Summarizing Congressional Bills: Identifying Financial Allocations and Scoring Relevancy

Alex ONeill
Sidenberg School of CSIC
Pace University
New York, NY. USA
ao24521p@pace.edu

***Abstract*—Federal legislation in the United States has been known to be complex, lengthy, and difficult to understand. Voters are expected to be informed when they vote for elected members of government, yet frequently, these elected representatives are asked to vote on lengthy legislation on such short notice, that they can't be informed on the context of the legislation up for a vote. Using regex matching, we have processed legislative text to identify areas of legislation that are concerned with government spending. Additionally, using doc2vec paragraph embeddings, we have scored sections of text based on their similarity to other text within the same document. Using this output, we provided transparency into the content of lengthy legislation and identified areas of the text that may require additional scrutiny by lawmakers.**

***Keywords*— doc2vec, regex, federal legislation, government spending, text similarity, legal text.**

I. INTRODUCTION

For many, government spending is a hot button topic. The spending deficit in the United States has been discussed in the news, in legislative chambers, and among voters for years. With such a high level of interest in how our government spends tax-payers' dollars, it should be expected that our elected officials are knowledgeable and informed of how much money is being spent—and, what the money is being spent on. Late in December of 2020, Congress passed HR 133, also known as the Consolidated Appropriations Act, 2021. This bill was 5,593 pages long and allocated $2.3 trillion for government spending. Of this $2.3 trillion, $900 billion was earmarked for Covid-19 stimulus relief, with the remaining $1.4 trillion allocated for the 2021 fiscal year federal budget. Keeping in mind the length of the bill and how much money was allocated, there are a couple of questions that quickly came to mind. Did anyone read the entire bill before voting on it? Did the representatives know what the money in the bill was being spent on? Because the final version of the bill was sent to representatives *hours* before they were expected to vote on its passing, the answer to both of these questions was, "No". We aimed to address this information gap by identifying sections of the legislative text that were concerned with government spending. By identifying these sections of text, we then hoped to present these in a meaningful and interactive way so that a user could quickly evaluate the context surrounding the financial allocation. The intent was that this would provide greater transparency and increased scrutiny over areas of controversial government spending.

The inability for representatives to be fully informed about the bills which they were voting on is not limited in impact to areas of government spending. Oftentimes, unrelated topics have been lumped together in a single bill, and politicians have even been known to introduce "pork" into bills as a way to push political agendas. A subsequent Covid-19 stimulus bill that was passed in early 2021 received a lot of media attention over the fact that a permanent increase to the minimum wage was initially one of the clauses introduced. Many people believed that this stipulation was unrelated to pandemic relief and was simply dirty politics. Fortunately, this stipulation received a lot of media attention, so it was not unknown, but unknown and unrelated clauses continue to be bundled into lengthy bills. We believed that if we were able to close the information gap on lengthy and unreadable bills by identifying areas of hot topic spending, we could apply similar logic to identifying areas of the text that were unrelated to the rest of the bill. By programmatically identifying areas of the text that may have warranted additional scrutiny, we hoped that lawmakers would be better informed and better able to accurately represent the interest of their constituents.

While both problems we identified were similar in that they required the same overarching solution, the means to developing this solution differed. Identifying areas of text where money was being discussed did not require a complex solution. Using the "Ctrl-F" command to search the text for specific keywords was actually discussed as a manual solution to this problem. Because we intended to provide a meaningful and user-friendly way to evaluate text, we felt it necessary to expand upon, and automate this overly simplistic "Ctrl-F" method. By using keyword and pattern matching, we aimed to identify sentences of interest accurately and efficiently. Identifying sections of text that were unrelated to the rest of the bill was more challenging and subjective. To score pieces of text, we chose to use a cosine similarity evaluation of sequence embeddings in a novel way that enabled us to use similarity scores in a one-to-many approach versus the one-to-one which was commonly used. This novel application of a standard sequence embedding model allowed us to take a more holistic view into how relevant individual sections of a document were

to the overall theme. By combining this approach with our method to identify areas of financial interest, we hoped to provide the user with the information needed to close the gap of being uninformed in two major topics, government spending and transparency.

