The Role of Key Phrases in Extractive Graph-Based Multi-Document Text Summarization

Dunia Hakim
Stanford University
Department of Computer Science
dunia@stanford.edu

Abstract

It is widely known that text summarization is extremely useful for various reasons, most notably for time efficiency. While there have been many approaches to automating text summarization, graph-based extractive approaches especially appeal to me due to the easiness with which they can be applied to any sort of text(s). Throughout this paper, I expand and try to improve on a graph-based multi-document extractive test summarization model by exploring how how key phrases can be used in the process of text summarization to produce better summaries. With this goal in mind, I came up with three different approaches (each with two versions) and ran initial tests on the dataset of DUC 2003 with 60 document clusters. I then decided on one version of one approach (with a specific value for the factor that I introduced) that seemed to work best. Finally, I tested this model (of the version, approach, and value that I chose) on previously unseen document clusters for two different summary lengths (one similar to the summary lengths I used for development and another that is significantly different from what I used during development). I found that my model truly did perform better for not only the development dataset but also the previously unseen document clusters (both summary sizes). The code that I used for implementation can all be found at this Github repository. However, the dataset is not available there because I'm not allowed to publicly distribute it.

1 Introduction

Given the amount and diversity of information that is now available to us, it can become often hard to go over enough sources to get a holistic understanding of a topic without some-times having to do an absurd amount of reading and thus spending a significant amount of time doing so. That is exactly why text summarization can be so useful. With text summarization, we could potentially get comprehensive coverage of a topic in a concise and quick-to-read format. That is why, in this paper, I focus on the task of automatic text summarization. More specifically, I focus on multi-document text summarization, which is the summarization of not only one, but multiple documents. The reason for this choice is that I believe it will provide us with a more holistic coverage of the topic at hand.

Additionally, I will be focusing on extractive text summarization, which means that the algorithm extracts whole sentences from the texts to create the summary, rather than abstractive text summarization, which creates its own sentences from words (Dong, 2018). The reason that I focus on extractive techniques rather than abstractive ones is that with the current approaches explored, it seems that extractive approaches still perform better than abstractive ones (Tan et al., 2017). The idea that extractive test summarization depends on is that we can create summaries by simply finding the most important sentences within a document or a cluster of documents, and we can then combine those sentences to create the summary.

In this paper, I propose that using the key phrases of a document cluster will help us better evaluate which sentences are important enough to include in the summary and therefore, create better extractive summaries. The intuition behind this approach is that the key phrases of a document cluster represent the core ideas and topics of the cluster. They can even be thought of as extremely short summaries of the cluster. Therefore, by taking into account those key phrases, and more specifically, by considering the similarity between those key phrases and the various sentences in the cluster, we can better evaluate which sentences are the most important, and then like in all extractive text summarization algorithms, combine those sentences to produce our summary.

2 Literature Review

There are many approaches to text summarization. The most commonly used are either graph-based or neural network-based (Allahyari et al., 2017). While neural networks in general have been receiving a lot of attention for being able to achieve state-of-the-art results at many tasks, when it comes to text summarization, there are still many challenges and flaws associated with using neural networks in text summarization that makes their use limited to the text(s) or summaries having certain properties. For instance, the models currently explored are unable to deal with sequences longer than a few thousand words due to the large memory requirement of these models (Dong, 2018). Additionally, as in all neural networks, their performance is highly dependent on the training data used. This means that the models' performance will highly fluctuate depending on how similar the given document or cluster of documents is to the training data. Since texts can greatly vary in text length, topic, language, style of writing, etc., this means that neural networks will require a humongous amount of data to be sufficient enough to use for summarizing any sort of text. On the other hand, graph-based approaches only depend on the given text(s), which means that those algorithms can be used for any text length, any domain, any language, etc.. I find this potential for generic-use extremely appealing and therefore have decided to expand on graph-based approaches in this paper.

