSTRACT

Text summarization is a method of shortening a document by producing a set of representative information. It can be generally categorized into extractive and abstractive summarization. Extractive summarization - the focus of this paper - is extracting sentences and phrases that are already present within the document in order to produce an outline. Term Frequency - Inverse Document Frequency is a score-based summarization technique that assigns a numerical value to each term in the document and then scores the relative importance of each sentence. TextRank is a graph-based summarization technique which uses a graph data structure to rank the document into the relevant sentences. This paper describes those two extractive text summarization methods and presents the results of running the algorithms on three different corpora: *Moby-Dick* by Herman Melville, a selection of Reuters news articles, and a selection of posts on Reddit. The algorithms rank the sentences by relevance to the document, take the top five sentences, and return them as a summary.

KEY WORDS

Natural Language Processing, Extractive Summarization, TF-IDF, TextRank, Graph Summarization, Ranked Summarization

1 Introduction

The goal of automatic summarization is finding a small amount of information that is most representative of a larger set of information. This can be done with images, video, or text. The need for efficient and accurate summarization algorithms has only risen with the onset of the Internet. According to the study done by the Media Insight Project [1], about 6 in 10 Americans only read the headline of a news article and delve no deeper than that. If people are willing to read one sentence headline, they may also be willing to read a 5-sentence summary. This summary would result in the people getting more information about the story, letting them be better informed.

Automatic summarization is categorized broadly into two techniques: extractive and abstractive. Extractive summarization takes words and sentences already within the document to form a summary, whereas abstractive summarization tries to understand the information within the document and summarize it by creating new sentences.

Arguably, one of the most cited and earliest works in automatic summarization is that of Luhn [2], which provides an overview of work by IBM in automatically generating technical paper abstracts using an early TF-IDF-esq algorithm. The original idea for this paper was inspired by Samir Bajaj [3] and his paper entitled *Shakespeare in One Hundred Words*. Samir investigated four abstractive summarization methods, such as TF-IDF–based relevance ranking, K-means clustering algorithm, singular value decomposition, and PageRank. In addition, Dipanjan Das, et. al [4] surveyed different summarization methods: naive-Bayes methods, rich features and decision trees, hidden Markov models, log-linear models, neural networks and third party features, deep natural language analysis methods, abstraction and information fusion, topic-driven summarization and maximal marginal relevance, graph spreading activation, centroid-based summarization, and so on. Dipanjan Das, et. al discussed some unconventional approaches that can be useful in the future of summarization research and elaborated some evaluation techniques as well as the standards for evaluating summaries automatically.

This paper focuses on extractive summarization methods: Term Frequency - Inverse Document Frequency (TF-IDF) and TextRank. These algorithms summarize a document in completely different ways. TF-IDF scores each sentence on relevance to the document as a whole, while TextRank is a graph-based sorting algorithm that separates each sentence into a node and links it to other sentences based on similarity. The methods TF-IDF and TextRank are explained in Section 2 and Section 3, respectively. The implementation details and the results of running these algorithms on three different corpora are presented in Section 4 and Section 5. Finally, the conclusion is given in Section 6.

2 Term Frequency - Inverse Document Frequency

Term Frequency-Inverse Document Frequency (TF-IDF) [5] is one of the most used weighting schemes for term importance. TF-IDF assigns a value to each word in a document and this assigned score indicates the importance of the word to the document. It is proportional to the frequency of the term in the document and offset by the term's use throughout the whole corpus. This offset ensures that common words such as "the", "and", "or", etc., do not skew the score since they have such low inverse document frequencies. The calculations used are shown in Equation 1. It should be noted that Equation 1 is one of the ways to calculate a TF-IDF score [3] and other non-logarithmic scales can be used, depending on the desired outcome.