II. Background

A. AI and Machine Learning in Law

For many, law and legal work is viewed as a form of art that requires a human touch to properly understand the applications of legal text and the judgments that can be made from these texts. In fact, in a ranking of the top ten majors for law-school students, Business, English, and Philosophy scored within the top five. On the other hand, Math and Science ranked ninth and tenth with Science only scoring in the top ten primarily out of interest in practicing specialized law [1]. The paradox of this view is that much of the law corpus is based on judgments from previous court cases. The successful practice of law should require a very strong grasp of logical rules and the ability to understand the relationship structure between various sections of text.

AI and machine learning (ML) applications are typically strongest when applying logical rules and weakest when emulating human creativity. With many viewing the practice of law as a balance between both ends, it brings into question how technology can improve our legal systems and the professional practice of law. Law has always been known to be a text-heavy field, but this is becoming compounded by the number of digital communications and digital footprints that we leave behind. A group of researchers from the Ankura Consulting Group, which has a large group of law consultants, have noted in their research on applying ML and keyword searching methodologies to document reviews that the cost of and volume of document reviews in legal matters continues to rise rapidly [2]. In fact, the dataset which they used for research was from an actual legal matter and involved a corpus of 8.7M documents. A thorough and accurate evaluation of this many documents would be an impossible task for a team of lawyers and assistants to perform manually. Luckily, there has been some success in using technology to improve on this workload. Argarwal [8] evaluated both predictive modeling algorithms and keyword searching methodologies which have been common tools for document reviews. Keyword searching resulted in a large number of false positives, whereas predictive modeling tools failed to identify many of the targeted documents. But, predictive modeling did successfully identify relevant documents that would have been missed otherwise by keyword searching. Combining these methods yields the best results but it speaks to how much more work still needs to be done in this field. Many companies are already trying to capitalize on this opportunity.

Sil, Roy, Bhushan, and Mazumdar [3] performed a survey of the legal field in an attempt to identify various tools and companies who were trying to apply AI and ML to address the challenges of growing law data. They identified over seventeen companies that were actively engaged in providing technology-based legal products. Many of these companies offer case summarization and case similarity products. Both are hot topics in the field, but the techniques for accomplishing these tasks still vary wildly from researcher to researcher and company to company. A review of prior work exploring the similarity of long text and legal documents shows that researchers have used latent semantic analysis (LSA), one-hot vectorization, TF-IDF, Word2Vec, BERT, TF-W2V, deep structured semantic models (DSSM), and convolutional latent semantic models (CLSM) among others [4], [5], [6], [7]. Furthermore, there is not a well-established way to measure how successful these models are. In other uses of natural language processing such as sentiment analysis, a score of *positive* or *negative* is much easier to evaluate than scoring a comparison of one long court case to another. Ultimately, scoring how similar one case is to another case or building a summary of a lengthy judgment is at the discretion of a human interpreter. Being unable to systematically define success adds to the complexity of research in this field, but it also leaves the door wide open for curious and motivated researchers to explore creative methods of solving these challenges without feeling bound to an "accepted standard".

B. Techniques and Groundwork for New Research

1) Comparing the similarity of long text: Anecdotally, comparing the similarity of text is the most popular area of research in the legal field when it comes to novel uses for machine learning and AI. In their exploration of research trends of AI and ML legal applications Sil, Roy, Bhushan, and Mazumdar [3] noted that these techniques have been used fairly effectively to perform language translation and other *human-intelligence* tasks. Modifying these methods to be successful in the identification of relevant case law from one case to another should not be that big of a technological leap. Even small successes in this field would be extremely beneficial to meeting the timeliness standards of the industry.

One of the more recent breakthroughs in research to identify similar court cases was conducted in [7]. In this research, TF-IDF and Word2Vec techniques were combined to maximize the strengths of each method while minimizing their respective pitfalls. This TF-W2V model combined "word frequency and word vector[s], which not only consider[ed] the different influence of keywords and common words on similarity calculation, but also consider[ed] the influence of words with different synonyms [7]". Since legal text frequently repeats phrases to ensure that the relationships between sections are maintained, the accuracy of certain models can be negatively impacted and improperly assign *weights* to certain sections. By using this TF-W2V combination model, [7] was able to maintain a balance between semantic similarities and word-frequency weightings.