The most well-regarded graph-based approaches are TextRank (Mihalcea and Tarau, 2004) and LexRank (Erkan and Radev, 2004). They both explore a very similar idea which is to represent a document or a cluster of documents as a complete graph in which every sentence is represented by a vertex and every edge represents the similarity between the two vertices (sentences) that it connects. In these approaches, the similarity between sentences/vertices is regarded as a measure of how much one sentence refers to, or ”recommends”, another. The more references, or similar sentences, a certain sentence has, then the more central that sentence is to the overall text(s), and so the higher its ”centrality score” is. Additionally, references from important sentences carry a greater value than references from unimportant sentences. This idea is similar to how the well-known PageRank algorithm, developed by Google, ranks web pages for web searches (Page et al., 1999). At the end, TextRank and LexRank rank all of the sentences using their ”centrality score” by decreasing order and the sentences with the highest scores are combined to create the summary (Erkan and Radev, 2004; Mihalcea and Tarau, 2004).

There are two main differences between TextRank and LexRank. The first main difference is that the paper by Erkan and Radev did not only explore the graph-based summarization algorithm (LexRank), but it also explored a redundancy penalty algorithm called Cross-Sentence Informational Subsumption (CSIS) to add to LexRank so that it can be applied to multi-document summarization rather than only to single-document summarization (2004). Meanwhile, Mihalceea and Tarau only tested TextRank on single-document summarization tasks. The second difference is how they compute the similarity between sentences. In TextRank, the edges are simply the number of words that two sentences have in common normalized by the sentences lengths (Mihalcea and Tarau, 2004). In LexRank, on the other hand, the edges represent a more mathematically complicated relationship between the sentences which uses the cosine similarity between the tf-idf vectors of the sentences (Erkan and Radev, 2004).

Overall, LexRank seems to produce better summaries than TextRank when comparing their results according to the evaluation metrics ROUGE-1, ROUGE-2 and ROUGE-SU4 which have been shown to be highly correlated with human judgments and are usually used as the evaluation metrics for text summarization tasks (Wikipedia contributors, 2019; Mihalcea and Tarau, 2004; Erkan and Radev, 2004). For this reason, my own approach expands on the LexRank algorithm to test my hypothesis about the potential benefit of considering the key phrases during the task of multi-document text summarization.

I could not find any papers that have tried to expand on LexRank by using key phrases from the document or cluster of documents that are being summarized. However, I did find a paper by Zhao, Wu, and Huang that expanded on LexRank to focus the summarization task on a specific query (2009). The introduction of a new factor that was used in that paper to introduce query-focused text summarization to LexRannk inspired how I introduced similarity to key phrases to LexRank in this paper in section 4.2.

3 Data

Since my project does not require learning, I dont actually need large amounts of data. However, for the development of my models (trying different approaches to including similarity to key phrases in the ranking process) and then the final testing of my hypothesis, I will be using the DUC 2003 and the DUC 2004 datasets. The DUC datasets were created by the Document Understanding Conference (DUC) which were a yearly text summarization community that ran conferences relating to text summarization from 2001 to 2007 . Although DUC was substitute for other text summarization conferences (such as text analysis conference (TAC)), the DUC datasets are still commonly used for extractive multi-document text summarization (Zhao et al., 2009; Zhu et al., 2007).

Each of the datasets, DUC 2003 and DUC 2004, contain multiple tasks. I will be using one task from DUC 2003 for development purposes. It includes 60 document cluster, each of which contains 12 documents of average length 450 words. Each document cluster has multiple corresponding model summaries that were created by humans. The model summaries that I have been able to get access to have an average length of 100 words. I will also be using one task from DUC 2004 to test my results at the end after all development has been done. It includes 50 document clusters, each of which contains 10 documents of average length 400 words. Each document cluster has multiple corresponding varying-sizes model summaries that were created by humans; some are short with an average length of 10 words, and others are longer with an average length of 100 words. This will allow me to test my approach and compare its results to model summaries of both similar and different summary lengths than I originally used during development.