$$\begin{aligned} tf(t,d) &= f_{t,d} \\ idf(t,D) &= log\frac{|D|}{|\{d \in D : t \in d\}|} \\ tfidf(t,d,D) &= tf(t,d) \times idf(t,D) \end{aligned} \tag{1}$$

where t = Term to calculate the score of
d = Document to calculate the score relative of
D = Corpus of documents such that $d \in D$

The *tfidf* is a score that is assigned only to a specific word within a document, so it cannot be used directly to assign importance-values to a whole sentence. In order to get an importance-value of a whole sentence, the average of each *tfidf* score is taken for every word within that sentence as shown in Equation 2.

$$\frac{1}{|S|} \times \sum_{w \in S} tfidf(w,d,D) \tag{2}$$

where w = Word present in d to calculate the score of
S = Set of sentences in d
d = Document to calculate the scores relative of
D = Corpus of documents such that d ∈ D

Algorithm 1 uses both of these equations. It calculates the *tfidf* score for each sentence in a document and then returns a list of the sentences within the document sorted by the sentence scoring. The term frequency score for each term is individual for the document that the term is in. The inverse document frequency of a term will only have to be recalculated each time when a document is added to the corpus. The order of the sentences as they appear in the original document is also important to the summary. In order to represent this fact, the sentences in the summary are sorted by the original location of the sentences within the document.

Algorithm 1: Return the ranked sentences for a document using TF-IDF

```
Input: A set of documents, $C$
       A document to summarize, $D$, such that $D \in C$
Output: A list of sentences sorted based on relevance
tfidf_scores[] ← 0;
for sentence ∈ D do
    tfidf_sum ← 0;
    for word ∈ sentence do
        tfidf_sum += $tfidf$(word, document, corpus);
    end
    tfidf_scores[word] ← $\frac{tfidf\_sum}{|D|}$;
end
return sorted(tfidf_scores)
```

3 TextRank

PageRank is a web page ranking algorithm developed for use with the Google search engine [6]. PageRank determines the importance of each web page returned in the search results through the number of web pages that link back to it. For example, a page such as an article on Wikipedia is important because many pages are linked to it. TextRank [7] takes the same graph-oriented approach as PageRank; however it deems a node in the graph important if that node is similar to many other nodes.

TextRank begins by separating each of the sentences in documents into nodes in a graph. Each node is linked to every other node with the weight of that edge being the similarity score. The most relevant sentences are represented by the nodes with the highest cumulative total of all edges that are connected to the node. The algorithm used in the code for this paper is shown in Algorithm 2 and an example of a document similarity graph is shown in Figure 1. In this example, the document consists of three sentences S_1, S_2, and S_3. S_1 has the similarity score 0.3 with S_2 and the similarity score 0.9 with S_3, so the sum of the similarity scores for S_1 is 1.2. The three sentences would be ranked in $[S_1, S_3, S_2]$ with their sums of similarity scores of all connected edges being $[1.2, 1.0, 0.4]$, respectively. As the result, S_1 would be ranked as the most relevant sentence and selected for the summary.

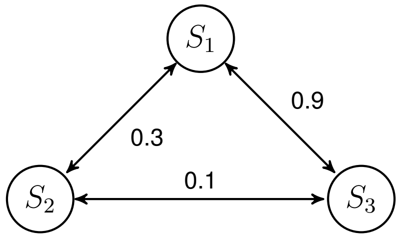

Figure 1: A sentence similarity graph for a document containing the sentences S_1, S_2, and S_3 represented as nodes on the graph where the edges represent the sentence similarities

TextRank relies on a concept called sentence embeddings to calculate the sentence similarity. A sentence embedding is a vector representation of that sentence. This vector representation is usually produced by a neural network that has been trained on a corpus. The neural network maps the sentence to a vector representation. It can glean this information from position in the document or surrounding sentences.