An argument against Word2Vec is that is it "only considers the local information of words, not considering the order of the words, also not taking into account the global words based on statistical information [6]". To close this gap, [6] chose to use BERT to evaluate the similarity of large text documents. The reasoning behind this is that researchers believe that BERT "can

better capture the relationship between words, thus making the representation of sequences with contextual information [6]". In short, a key component of BERT is a bi-directional technique where words are masked, then predicted based on the surrounding text, then re-evaluated. The second key component is a *next sentence prediction* procedure. By combining these two components, it has been found that BERT maintains better awareness of the document context. This is something that can be critically important during reviews of legal text.

While there are challenges in applying AI and ML methods to legal text similarity analyses, that does not mean that similar research in other areas cannot be drawn from. For example, Qurashi, Holmes, and Johnson [5] worked to apply text-similarity analysis techniques to rules and regulations relating to railway safety. As the list of regulations and regulatory agencies with overlapping responsibilities grows, the risk that conflicting regulations are approved by different (or the same) agencies grows. A text similarity program would help pull relevant documents into scope allowing for a more detailed examination. They used Word2Vec procedures to conduct this work foregoing the advanced TF-W2V used in [7] but were still able to produce promising results. By enabling a system that allows users to conduct a *consistency check*, not only does efficiency increase but so does the quality of regulations.

2) Summarizing long and legal text: The second area of text analysis that is being conducted in the legal field and others is the study of programmatically summarizing text. In the paper by Argarwal [8], various methods of text summarization are described and compared. Argarwal also offers up some summaries of their own by defining three key areas of text summarization techniques. These three areas were generally defined as: extraction-based summarization - an automated summarization technique where the content objects are extracted as is; abstraction-based summarization – an automated technique revolving around paraphrasing; and, aided summarization – a combination between automated summarization and human-dictated tuning. When it comes to scoring the successes of these techniques, a similar challenge to scoring similarity is found. There is a dependency on human interpretation which can result in a certain level of ambiguity in the scoring systems. ROUGE scores were used to compare different models, but ROUGE scores are still limited in the fact that this metric compares programmatic summaries to reference sets produced by humans [8].

An additional study by Merchant and Pande [4] explored using Latent Semantic Analysis (LSA) to produce summaries of legal text. As described "LSA assumes that semantically similar terms will occur in similar pieces of text. [LSA] analyzes the relationship between a set of terms and documents containing them." Ultimately, the summary results in this research were produced by picking the top-scoring sentences to construct a summary from. The length was limited to five percent of the original document length. Scoring was done via ROGUE scores, but as with the work in [8], similar downsides of using ROGUE scoring were expressed.

One key difference between the bodies of work to summarize text when compared to the research of comparing the similarity of documents is that in this case, the work is done intra-document versus comparing across different texts. This is an important consideration when reviewing the methodology choices in previous and future research opportunities as it changes the scope and how much data is available to be ingested by the model.

3) Identifying the relationships between texts: One novel approach to legal text exploration was conducted by Opila and Pelech-Pilichowski [9]. In this study, the goal was not to look at contextual similarities or to summarize long texts, but instead, the intent was to identify and visualize the logical relationships within and between texts. Within legal text, a single document may refer to other sections or definitions within the text and use these to establish relationships between areas. Imagine one section where facts *x* and *y* are established as evidence and therefore produce a product of *a*, a subsequent section referring to product *a* will have an intrinsic dependency on *x* and *y* being previously established. If section one cannot be established, section two is therefore not valid. [9] cleverly produced visualizations to depict these types of relationships. This is an exciting area of research and could be coupled with some of the other active research in this field to create incredibly impactful tools for the legal industry. Quick reference tools would have the potential to be very accurate and efficient.

4) Keyword and expression matching: A final area of relevant research for our work falls within the realm of keyword and regular expression (regex) matching. For many in the technology field, regex methodologies are commonly known and used everywhere for search tools, SQL queries, application code, and in many areas where human input is accepted. Atasu [10] conducted a study to evaluate resource efficiency for regex analytics with an understanding that our data collection is starting to push the limits of our standard technology solutions. Although regex is commonly accepted, the onslaught of big, unstructured, text data has instilled the need for continual improvement and research. As the number and length of legal documents continue to grow, regex will inevitably be closely tied to research previously mentioned. Once tools that can capitalize on similarity, summarization, and relationship analysis amongst legal texts hit the mainstream market, regex methodologies will need to be put in place to connect the dots between user and algorithm.