4 Models

4.1 Baseline Model

The model that I will be using as a baseline is the summarization system discussed in the LexRank paper (Erkan and Radev, 2004). As I explained in my literature review, LexRank represents the text(s) as a graph where each sentence is represented by a vertex, and each edge represents the lexical similarity between the two sentences that it connects. More specifically, the edges are the cosine similarity between the tf-idf representation of the two sentences (sometimes thresholding is also used to ignore edges below a certain value). Each sentence is then assigned a centrality score by finding its probability under the stationary distribution of a random walk on this graph (Erkan and Radev, 2004). After this first step of computing centrality scores using LexRank, a second step performs re-ranking to avoid redundancy in the highly ranked sentences. Since different sentences may contain similar content, in order to remove redundancy and increase the information coverage in the summary, Erkan and Radev use a greedy algorithm to impose redundancy penalty to the sentences and compute the final overall ranking scores. The details of the exact redundancy penalty algorithm that they use, Cross-Sentence Informational Subsumption (CSIS), were not shared, nor have I been able to find them in other documentation. However, a redundancy penalty algorithm that is commonly used after LexRank is as follows (Zhao et al., 2009):

1. Define two sets $S_1 = \emptyset$, $S_2 = \{s_i | i = 1, 2, ..., N\}$, and initialize the ranking score of each sentence to its centrality score, i.e. $RankScore(s_i) = {p_i}^*, i = 1, 2, ..., N$.

2. Sort the sentences in S_2 by their current ranking scores in descending order.

3. Suppose s_i is the highest ranked sentence in S_2. Move s_i from S_2 to S_1, and re-calculate the ranking scores of the remaining sentences in S_2 by imposing the redundancy penalty as follows: For each sentence $s_j \in S_2$,
$RankScore(s_j) = RankScore(s_j) - A_{ij}{p_i}^*$
where A_{ij} is the cosine similarity between the tf-idf representations of sentence i and sentence j.

4. Repeat step 2 until $S_2 = \emptyset$ or the iteration reaches a predefined maximum summary size.

The third step in the above algorithm is to decrease the ranking scores of less informative sentences by the part conveyed from the most informative one.

The more a sentence is similar to the most informative one, the more penalties it receives and its overall score is decreased.

4.2 My Model

My own approach takes the centrality score computed by LexRank from the first step of the baseline described in 4.1, then it modifies this score to include a different score that represents the sentence's similarity to the key phrases in the document cluster. More concretely, let $S_{centrality}$ be the centrality score computed by LexRank for the given sentence. Let $S_{keyphrase}$ be the score that represents the sentence's similarity to the key phrases in the document cluster. The modified score for the sentence according to which the sentences will be ranked, $S_{modified}$, is then given by: $S_{modified} = d * S_{keyphrase} + (1 - d) * S_{centrality}$ where d is the factor that represents how highly to consider the key phrase score.

In order to compute $S_{keyphrase}$, I first need to extract the key phrases. I used the online multi-lingual key phrase extractor pke to do this (Boudin, 2016). I then experimented with 3 different approaches to computing $S_{keyphrase}$ for a given sentence. For each of the 3 approaches, I tried two version: First using only the key phrases extracted by pke, second using the key phrases extracted by pke along with similar key phrases that can be found using WordNet (Feinerer and Hornik, 2017). Here are descriptions of the 3 approaches:

- **Approach 1:** $S_{keyphrase}$ is equal to the number of key phrases that is present in the given sentence.
- **Approach 2:** $S_{keyphrase}$ is equal to the sum of the cosine similarity between the tf-idf representations of each key phrase and the given sentence.
- **Approach 3:** $S_{keyphrase}$ is equal to the weighted sum of the number of key phrases found in the given sentence where the weight for a certain key phrase will depend on the key phrase's importance. In my own implementation, I was able to quantify this importance using the scores/ranking provided by the pke package for each key phrase.

After computing $S_{modified}$ for each sentence in the document cluster, I rank the sentences in decreasing order by their $S_{modified}$ scores. I then re-rank the sentences using same the redundancy penalty algorithm described earlier in section 4.1 to increase the information coverage in the final summary. Finally, I combine the most highly ranked sentences to create the summary. Note that the number of sentences I use to create the summary depends on the predefined size of the summary set by the user (or specified by the model summary in the case of testing).