Cosine similarity is used to calculate the similarity of two

Algorithm 2: Return the ranked sentences for a document using TextRank

```
Input: A set of documents, $C$
       A document to summarize, $D$, such that $D \in C$
Output: A list of sentences sorted based on relevance
G ← empty graph ;
for sentence $\in$ $D$ do
    G.add_node(sentence);
end
for outer node $\in G$ do
    for inner node $\in G$ : inner node $!=$ outer node do
        G.add_vertex(outer_node, inner_node,
          similarity(inner_node, outer_node));
    end
end
sentence_scores[] ← 0;
for node $\in G$ do
    sentence_scores[node] = sum(G.get_edges(node));
end
return sorted(sentence_scores)
```

vectors rather than Euclidean distance due to the high dimensionality of the data. Cosine similarity measures the cosine of the angle between two vectors, whereas the Euclidean distance calculates the length of the line segment that connects them. For example, two similar vectors, $the\, man\, is\, good$ and $the\ king\ is\ just$, could be far apart from each other in distance, but this may only due to the different context they are used in. However, the result of the cosine similarity is the angle between the two vectors, so it will take into account more the similarity of specific indices of each vector.

It may be simpler to think first about word embeddings. The same technique can be used to produce vectors for words. Translating words to vectors opens up the possibility of performing vector operations on words. For example, one can produce the vectors for king, queen, woman, and man. Then the vector resulting from the operation $king - (man + woman)$ would be relatively close to $queen$. As well as simple addition and subtraction, the same similarity calculations for sentence embeddings can be performed in word embeddings.

4 Implementation

All of the programs were written in Python 3.6, using the following packages: NetworkX [8], PRAW [9], NLTK [10], and NumPy [11]. The codes can be found at the authors' GitHub repository [12].

5 Results

These two algorithms have been tested on three separate corpora composed in three different styles. The first is the novel *Moby-Dick* by Herman Melville, which is a semi-formal fictional novel. The second is a corpus of Reuters news articles. The third, and last corpus, is informal posts on the Reddit subreddit, "Legal Advice." The main purpose of testing the algorithms on three separate corpora was to determine how each algorithm will act in different use cases. Section 5.1 presents the summary results of *Moby-Dick*, Section 5.2 presents the summary results of Reuters news articles, and Section 5.3 presents the summary results of Legal Advice subreddit. Note that each summary contains five sentences as a result.

- On the contrary, passengers themselves must pay.
- Whaling voyage by one ishmael."
- For to go as a passenger you must needs have a purse, and a purse is but a rag unless you have something in it.
- Why upon your first voyage as a passenger, did you yourself feel such a mystical vibration, when first told that you and your ship were now out of sight of land?
- Right and left, the streets take you waterward.

Figure 2: Summary of Chapter 1 of *Moby-Dick* using TF-IDF

5.1 Moby-Dick

The complete work of *Moby-Dick* was obtained through the NLTK package [13]. This novel was then split into chapters and each chapter was summarized. The corpus that was used contained all 135 chapters and approximately 253,129 words. Figure 2 shows the summary result of Chapter 1 of *Moby-Dick* using TF-IDF, and Figure 3 shows the summary result of Chapter 1 of *Moby-Dick* using TextRank.

5.2 Reuters News Articles

The second round of testing was done on another corpus from NLTK [13]. The corpus contains 10,000 news articles from Reuters. These articles combine works from different writers which produce interesting results. However, they are all achieving the same tone and follow Reuters style guide. Figure 4 shows the summary result of Reuters news articles using TF-IDF, and Figure 5 shows the summary result of Reuters news articles using TextRank.