Some might say that keyword matching is an over-simplification of regex, but for many, that is all it is ever used for. The measurable importance of this statement is evidenced by the research of Gronvall and Huber-Fliflet [2]. Keyword searching is one of the primary tools used by legal teams to identify relevant or privileged documentation. Even though it is a simplistic approach and can generate huge numbers of false positives, there is still a place for keyword and regex methodologies in the future of AI and ML applications in legal practice. In fact, the conclusion section of [2] research states

that "the legal community can be confident in their use of keyword searching," and even recommends combining keyword searching with other advanced and predictive algorithms to get the best of each method.

III. Methodology

A. Data Collection

Much of the labor in this project was spent collecting and processing the raw data so that it would be usable for our actual analysis. This raw data was scraped from the endpoints which are available from www.govinfo.gov/bulkdata. This site serves as a government-sponsored bulk data repository for certain collections of text ranging from congressional bills, bill summaries, rules manuals, and Supreme Court decisions. The documents listed in this repository are machine-readable friendly as all endpoints are in XML format.

To scrape these documents, multiple python scripts were written primarily utilizing the "requests" and "BeautifulSoup" libraries. The first batch of web-scraping was limited to congressional sessions 116 and 117 which covered the years 2019-2022. During the first batch, endpoints were not fully scraped. This scraping was conducted at directory levels just before the document endpoint. This allowed us to collect and store information about the endpoints without fully navigating to them. This was a deliberate decision to avoiding hitting the webserver with more than 36,000 requests—one for each endpoint. The information from this portion of scraping was stored in a local PostgreSQL database.

For endpoint data collection, a random selection of 1000 bill summaries was chosen from the bill information tables that were created during the initial web-scraping batch. The endpoint URLs from these 1000 rows were then navigated to, parsed into relevant sections of text, then loaded back into the PostgreSQL database. Secondary to this, the URLs for the full-text versions of the corresponding 1000 summaries were also requested, parsed, and loaded into the database. Because many full-text versions of the bills existed in different committees and chambers, there was a 28% increase in the number of endpoints that needed to be navigated to compared to the number of summary endpoints.

When viewing and inserting the raw XML information into the database, there were some issues with formatting and encoding. In some cases, regex substitution was required to remove lingering XML tags. Other errant encoding-specific characters that stemmed from certain character formatting were left in place and removed during later processing. As the full text of the bill was saved into the database, each section of text (as parsed), was stored with a key:value pairing representation so that individual lines of text could be separated at a later time. Storing the bills in this format was done as a means of tagging sections with a unique identifier. These tags could then be used later to identify lines that matched in our financial matching processes and to pair lines of text with their corresponding similarity scores. Although the data collection process comprised of multiple python scripts and database tables, the usable data was fairly simple and unassuming. The remaining columns include names such as "bill title", "bill summary", "session", "row number", and "text". Ultimately, 1271 full-text documents were collected comprising of 989 different bills which were split into 94,312 individual rows of text.

B. Identifying Financial Allocations

The first problem that we attempted to address was identifying where money was being introduced, allocated, or discussed in the legislation. This was conducted by searching for matching phrases in the lines of text. All individual rows of text were selected from the database and streamed through a python script looking for matches.

1) Searching for "$": The first approach was fairly straightforward. As rows of text streamed through our python script, we searched for instances of a dollar sign ($). If a "$" was found in the text, we were confident that money was a key factor. Lines with matches to a "$" were then tagged and loaded into a new PostgreSQL table. This resulted in 5902 matching rows.

2) Phrase matching: If no "$" was found in the text, the text was then processed through phrase matching. By reading through random selections of text that did not include a "$", it was identified that the phrases, "percent of fund" and "not more than <wildcard> percent" were most likely to be included in the remaining rows of text that related to spending. Similar to the "$" matching process, if a row of text matched on these phrases, it was tagged and loaded into the database. This technique resulted in 168 matches.

3) Keyword matching: For the remaining rows of text, we identified keywords that would likely be found in legislation related to spending. This method appeared to be the least accurate and is why it was chosen as the last matching strategy. The keywords that were chosen for matching were, "fund", "funds", "funded", and "dollar". Of the remaining unmatched rows of text, 3250 matched in this section.

All in all, these three matching techniques resulted in 9320 rows of text being flagged as having a possible financial component. This is just under 10% of the 94,312 rows that were selected during the web-scraping process.