5 Testing

5.1 Evaluation Metric

I will be using the ROUGE metric for evaluation (Lin, 2004). ROUGE was the standard evaluation metric throughout the DUC conferences and it usually the standard evaluation metric across text summarization work in general (Wikipedia contributors, 2019). Therefore, I found it most suitable to use ROUGE as the evaluation for my work. There are different variations of ROUGE that are used in text summarization evaluation such as ROUGE-N, ROUGE-W, ROUGE-SU, etc.. I will be using ROUGE-1 which is simply the overlap of 1-gram (each word) between the summary my system produces and the model summary provided in the dataset. This overlap is computed after stemming the words and removing stop words. The reason that I will be using ROUGE-1 is that it has been found to correlate among the best with human judgments (Lin, 2004).

5.2 Experimental Setup

To test my model and hypothesis, I compare the average ROUGE-1 results of my baseline against each version of each approach of my own model on the DUC 2003 dataset (my development dataset). Since there are 3 approaches, and each approach has two versions, then there are 6 possibilities overall.

Remember that the only difference between the baseline and my own approaches is that my approaches compute $S_{keyphrase}$ and modify the centrality score according to the following equation:
$S_{modified} = d * S_{keyphrase} + (1 - d) * S_{centrality}$
Therefore, notice that when $d = 0$, then my approaches are simplified to the baseline.
Also notice that we set the value of d ourselves. Therefore, we need to explore which value of d works best. Therefore, I test each of the versions of each of my approaches when various values of d. More specifically,

I tested each of them with values of $d = 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0$.

5.3 Results

Figure 1, 2, 3, 4, 5, and 6 below show the resulting average ROUGE-1 scores as a function of d for each of the 6 test runs (for the 6 possibilities since there are 3 approaches and 2 versions for each approach). Each run consists of running a single version of a single approach and computing the average ROUGE-1 score for the difference between my produced summary and the model summary for all 60 DUC 2003 document clusters. The titles of the graphs point out the approach used (one of the three approaches described in 4.2) and whether or not synonyms/similar key phrases to the extracted key phrases were used (which of the two versions). The redundancy penalty algorithm discussed in 4.1 was used during all test runs. The resulting graphs were as follows:

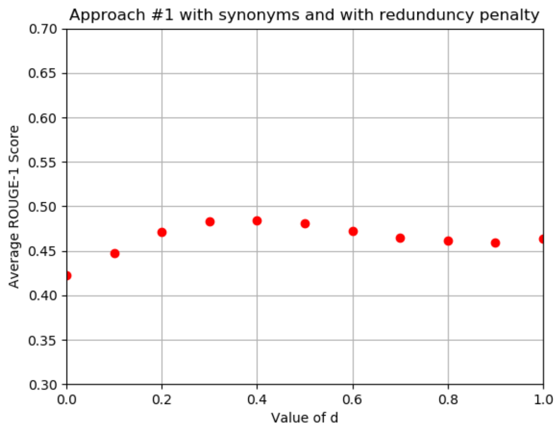

Figure 1: Approach #1 with synonyms

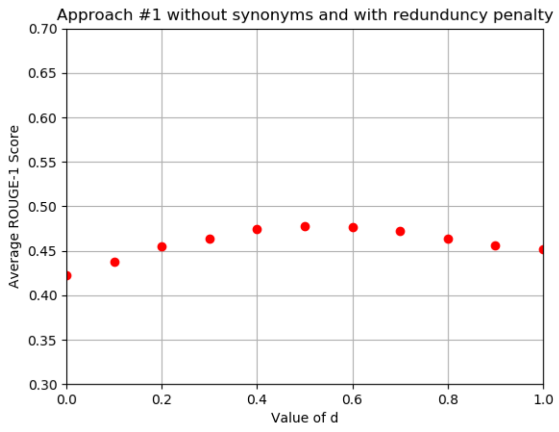

Figure 2: Approach #1 without synonyms

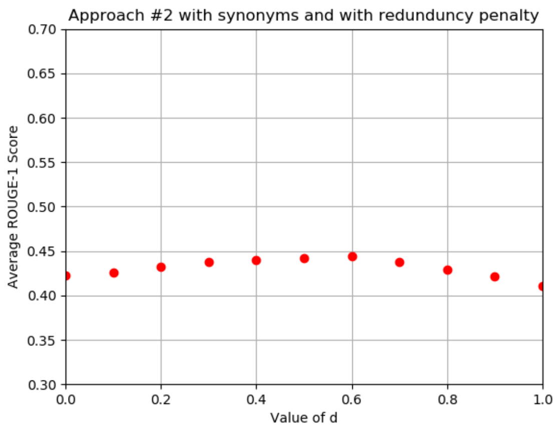

Figure 3: Approach #2 with synonyms

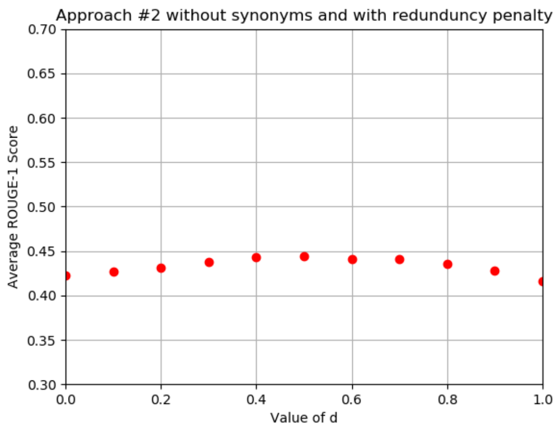

Figure 4: Approach #2 without synonyms

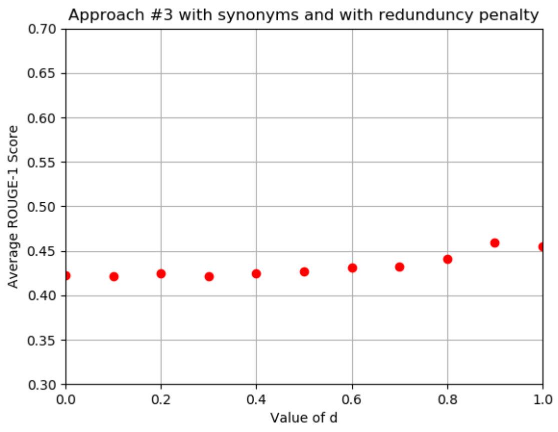

Figure 5: Approach #3 with synonyms

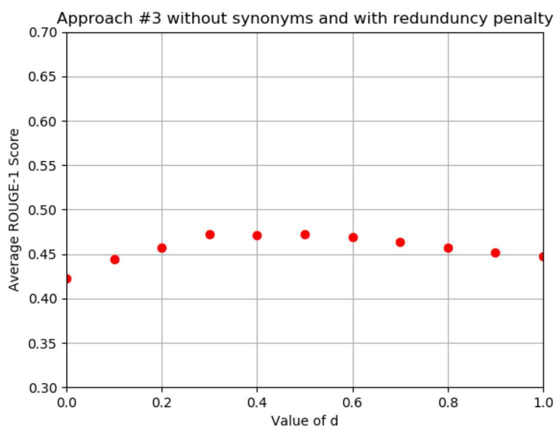

Figure 6: Approach #3 without synonyms

6 Analysis

First notice that almost for all d values that were tested and all versions of all approaches, having a d value other than 0 produces better summaries (according to ROUGE-1) than when $d = 0$ which is the baseline model (using LexRank then the redundancy penalty algorithm). This proves that my hypothesis, that including the similarity between the key phrases and the sentences would produce better summaries, is indeed generally correct. The fact that the produced summaries were improved by taking into account the key phrases of the text(s) through many approaches and not only one specific approach proves that the idea of including the similarity to key phrases in general has merit, regardless of the details of the approach. However, the details of the approach are, of course, necessary to guarantee the maximum improvement possible.

