TEXT SUMMARIZATION OF CONGRESSIONAL LEGISLATION

Connor Cabrey[1] and Dalia Habiby[1]
[1]*School of Public Affairs, American University, Washington, DC, 20016*

ABSTRACT

This project aims to address the issue of accessibility concerning Congressional legislation. Using a selection of House Resolutions from the 117th Congress Second Session, we conducted a text summarization task through an extractive approach. We employed a pre-trained summarizer pipeline called LexRank from 'sumy' in python to generate summaries of 0.3 times the size of the bill. The success of this summary generation was evaluated through comparing readability scores and the cosine similarity of the bill text and their summaries. This evaluation revealed that LexRank failed to summarize many of the bills, and only generated shorter summaries for 22 bills. The readability scores of these summaries were similar to those of the original bills, but slightly higher on average. The average cosine similarity between these bills and summaries was moderate to weak. This study concluded that LexRank is a poor summarizing method for Congressional texts.

1. INTRODUCTION

Many bills are over thousands of pages long and contain verbose language that many people do not have the time or bandwidth to decipher, which can lead to slow decision-making and fundamental misunderstandings of the content [1]. Furthermore, the summaries offered on Congress.gov are slow to appear and tend to be vague or very brief.

Creating accurate, automatic summarizations of these bills would not only be beneficial for the general public, but also for government officials who work in busy environments and need to make well-informed decisions about whether to vote in favor of or against a bill. Additionally, citizens will be more aware of what their representatives are voting for, as these summaries will be objective descriptions of the bills, rather than explanations filtered through the media that may have a partisan spin.

2. LITERATURE REVIEW

There is a substantial collection of literature on text summarization for many professional areas, especially in the legal field. Manual drafting of case summaries is a long and grueling process that is slow and expensive. The process relies on a group of "specialized staffs whose task is to summarize cases" [3]. The number of legal documents produced every year seems to be comparable to the number of legislative documents our Congress produces as well, and methods used in this research may lead to similar results.

A study published in the Expert Systems With Applications international journal presents an unsupervised method for extractive multi-document summarization using a centroid approach and sentence embedding representation [7]. The authors utilize a sentence importance score that is a composite of three other scores to create summaries of the Multi-News dataset. After evaluating with cosine similarity and Recall-Oriented Understudy for Gisting Evaluation (ROUGE), they concluded that their approach is comparable to the current cannon of text summarization methods, including supervised deep learning [7].

Automatic summarization of Congressional bills is a relatively new field, and thus there is much room for improvement. In 2019, two researchers from FiscalNote Research created the first corpus for this task, titled "BillSum" [4]. In this study, they applied a "Document Context Model" (DOC) using TF-IDF, a "Summary Language Model" (SUM) using a BERT model, and an ensemble of both models to predict a label they created indicating whether or not it belongs in the summary.

We intend to build upon the aforementioned studies by implementing an unsupervised, centrality scoring approach to summarization of Congressional bills.

3. DATA

The data we are using for this project comes from the govinfo.gov web API [2]. We used this API to download only the House Resolution (HR) Bills from the 117th Congress Second Session that were "enrolled," which means they were passed in identical form by both houses [5]. We chose to only evaluate this subset because the total number of HRs is slightly overwhelming for an experimental analysis such as this one.

In total, there were 4674 files containing different HRs as well as various iterations of each one based on its status in Congress. Our final data set included only 152 of these files, the ones that had been "enrolled."

Since the files were originally in XML format, the first step was extracting the text elements using the ElementTree module of the xml package in python. This included parsing each file and extracting the root element, which identifies the heading of the file. Then, we located all elements that included text relevant to the bill and extracted them into a data frame. Each observation in the data frame consisted of an identifying row number, the raw bill text, and an initialized column to hold the generated summaries.

4. METHODS

4.1. Summarization

For text summarization tasks, there are two main approaches that can be used: Extractive and Abstractive. The Extractive approach uses simple and traditional algorithms that observe the frequency of words within the text and look for sentences that contain those high frequency words to use in the final summary. The Abstractive method attempts to make smaller sentences with the same semantic meaning as the original sentences, generally using deep learning transformers like the BERT method.