C. Scoring Relevancy of Sections

Scoring relevancy of sections proved to be much more complex than identifying areas of financial terminology. To get the process up and running initially, it was decided to use doc2vec to vectorize lines of text, then use cosine similarity to evaluate the relationships between lines of text within the same document. This initial push provided for a baseline and proof of concept that can be expanded upon and tuned in the future. There were two main tasks associated with this portion of our research. Data needed to be cleaned and prepared for doc2vec processing, then modeled and compared.

1) Cleaning and preprocessing: As discussed earlier in the web-scraping section, our dataset included lines of text from different versions of the full-text documents, but from the same

over-arching bill. As legislation moved around chambers, it is possible that small adjustments were made to this text. This complicated our process as we risked introducing duplicate sections of text to our model. To control for this, the SELECT query that was used against the table of text rows was adjusted to select distinct rows when duplicates of the combination of bill title and row number were found. In some cases, this resulted in all lines of text being from the same version of the bill, where other bills included a combination of rows from various versions. We believe that any variance that this method could potentially introduce would be more beneficial than harmful.

Once lines of data were received from the database, each line was streamed through a standardization function that cleaned the text. Every sentence was split into single words then forced into lower case. Each word was then encoded in ASCII format and decoded back in UTF-8 to remove any errant encodings for unrecognized symbols. The words were then processed further to remove all numbers and any non-a-z characters or symbols. Lastly, the individual words were joined back together in a single string, then re-split on whitespace to remove any additional spaces that might have been added in the previous two steps. The individual lines of text (along with other line information) were passed back to the script to be combined based on the bill title.

A document class was created that would handle document objects and the corresponding rows of text, title information, and summary information. Each line of cleaned text and the corresponding row number information was passed into a corpus function. If a document object already existed for the parent legislation of the line of text being processed, the document object would be appended with the new line of text. If an object did not exist yet, a new document object would be created to handle this new line and any subsequent lines of text that matched the bill title. The final result of this function was a list of document objects that held the bill title, summary, and a list of all rows of text for each bill. Doc2vec functions by vectorizing each section of text within a larger corpus. In our case, each document object (bill) that was created served as a single corpus with the individual lines of text serving as the documents within the corpus.

2) Modeling the documents: For each document to be modeled with doc2vec, documents needed to be tokenized and tagged. To perform this administrative task, the bill was read into a python script as a full data corpus. The individual rows of text within the corpus were passed through a final pre-processing utility that once again forced all text to lower case and split by white space into a list. This list was then combined with an id tag for handling by the model. This corpus of processed and tagged documents was then used to train the doc2vec model.

Because many of the bills were smaller in nature, we selected a single lengthier bill to be used as our proof of concept. This bill has just over 10,000 individual lines of text. Knowing that this was still a *smaller* corpus than usually used for doc2vec processing, a test of various model parameters was conducted. This training loop ran our corpus through epoch sizes ranging from 40 – 120 and document vector sizes ranging from 24 – 60. To score the performance of these sample tests, every row of text was fed back into the model to look for the *most similar* row of text. Every row of text in the document was ranked based on similarity to the *test row*. Since a replica of the *test row* was in our document, we would expect that the *most similar* row of text would be itself. To score our model tests, we calculated how often the *test row* was listed in the top five, top 10, and top 20 *most similar* rows of the full text. Depending on scale and interpretation, this test could be viewed as a way of seeking out an overfit model vs. a good model. Due to our small sample size and our understanding of the data, we believed that this was a viable way to test what model parameters would perform best.

After this testing was completed, it was decided that the doc2vec model would be created with a vector size of 24, a minimum vocabulary size of five, and 100 epochs. Additionally, the alpha was restricted to an initial of 0.025 with a minimum of 0.020. The minimum vocabulary size was selected to remove all instances of less frequently occurring words and it is worth noting that all words with less than three characters were also dropped.

Upon completion of model training, a scoring method was conducted similarly to the model parameter testing design. Each line of text was fed back into the model and the similarity scores of every other line were calculated. To score lines in a way that we could compare against each other, the count of positive and negative scores was tallied for each test line. The logic behind this approach was that if a sample line of text was found with 80% of the document scoring with negative similarity, this line of text would be *less similar* to the rest of the document as opposed to a test line where only 30% of the document had negative similarity scores. Since each line was tagged with a row number and a count of positive and negative scores, we could easily inspect the lines of text with scores on opposite ends of the spectrum to better understand the context behind each line.