Interestingly, in some cases, such as in both versions of approach #1, using $d = 1$, or only the similarity between key phrases and the sentences without using the similarity among sentences but with the redundancy penalty, produces better summaries (according to ROUGE-1) than when $d = 0$ which is the baseline LexRank + redundancy penalty. This suggests that the similarity between sentences and key phrases could even be a better estimate of sentence importance than the relationship among sentences themselves (the core idea of LexRank).

To get a clearer view of how the approaches and the different versions compare to each other, below is a table, Table 1, showing the best average ROUGE-1 score achieved by each version of each approach and the associated d value that acheived those scores.

Approach	**value of d**	**Average ROUGE-1 Score**
Approach #1 with synonyms	$\mathbf{0.4}$	$\mathbf{48.548\%}$
Approach #1 without synonyms	0.5	47.740%
Approach #2 with synonyms	0.6	44.425%
Approach #2 without synonyms	0.5	44.451%
Approach #3 with synonyms	0.9	45.878%
Approach #3 without synonyms	0.5	47.226%

Table 1: Best Average ROUGE-1 score for each version of each approach

We can see that the best result achieved over all tests is the first approach with synonyms, or in other words, when equating $S_{keyphrase}$ to the number of key phrases, and their similar key phrases according to WordNet, in the given sentence. The average ROUGE-1 score for approach #1 with synonyms is 48.548% and it is achieved with $d = 0.4$.

In order to evaluate how well such an algorithm (using approach #1 with synonyms and $d = 0.4$) would perform on unseen document clusters, I will compare the average ROUGE-1 score of using approach #1 with synonyms and $d = 0.4$ against the average ROUGE-1 score of using the baseline model ($d = 0$) on my test dataset which is the 50 document clusters of DUC 2004. I will use the model summaries with an average length of 100 words for now.

Model	**value of d**	**Average ROUGE-1 Score**
My model: Approach #1 with synonyms	$\mathbf{0.4}$	$\mathbf{58.431\%}$
Baseline Model	0	48.148%

Table 2: Comparing Average ROUGE-1 Score Between Baseline Model And My Model on the DUC 2004 Dataset (summaries with average length of 100 words)

It seems from the results in Table 2, that even when evaluated on a previously unseen dataset, my own model, more specifically approach #1 with synonyms and $d = 0.4$, produces better summaries than simply using LexRank along with the redundancy penalty. These average ROUGE-1 scores were computed when comparing the produced summaries by either model (baseline model or my model) to the model summaries of an average length of 100 words.

Moreover, the Figure 7 below shows the average ROUGE-1 scores as a function of d when using approach #1 with synonyms on the DUC 2004 dataset and model summaries of average length of 100 words.

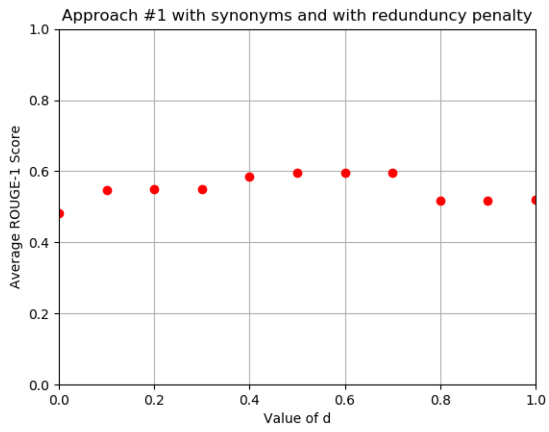

Figure 7: Approach #1 with synonyms on DUC 2004 (model summaries of average length 100 words)

We can see from Figure 7 that the best values for d to use with approach #1 with synonyms on DUC 2004 (model summaries with average length of 100 words) are around the range of $0.4 \leq d \leq 0.7$. Therefore the value of d that we chose based on DUC 2003, $d = 0.4$ comes quite close to be the best value for d based on DUC 2004 for the model summaries with an average length of 100 words.

So far I evaluated my model on an unseen dataset with model summaries of length very similar to that of the development dataset (100 words on average). To see how well my model performs compared to the baseline for significantly shorter model summaries, I will compare the average ROUGE-1 scores between my model's produced summaries and the baseline's produced summaries when compared to model summaries of an average length of 10 words.