5.3 Legal Advice Subreddit

The last corpus is a collection of posts from a site called Reddit. Reddit is a website that allows the general public to post within forums dedicated to specific topics. The forum chosen for this corpus is called "Legal Advice". This forum contains posts by users describing situations for which they believe they need advice from legal counsel. Using the Python package PRAW [9], the top 1,000 most liked posts of the last 30 days were retrieved, and then the posts long enough to warrant a summary were passed through the algorithms. Figure

- Deep into distant woodlands winds a mazy way, reaching to overlapping spurs of mountains bathed in their hill-side blue.
- Why did the poor poet of tennessee, upon suddenly receiving two handfuls of silver, deliberate whether to buy him a coat, which he sadly needed, or invest his money in a pedestrian trip to rockaway beach?
- Well, then, however the old sea-captains may order me about – however they may thump and punch me about, I have the satisfaction of knowing that it is all right; that everybody else is one way or other served in much the same way – either in a physical or metaphysical point of view, that is; and so the universal thump is passed round, and all hands should rub each other's shoulder-blades, and be content.
- But wherefore it was that after having repeatedly smelt the sea as a merchant sailor, I should now take it into my head to go on a whaling voyage; this the invisible police officer of the fates, who has the constant surveillance of me, and secretly dogs me, and influences me in some unaccountable way – he can better answer than any one else
- With other men, perhaps, such things would not have been inducements; but as for me, I am tormented with an everlasting itch for things remote.

Figure 3: Summary of Chapter 1 of *Moby-Dick* using TextRank

- New crop sales were also light and all to open ports with June/July going at 1,850 and 1,880 dlrs and at 35 and 45 dlrs under New York july, Aug/Sept at 1,870, 1,875 and 1,880 dlrs per tonne FOB.
- March/April sold at 4,340, 4,345 and 4,350 dlrs.
- April/May butter went at 2.27 times New York May, June/July at 4,400 and 4,415 dlrs, Aug/Sept at 4,351 to 4,450 dlrs and at 2.27 and 2.28 times New York Sept and Oct/Dec at 4,480 dlrs and 2.27 times New York Dec, Comissaria Smith said.
- Cake sales were registered at 785 to 995 dlrs for March/April, 785 dlrs for May, 753 dlrs for Aug and 0.39 times New York Dec for Oct/Dec.
- Liquor sales were limited with March/April selling at 2,325 and 2,380 dlrs, June/July at 2,375 dlrs and at 1.25 times New York July, Aug/Sept at 2,400 dlrs and at 1.25 times New York Sept and Oct/Dec at 1.25 times New York Dec, Comissaria Smith said.

Figure 4: Summary of a news article using TF-IDF

- There are doubts as to how much of this cocoa would be fit for export as shippers are now experiencing difficulties in obtaining +Bahia superior+ certificates.
- March/April sold at 4,340, 4,345 and 4,350 dlrs.
- April/May butter went at 2.27 times New York May, June/July at 4,400 and 4,415 dlrs, Aug/Sept at 4,351 to 4,450 dlrs and at 2.27 and 2.28 times New York Sept and Oct/Dec at 4,480 dlrs and 2.27 times New York Dec, Comissaria Smith said.
- Cake sales were registered at 785 to 995 dlrs for March/April, 785 dlrs for May, 753 dlrs for Aug and 0.39 times New York Dec for Oct/Dec.
- Liquor sales were limited with March/April selling at 2,325 and 2,380 dlrs, June/July at 2,375 dlrs and at 1.25 times New York July, Aug/Sept at 2,400 dlrs and at 1.25 times New York Sept and Oct/Dec at 1.25 times New York Dec, Comissaria Smith said.

Figure 5: Summary of a news article using TextRank

- I will just be saying MY home or MY driveway so I don't have to keep typing "my parent's driveway" or "my parent's home" over and over.
- My parent's neighbor's kid (a very immature 20 year old) has a beater he leaves parked in front of my parent's front yard.
- Now the parents are retaliating too.They finally moved the beater, but only to move their cars from the driveway to taking up the two spaces in front of our yard adjacent to their driveway.
- The one car parked just enough to have the front poking into our driveway.
- TLDR - neighbors parked all their cars in front of my parents home and wont move them, only rearrange them.

Figure 6: Summary of a Reddit posts using TF-IDF

6 shows the summary result of Reddit legal advice posts using TF-IDF, and Figure 7 shows the summary result of Reddit legal advice posts using TextRank.