This project implements an Extractive approach through the LexRank summarizer from the sumy python library. This summarizer is an unsupervised, graph-based algorithm that uses eigenvector centrality to identify the most important sentences in a text. LexRank is based on the PageRank algorithm from Google that defines the relevance of each site in relation to the search. This method calculates an importance score for each token, or sentence in this case, in a bill. We then use these scores to extract the most important sentences and combine them to create a summary.

The LexRank summarizer has its own rules about what constitutes a sentence, and vectorizes sentences based on those rules. It also removes stop words, punctuation, and unnecessary characters. Therefore, we did not preprocess the raw bill text before applying the summarizer. We decided against lowercasing and stemming or lemmatization to preserve the grammatical accuracy, context, and proper governmental terms in the summary.

For a summary to be useful, it should be shorter than the original text. Therefore, our first attempt to limit the summary length was to cap it at 500 characters by breaking the loop. However, this caused the pipeline to output nothing if the summary was longer than this limit. Therefore, we investigated how to limit the length within LexRank. The summarizer does not take summary length as a parameter, and thus we changed the base definition of the function itself to create summaries of 0.3 times the length of the bill.

4.2. Evaluation

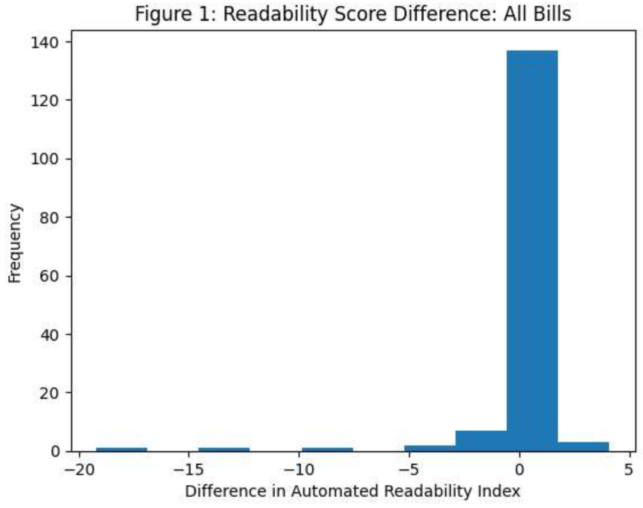
Two of the most popular ways to evaluate the success of text summarization are readability scores and cosine similarity. We began with investigating readability, with the intention of producing summaries that are more readable than the original text. We utilized the textstat library in python to access the Automated Readability Index evaluation and compare the scores for the original bills to the scores for the summaries. For visualization purposes, the graphs display the score of the summary subtracted from the score of its corresponding bill.

The inclusion of cosine similarity was for the purpose of ensuring that the summaries could convey the same ideas as the bill and remain conceptually similar. First, we tokenized using the nltk library and preprocessed the bill text and the summary text. This included lowercasing, stop word removal, and punctuation removal. The next step was to apply a count vectorizer to create a token count matrix of both texts combined. Then, we calculated cosine similarity for each observation through the sci-kit learn library.

5. RESULTS

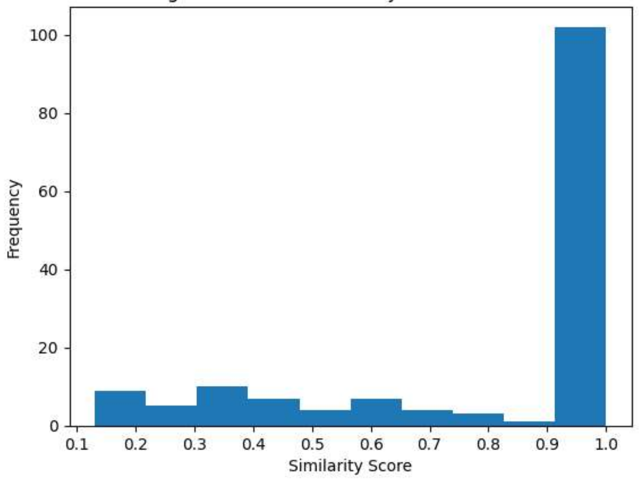
The set of enrolled House Resolutions we analyzed included 152 different bills. In observing the results of the LexRank summaries, we discovered that many of them were exact copies of the original bill, and this mostly occurred for the shorter bills. We found that 113 out of 152 bills had summaries that were shorter than the original text. However, many of these were only shorter by one or two characters. Thus, we extracted only the 22 summaries that were over 100 characters shorter than their corresponding bill.