D. Visualization and Drill Down

The primary problem that we attempted to solve was that our lawmakers did not have any reasonable way to gain transparency or insights into some of the lengthy bills that they were expected to vote on. As interesting as it may be to test random sentences against the legislative text to see what the most similar sentences might be, if our results were unable to be used in a functional way to address our problem statement, the project would not have been a success. By taking the output of our processing methods and loading them back into our PostgreSQL database, we were able to leverage the python Dash and Plotly libraries to build a dashboard that would help users interact with our results. In a similar theme to our previous methods, our functional build-out was done in two parts within the same dashboard.

1) Viewing text with spending conditions: As discussed in the “Identifying Financial Allocations” section, we were able to identify lines of text that discussed government funding without

much problem. In many cases though, viewing these individual lines of text did not provide enough context for someone to decide whether or not the financial allocation aligned with its purpose. To address this concern, we leveraged callback functions to display the text before and after lines of interest.

Within the dashboard, a table was displayed that included all lines of text which matched a specific financial allocation matching criteria. If there was a particular line of interest to the user (ex: $4b for coffee machines) they could then click on that line item and see the preceding and following lines of text from the legislation. This allowed the user to gain more context into why so much money was being spent on coffee machines.

2) Viewing unrelated areas of legislation: For visualizing the consistency of the bill as a whole, a back-to-back bar chart was produced that displayed the negative and positive score counts for each line of text. This offered a clearer view into what proportion of the legislative text was similar to the overall theme and an opportunity to visualize how wide the range of similar scores was on a per-line basis. Similar to the financial allocations charts, this information alone was not overly useful without a drill-down function.

A secondary table was created with the same callback logic as before. As users filtered through and explored the positive/negative score disposition chart, selecting a particular line of interest would populate the secondary table with contextual information. The default number of surrounding lines that we chose to display was five lines before the line of interest, and three lines after.

IV. Results

Evaluating the results of our research was not typical in the sense of using measurement scores aimed at accuracy, precision, and recall. The primary problem that we aimed to solve was best described as a gap in the tooling and functional solutions with regards to how a user can evaluate and interpret the text. We did not aim to interpret the text for the user but instead hoped to guide the user to areas that *may* warrant further evaluation. It is not possible for a lawmaker to read 5000+ pages of legislative text within the two-hour time window before voting on whether the bill should become law or not. By highlighting areas of the text that divert from the overarching theme and by bringing all mentions of money to the forefront, a lawmaker would be better equipped to home in on key areas of contention. We believe that we were successful in accomplishing this objective.

Identifying areas of the text that discuss spending plans was not a complex task. There is no reason that a lawmaker cannot use the "Ctrl-F" command to search for keywords or "$" symbols in the text manually. Our success comes from the fact that we were able to automate this task and present it in a way that is more efficient and user-friendly. Unfortunately, there is no way to numerically measure how effective our results were. Part of our method to identify keywords and phrases related to spending was done by reading random lines of text that were *not* matching on any of our criteria. By reading through the text like this, we found that this was the best way to identify if lines of interest were being missed from our model. In general, we did not find that text related to spending was being missed. To validate that our matching engines were not over-matching, we randomly read through lines of matching text. While there were texts that matched our filters that we felt were unrelated to spending, this seemed to be a rare occurrence. As with identifying text that was missed, there was no proper way to systematically measure our accuracy in identifying text that was over-matched. Furthermore, there is a small amount of variation in how users can interpret whether the text is related to government spending or not.

Similar to our difficulty in measuring how accurate we were in identifying sections of text related to spending, it was also challenging to measure how accurate our similarity scores were. Classifying a document as being similar or not is generally subjective and open to the interpretation of the reader. Classifying the similarity of one word to another is somewhat subjective, comparing a full sentence to another sentence is even more subjective, and ultimately, comparing one sentence to the entire theme of a document is subjective even more. The bulk of our measurement of model strength was done by comparing the most and least similar texts to random prompts. A subjective decision was then made as to whether we felt the model was performing well or not. One numeric measure that we did utilize as an additional indicator was the percentage of rows of text that listed the "most similar" text as itself. Essentially, we pursued a test score that would induce overfitting for the most similar vector. For example, when comparing the entire corpus against document *x*, document *x* was not removed from the corpus. As document *x* remained unchanged, we then expected that document *x* would be the "most similar" document within the corpus. Due to how the text vectors were inferred against the trained model, this was not always the case. But, as the model became stronger, the higher the percentage of documents matching against themselves should be. To calculate the success of this method, we scored what percentage of lines had a *most similar* match to themself. We found that using a vector size of 24 with 100 epochs, 91% of the measured lines had themselves as one of the top five *most similar* lines.

```
Model: Doc2Vec(dm/m,d24,n5,w5,mc5,s0.001,t3)
Top 5 Count: 9497/10367: 91.60798688145076%
Top 10 Count: 9756/10367: 94.10629883283495%
Top 20 Count: 10011/10367: 96.56602681585801%
Epochs: 100 Vector-Size: 24
```