From Table 3 we can see that my model produces better summaries than the baseline model

Model	**value of d**	**Average ROUGE-1 Score**
My model: Approach #1 with synonyms	$\mathbf{0.4}$	$\mathbf{29.078\%}$
Baseline Model	0	23.464%

Table 3: Comparing Average ROUGE-1 Score Between Baseline Model And My Model on the DUC 2004 Dataset (summaries with average length of 10 words)

even for shorter summaries, even though I only used longer summaries during the development process of my model. This proves the effectiveness of my model because it seems to work for varying summary lengths, regardless of the summary lengths that I used to develop my approaches and to choose the values of d. As I previously mentioned, this independence from the development/training data is a huge benefit of using graph-based model rather than other models that require learning.

Finally, notice that both the baseline model and my model perform significantly worse for shorter summaries than for longer summaries. However, this makes intuitive sense since every word in shorter summaries makes a significant difference for the final ROUGE-1 score. Whereas for longer summaries, single words do not have as large of a weight in the final ROUGE-1 score.

7 Conclusion

After seeing that my model truly does seem to produce better summaries than simply using LexRank then a redundancy penalty algorithm on the DUC 2003 and DUC 2004 datasets for summaries of length 100 words and 10 words on average, it would be interesting to test the limits of my model even further. For instance, it would be valuable to test my model on document clusters in different languages, or on very technical document clusters that use extremely different formats and terminology. This is especially valuable given that the biggest advantage of working on models that are solely dependent on the given text(s) is that they can be used for any document(s), and are not restricted by a certain genre, domain, language, or length. Seeing that my model performed equally better on varying summary lengths than the baseline model, I would guess that it would perform just as well on any document cluster, however, this should definitely be tested and proved with the proper experiments.

Another valuable experiment that should be done, if the proper labeled dataset is available, is to test how well my model performs at generating two or three word titles for a document clusters. Since titles can be thought of as extremely short summaries, then my model could potentially be used to generate titles and headlines. This would be incredibly useful, and so it would be extremely valuable to run such an experiment to investigate whether this is possible with my model.

Lastly, it would be interesting to see what other models are possible with the same idea of using the similarity between key phrases and sentence to produce better ranking of sentence importance and therefore better final summaries. I only experimented with three approaches in this paper, but there are many more that could be explored. There could possibly be a better approach with a better d value that could be used to produce even better final summaries than my model produced.

Acknowledgments

I would like to thank the instructors and the teaching assistants of CS 224U at Stanford for guiding me throughout my work on this project. I would like to especially thank Atticus Geiger, my project mentor, for the helpful and supportive feedback throughout the quarter, and Professor Christopher Potts for helping me get access to the DUC datasets.

Authorship

I, Dunia, worked on this project alone, and therefore did the work, including the writing and the implementations, myself. However, as previously mentioned, I used the python packages pke and $lexrank$ in my implementations. They can be found at https://github.com/boudinfl/pke and https://github.com/crabcamp/lexrank respectively.

References

Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. 2017. Text summarization techniques: a brief survey. *arXiv preprint arXiv:1707.02268*.

Florian Boudin. 2016. pke: an open source python-based keyphrase extraction toolkit. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 69–73, Osaka, Japan.

Yue Dong. 2018. A survey on neural network-based summarization methods. *arXiv preprint arXiv:1804.04589*.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479.

Ingo Feinerer and Kurt Hornik. 2017. *wordnet: WordNet Interface*. R package version 0.1-14.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1181.

Wikipedia contributors. 2019. Rouge (metric) — Wikipedia, the free encyclopedia. [Online; accessed 9-June-2019].

Lin Zhao, Lide Wu, and Xuanjing Huang. 2009. Using query expansion in graph-based approach for query-focused multi-document summarization. *Information Processing & Management*, 45(1):35–41.

Xiaojin Zhu, Andrew Goldberg, Jurgen Van Gael, and David Andrzejewski. 2007. Improving diversity in ranking using absorbing random walks. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 97–104.