6 Conclusion

Both TF-IDF and TextRank algorithms produced a much more representative summary of the Reddit posts, while they performed decently with the fictional works and the formally styled news articles. This disparity between summary quality may be because the Reddit posts are focused only around one situation that happened, whereas in both *Moby-Dick* and the news articles, the reader is presented with a background story and sometimes a story arc. The algorithms presented for extractive summarization do not perform optimally when presented with a story line. However, if a summary needs to

- These neighbors moved in a year or two ago and have made life so uncomfortable for my parents they are actually talking about selling their home to move, their marriage home with ALL the memories.
- He has it parked in the middle so that it takes up ALL the space and no one can park on either side of it without blocking a driveway.
- I assumed mom would just tell 'Billy, go move your damn car' or something and it would be taken care of.
- I get the dad yelling at me to *** off and get off his property as the mom (from another room) starts bellowing about how I did NOT just tell her how to parent and he can do whatever he wants and **** me and my parents.
- Not only do they say the street parking is technically public parking and they can't officially complain about it, but they complain the neighbors will only retaliate worse.

Figure 7: Summary of a Reddit posts using TextRank

be generated for a news article focused on one specific topic or a short story focused on one situation, both TF-IDF and TextRank are good choices.

It seems that TextRank has a tendency to favor longer sentences. This could be because longer sentences have more space to include any buzzwords. It can be offset by scaling the similarity score for each sentence by the length, just like the TF-IDF algorithm, although this behavior may be preferred for a more complex corpus.

In the future, abstractive summarization will be explored to determine an algorithm that returns a summary by utilizing natural grammar structures. This will be done with the aim of producing more representative summaries, specifically for *Moby-Dick* and the Reuters articles. And, the usefulness of neural networks, specifically long short-term memory neural networks (recurrent neural network) [14], will be tested to increase the performance. In addition, evaluation methods of text summarization will be investigated to measure the quality of the summaries.

References

[1] T. Rosenstiel, J. Sonderman, K. Loker, M. Tran, T. Tompson, J. Benz, D. Junis, The personal news cycle, *Chicago, The Media Insight Project*.

[2] H. P. Luhn, The automatic creation of literature abstracts, *IBM Journal of research and development*, *2*(2):(1958), 159–165.

[3] S. Bajaj, Shakespeare in one hundred words, *CS224N/Ling284 Final Projects 2013-2014*.

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

[5] J. Beel, B. Gipp, S. Langer, C. Breitinger, Research-paper recommender systems : a literature survey, *International Journal on Digital Libraries*, *17*(4):(2016), 305–338, ISSN 1432-5012, doi:10.1007/s00799-015-0156-0.

[6] L. Page, S. Brin, R. Motwani, T. Winograd, The pagerank citation ranking: Bringing order to the web., Technical report, Stanford InfoLab, 1999.

[7] R. Mihalcea, P. Tarau, Textrank: Bringing order into text, in *Proceedings of the 2004 conference on empirical methods in natural language processing* (2004).

[8] A. Hagberg, P. Swart, D. S Chult, Exploring network structure, dynamics, and function using networkx, Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[9] j. Bryce Boe, et al., PRAW: The python reddit api wrapper, 2012–, [Online; accessed Feb 2nd 2019].

[10] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, P. Blunsom, Teaching machines to read and comprehend, in *Advances in Neural Information Processing Systems*, pages 1693–1701 (2015).

[11] T. Oliphant, NumPy: A guide to NumPy, USA: Trelgol Publishing, 2006–, [Online; accessed Feb 13th 2019].

[12] A. Day, Atuomatic extractive summarization, https://doi.org/10.5281/zenodo.2563845, 2019.

[13] S. Bird, E. Klein, E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit* (" O'Reilly Media, Inc.", 2009).

[14] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, Y. Shi, Spoken language understanding using long short-term memory neural networks, in *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 189–194 (IEEE, 2014).