For the purpose of scoring similarity, our method of tallying positive and negative similarity counts is not only intuitive, but easy to interpret. In our test document of 10373 lines of text, only 14 lines of text had a negative count that was greater than 50% of the document. While the full strength of this model is truly subjective, a look into the worst-performing lines of text is somewhat telling—even with the worst scoring lines being short phrases and/or single words. The bill we explored as a sample was titled, "National Defense Authorization Act for Fiscal Year 2020." The two lines of text with the highest number of negative scores were "torture", and "scandium."

Some of the other lines of text in the top 10 worst scoring were "self-propelled", "classical languages", and "street sweeping". While there may be a perfectly good reason for "classical languages" and "street sweeping" to be mentioned in a bill regarding Department of Defense programs and activities, we believe it might be worth asking "why?" and doing some investigation into the context surrounding these sections of text. Because of this, we believe that our results were meaningful, and our analysis was successful.

V. Future work

It should be a common belief that good stewards of society should take an interest in improving upon the work of others who have tried to tackle difficult problems. Politics and government are no strangers to these problems. We aimed to take a novel approach to solve the problem of uninformed voting among our elected representatives. Even though the scientific methods we employed were not overly complex, we hope that they planted a seed that can continue to be researched upon to become more accurate and usable for the end-user.

A. Identifying Financial Allocations

Within our research, the methods we employed to identify financial allocations in the text were iterative. Future work in this area should continue this iterative process. Text that we incorrectly identified as a match should be collected and explored more in-depth so that the pattern which hits on this text can be tuned further. Additionally, text that was not identified as a match should continue to be evaluated to check if other patterns should be included in the matching criteria. We also would advise that future work explores whether there are significant differences among pieces of legislation. It is possible that phrases used in one bill will be worded differently in legislation written by different authors.

An additional area of future work that should be enhanced upon is the useability of the matching results. Matches that have a specific dollar amount tied to them can be processed further so they can be sorted by the amount and combined to calculate total spend. An intelligent way to calculate dollar amounts based on phrases such as "no more than 5 percent" should also be explored. Lastly, an area of future work that would enhance the usability even further would be the creation of a method to match dollar amounts with the recipient of the funds. This was one of the goals that were originally discussed during the onset of our research, but it was not pursued further at the time.

B. Scoring Similarity

There is a lot of room for future work on this section of our research. It was found that even the longest legislative texts were comparatively small to the standard size of the training corpus typically used for word and sequence embedding. HR 133 at 5,593 pages was one of the longest pieces of single legislation ever voted on, but this pales in comparison to some of the more commonly used Wikipedia and Google News datasets which contain multiple billions of words. There is a multitude of different routes to address the size of our training data that can be explored in future research, so there are a lot of opportunities to test different methods. Doc2vec can be substituted with other tools, parameters can be adjusted to produce better results with the current dataset, different models can be pipelined into a combined model, and there may be creative ways to pre-train the doc2vec model on other legislative text. We explored the idea of using all legislative texts for training but believe that this would poison the validity of scoring relevancy within the scope of a single bill. With that, we do believe that there may be creative approaches to solving this problem by training an initial model and then re-training with a single bill that carries increased weight.

Although we chose to use doc2vec and sequence embeddings to score similarity, we believe there are opportunities to explore completely different approaches to solving the same problem. One area of future work that we would recommend being explored is in using anomaly detection methods. While we focused on leveraging semantic context and the comparison of different vectors to one another, unsupervised clustering and anomaly detection might be more successful in identifying outliers in the text. The primary problem that we aimed to solve should be important to many, so all attempts to address this issue should be applauded equally. Whether future research continues to build upon the sequence embedding method that we introduced or takes a different path, we hope that the importance of continuous improvement in this area remains high.