Figure 1: Readability Score Difference: All Bills



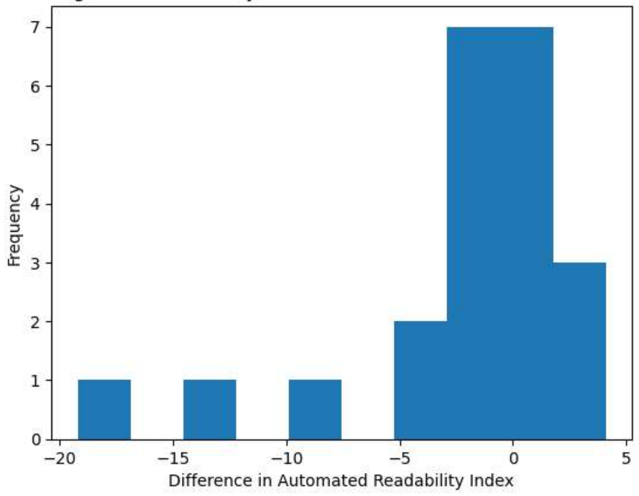
For all 152 bills, the original texts had an average Automated Readability Index score of 31.068, while the average score of the summaries was 31.368. Both scores indicate high difficulty in reading the texts, and the higher score of the summaries indicates that they are harder to read. Figure 1 shows that most of the bills had little difference in Automated Readability Index score, and the ones that had a larger difference indicated that the summaries were more difficult.

Figure 2: Cosine Similarity Scores: All Bills



In terms of cosine similarity, the mean for all the bills is 0.81, indicating a strong similarity between bill and summary. Figure 2 reflects this finding; however, it is because many of the generated summaries are copies of the bill itself.

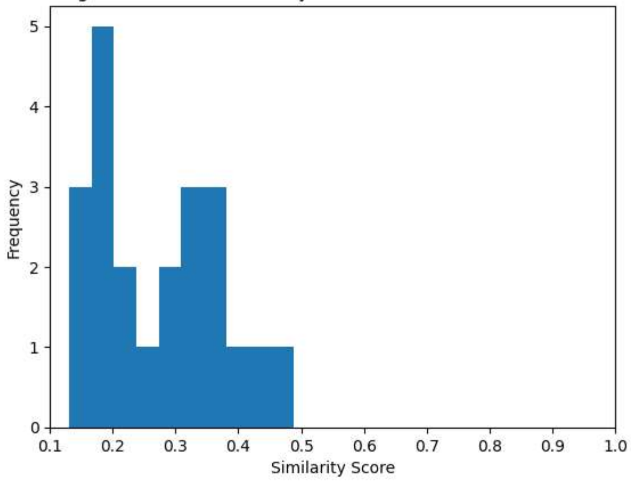
Figure 3: Readability Score Difference: Shorter Summaries



For the 22 bills with notably shorter summaries, the average Automated Readability Index score of the original texts was 28.745, while the average score of the summaries was 30.813. Overall, these are lower than the full corpus, however the summaries are still harder to read than the bills. Figure 3 indicates that when there are differences in readability, the summaries tend to be harder and can be almost 20 points higher than the bills.

When investigating cosine similarity for these 22 bills, the average value is 0.273, indicating a moderately weak similarity. This is striking, as the summaries were extracted directly from the bills. Therefore, the algorithm is missing key information when calculating importance. Figure 4 displays the distribution of cosine similarity scores, ranging from about 0.1 to about 0.5.

Figure 4: Cosine Similarity Scores For Shorter Summaries



6. DISCUSSION

6.1. Conclusions

There are many methods available today in the realm of Natural Language Processing that aim to effectively summarize large bodies of text while retaining as much of the initial corpus' intent and to convey the overall meaning of the text. These methods fall into both the abstractive and extractive groups.

Our project attempted to use an extractive method, LexRank summarizing, which is a graph-based, unsupervised method to summarize US House Resolutions from the 117th Congress 2nd Session. Based on our readability scores comparing the summaries and the original bill texts, we believe that LexRank is a poor method for this particular problem.

Our summaries tended to be nearly as long as, or in some rare cases, longer than the original text. We believe that this may have to do with how the "sentences" are defined by the tokenizer package used in 'sumy', and the underlying grammatical rules defined in the package for determining what constitutes a sentence. There are many non-conventional words and phrases in legal texts, such as "Fed. Reg. 13625" which may lead the summarizer to categorizing abbreviations and titles as separate sentences due to the period.

The summaries that were successfully shorter than the bill tended to have only a weak similarity to the bill from which it was generated in terms of cosine similarity. This indicates that the summaries do not accurately represent all of the ideas found in the original text. For example, the resolution titled "To designate the Department of Veterans Affairs community-based outpatient clinic in French Camp, California, as the 'Richard A. Pittman VA Clinic'" describes the honorable service of Richard A. Pittman in Vietnam and names a veteran's clinic after him. However, the summary only includes the description of his service and does not convey the renaming of the clinic.

Furthermore, the readability score of the 22 summaries was on average, 2.1 points higher on the Automated Readability Index than the bills. This asserts that the summaries are slightly harder to read, likely due to the misclassification of sentences and the lack of contextual relation within this extractive method.

6.2. Future research

One of the best methods for assessing text summarization pipelines is by comparing the resulting summary with human written summaries. Using or creating these human summaries for congressional bills would help in comparing and assessing the results of text summarization methods. Readability is a sufficient starting point, along with cosine similarity, but a comparison to human written summaries would be potentially helpful in assessing model performance.

Another opportunity for further research could be the implementation of other methods we have discussed in the paper, both extractive and abstractive. The implementation of a vectorizer outside of the 'sumy' package such as TFIDF or Word2Vec before applying the extractive method could have differing results that might improve readability and cosine similarity.

The poor performance of this method may suggest that an abstractive approach will be better. One avenue for pursuing this alternative would be to train a deep learning model on human written summaries of legislative texts to ensure that the complexity of these bills is captured.

7. CLOSING REMARKS

7.1. Limitations

The limitations of this analysis include the extremely large available corpus, the complexity of Congressional writing, and the subjective nature of text summarization. There are thousands of Congressional bills that have been written, and thus limiting our data to the second session of the 117th Congress excludes the vast majority of available data.

Furthermore, the unique way that legislation is written was an obstacle in this analysis. There are many instances where an algorithm may incorrectly identify a sentence or fail to classify the meaning of rare phrases. Additionally, text summarization is a nuanced task, and what one person thinks is a good summary may not be what another person thinks, and evaluation methods cannot capture every aspect of a summary.

Another challenge we faced was the file format from the Congressional Bill database. Xml files are similar to html formats with tags and headers breaking up the overall text. Using python to break out the important material from the bill was challenging using the Elementree package. We believe that we may have missed some content and important context from our bills when reading them into python using this package.

8. REFERENCES

1. Bennani, H., & Neuenkirch, M. (2022). Too complex to digest? federal tax bills and their processing in us financial markets.

2. Discover u.s. government information. (n.d.). https://www.govinfo.gov/

3. Kanapala, A., Pal, S., & Pamula, R. (2017). Text summarization from legal documents: A survey. *Artificial Intelligence Review.*

4. Kornilova, A., & Eidelman, V. (2019). Billsum: A corpus for automatic summarization of us legislation. *arXiv preprint arXiv:1910.00523.*

5. Key to versions of printed legislation. U.S. Senate: Key to Versions of Printed Legislation. (n.d.). https://www.senate.gov/legislative/KeytoVersionsofPrintedLegislation.htm

6. Md, Abdul Quadir, Raghav V. Anand, Senthilkumar Mohan, Christy Jackson Joshua, Sabhari S. Girish, Anthra Devarajan, and Celestine Iwendi. (2023). "Data-Driven Analysis of Privacy Policies Using LexRank and KL Summarizer for Environmental Sustainability" *Sustainability* 15, no. 7: 5941. https://doi.org/10.3390/su15075941

7. Salima Lamsiyah, Abdelkader El Mahdaouy, Bernard Espinasse, Saïd El Alaoui Ouatik, (2021). An unsupervised method for extractive multi-document summarization based on centroid approach and sentence embeddings, *Expert Systems with Applications*, Volume 167, 114152, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2020.114152