VI. Conclusion

There are new uses for NLP and text processing uncovered daily, many of which are related to conducting a similarity analysis across different documents. We took a novel approach to these processes in an attempt to better equip our lawmakers. The text of federal legislation can be incredibly long, and too often, our lawmakers are forced to vote on legislation before they can fully read and understand the document. As citizens, we have a responsibility to be informed voters, yet our legislative process at times can prevent our lawmakers from being informed about the very bills they are voting on. HR 133, the Consolidated Appropriations Act, 2021 was 5,593 pages in length. The final version of the bill was sent to lawmakers hours before a vote was cast. This should be concerning. We set out to address the inability to thoroughly review the text by programmatically extracting and analyzing the text in legislative documents. There were two main areas that we thought were most important to evaluate: 1) areas of the text where government spending was involved, and 2) identifying sections of the text that were unrelated to the theme of the bill. With the help of regular expression pattern matching and text parsing, we were able to identify sentences related to spending. To compare the similarity of sections of text to the overall theme, we used sequence embedding and cosine similarity measurements made possible by the gensim doc2vec python library. Our results were then consolidated and presented in a user-friendly dashboard.

Although we were not able to effectively complete all of our objectives, we do believe we made a significant contribution to the community by providing a proof of concept that can be

expanded upon. Scanning for financial allocations in the text should continue to remain an iterative process. By adjusting the matching patterns as new information is found, the accuracy of the matching engines will continue to improve. Furthermore, a deeper analysis into how to pair the recipients of government funding to the value that is allocated would be a significant improvement to the project. The scope of improvements that can be made to our scoring of relevancy is widespread. With a large number of semantic analysis algorithms available, we hope that future researchers will continue driving forward with new and creative ways to improve upon our work. As voters, we expect our elected representatives to vote on the behalf of our best interests. Unfortunately, we cannot adequately hold our representatives accountable if they are uninformed of the laws on which they are voting on. It is unlikely that legislative text will be limited in length, and the compressed timelines for voting will likely stay compressed. If these remain unchanged, the next best solution to ensuring our representatives are informed is to provide them with the tools to properly analyze the laws they are voting on.

REFERENCES

[1] Collegeconsensus.com, “10 best degrees for getting into law school,” *College Consensus: School Rankings & Student Reviews Aggregator*, 29-Dec-2020. [Online]. Available: https://www.collegeconsensus.com/features/best-degrees-for-law-school. [Accessed: 19-Feb-2021].

[2] P. Gronvall, N. Huber-Fliflet, J. Zhang, R. Keeling, R. Neary and H. Zhao, "An empirical study of the application of machine learning and keyword terms methodologies to privilege-document review projects in legal matters," *2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 2018, pp. 3282-3291.*

[3] R. Sil, A. Roy, B. Bhushan and A. K. Mazumdar, "Artificial intelligence and machine learning based legal application: the state-of-the-art and future research trends," *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 2019, pp. 57-62.*

[4] K. Merchant and Y. Pande, "NLP based latent semantic analysis for legal text summarization," *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Bangalore, 2018, pp. 1803-1807.*

[5] A. W. Qurashi, V. Holmes and A. P. Johnson, "Document processing: methods for semantic text similarity analysis," *2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA), Novi Sad, Serbia, 2020, pp. 1-6.*

[6] G. Wang, T. Zhang, G. Xu, Y. Zheng, Z. Du and Q. Long, "A deep learning based method to measure the similarity of long text," *2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE), Dalian, China, 2020, pp. 173-178.*

[7] F. Yang, J. Chen, Y. Huang and C. Li, "Court similar case recommendation model based on word embedding and word frequency," *2020 12th International Conference on Advanced Computational Intelligence (ICACI), Dali, China, 2020, pp. 165-170.*

[8] K. Agrawal, "Legal case summarization: an application for text summarization," *2020 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2020, pp. 1-6.*

[9] J. Opiła and T. Pełech-Pilichowski, "Visual analysis of similarity and relationships between legal texts," *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO), Opatija, Croatia, 2020, pp. 1482-1487.*

[10] K. Atasu, "Resource-efficient regular expression matching architecture for text analytics," *2014 IEEE 25th International Conference on Application-Specific Systems, Architectures and Processors, Zurich, Switzerland, 2014, pp. 1-8.*