Supreme Court Oral Arguments: Summarization and Simplification

Developing and Evaluating Models to Improve Legal Document Accessibility

COS401: Final Project Report

John Van Horn, James Zhang, Gillian Rosenberg (JJG)

Princeton University

1 | Introduction

A crucial pillar of American democracy is the presence of informed citizens that trust the political process. This trust is especially important for the judicial branch of government because citizens must agree to the social contract for the system to be effective in ensuring order and justice. Agreeing to the social contract means that citizens believe that their courts are fair and impartial. In order to develop and strengthen these sentiments, people need to be informed and educated on the processes and capacities of the system. In October 2006, the Supreme Court decided to release the transcripts of oral arguments (*Transcripts and Recordings of Oral Arguments - Supreme Court of the United States*, n.d.). Oral arguments are the part of the legal process in which lawyers have the opportunity to verbally present their case to the justices and for the justices to ask questions (*Oral Arguments - Supreme Court of the United States*, n.d.). The Supreme Court hears around 70 to 80 oral arguments per year (*Oral Arguments - Supreme Court of the United States*, n.d.). Clearly, this step in the legal process is an integral part of the judicial process for both lawyers and justices. Yet, it is also crucial that citizens have access to these oral arguments. Stephanie Wylie, a council member in the Brennan Center's Justice Program, points out that these documents are capable of offering "valuable information about the justices' reasoning behind decisions, court norms, and the judicial process as a whole. The lack of access to the court restricts public understanding of the issues it deliberates—and who is allowed to shape that understanding" (Wylie, 2022). In just a span of six months from May to November 2020, more than 2 million people listened to oral arguments from the May 2020 argument session at the Supreme Court (Wylie, 2022).

While it is true that access to the transcripts has allowed many people to be more informed about the legal proceedings of the courts, limitations and challenges still persist. Currently, the most common way of accessing the transcripts are through governmental websites. Because these sites only feature raw, unfiltered documents that model what exactly occurred in the court, transcripts are often filled with complicated jargon and can be long, containing anywhere from 60 to 80 pages. Oral arguments kept in these formats undoubtedly make it challenging for citizens to sift through cases and learn about them.

Our project seeks to address these challenges and improve accessibility of legal court documents by examining and evaluating modern natural language processing (NLP) and linguistics methods for automatic summarization and simplification of Supreme Court oral arguments. Specifically, we strive to find the best methods for summarizing and simplifying text to the American middle school level. We pick middle school students as the target audience because we believe that this is the youngest audience that potentially are interacting with these texts. We believe that individuals need not to wait until they are 18 and able to vote to start the process of becoming an informed citizen who understands governmental systems.

We explore three different methods for summarization. As we explain more thoroughly in the methods section below, we take a hybrid (extractive and abstractive summarization) approach. These methods differ amongst their techniques for extractive summarization – we use the same transformer model for abstractive summarization. After obtaining the generated summaries, we noticed that they were still difficult for a middle school student to understand. We then tested six different methods for lexical simplification and proposed one rule-based syntactic simplification. We evaluate the output of our methods based on scoring mechanisms that are well-known for critiquing text readability. We hope that our findings and approaches can be applied to other legal and governmental documents in order to improve upon our ultimate goal of informing and educating American citizens.

2 | Background and Literature Review

2.1 | Automatic Text Summarization

Broadly, the task of automatic text summarization aims to shorten the length of input text while preserving as much salient information as possible. Because many of the textual sources available today for analysis are rather long, and thus unwieldy, automatic text summarization is greatly applicable and beneficial in a plethora of domains. Approaches to the problem have been categorized traditionally into two groups: extractive and abstractive. However, it is important to note that the two ways are not mutually exclusive, as hybrid models are also employed effectively.

2.1.1 | Extractive Summarization

To produce the summary, extractive methods identify and select the most defining and pertinent sections (usually partitioned on the sentence level). No extra layer of modifications are applied to the extracted sentences. Where these methods vary is with respect to their tactics in determining the importance of a sentence and filtering which particular slivers of the text to retain. TextRank, drawing inspiration from the PageRank algorithm that Google uses to empower its search engine, is a popular solution that processes text and ranks sentences with a graph-based approach (Brin & Page, 1998; Mihalcea & Tarau, 2004). LexRank is a very similar algorithm that exploits the graph structure constructed by Term Frequency - Inverse Document Frequency (TF-IDF) vectors and their cosine similarities (Erkan & Radev, 2004). The TF-IDF measure is applied in many extractive summarization applications as it seeks to represent the amount of information a given term offers. Cosine similarity is widely used in statistical analysis to measure the similarity between two vectors in a given inner product space. The formulas for TF-IDF for a single word in a document and cosine similarity are given below in Figures 1 and 2, respectively.

TF-IDF for term a in document b: $\text{TF-IDF}_{a,b} = \text{TF}_{a \text{ in } b} * log(\frac{\text{N}}{\text{DF}_a})$, where

$\text{TF}_{a \text{ in } b} = \text{frequency of } a \text{ in } b$

N = number of total documents

DF_a = number of documents containing term a

Figure 1. TF-IDF increases as there are more *a* in *b* and decreases as more documents contain *a*. A high TF-IDF score suggests that the given term is not only relevant but also key in the provided document.

$$\text{Cosine Similarity} = \frac{A \cdot B}{|A||B|}, \text{ where } A \text{ and } B \text{ are } n\text{-dimensional vectors}$$

Figure 2. Cosine similarity ranges from [-1, 1] and values closer to 1 signify similarity.

More modern methods have been developed with the assistance of the state-of-the-art pre-trained large language models – Google's Bidirectional Encoder Representations from Transformers (BERT), for instance (Abdel-Salam & Rafea, 2022; Liu & Lapata, 2019; Shukla et al., 2022). Miller employed BERT to create text embeddings that were analyzed by the *k*-means clustering algorithm for identification and extraction (2019). PacSum is another BERT-based approach that conducts its analysis and ranking with graphs (Zheng & Lapata, 2019).

Though extractive summarization approaches have been quite developed and have become rather successful solutions over the years, these models are unfit for certain requests because the summary sentences provided in the output are exactly as they are in the original input. For example, an ideal summary of an oral argument transcript would be written in a third-person point of view, but an extractive approach is restrained to the very language used by the transcript, which likely is in first-person.

2.1.2 | Abstractive Summarization

On the other hand, abstractive summarization techniques seek to learn and understand the given inputs so that they can generate novel summaries that are not simply copy-and-paste. Thus, unlike the output extractive summarization, the created summaries from these models are more comprehensible and creative, as they are not constrained to the specific sentences in the original text input. Much of research today on these approaches incorporate the Sequence-to-Sequence (Seq2Seq) architecture, which is built upon two main components: an encoder and a decoder. Simply put, encoders “encode” the input sequence of text into some representation that captures the textual meaning (often called context vectors). These context vectors are manipulated and passed into the decoder, which generates the output.

A popular type of Seq2Seq model exploited by many researchers today is the transformer, which features the attention mechanism (Vaswani et al., 2017). BART (Lewis et al., 2019) and Pegasus (Pre-training with Extracted Gap-sentences for Abstractive Summarization) (Zhang et al., 2019) are recent examples. To tackle the issue of traditional transformer models being only able to take a relatively small number of tokens as input, the Longformer architecture was proposed (Beltagy et al., 2020). However, the extent of its success has been found to be limited amongst its rivaling approaches (Shukla et al., 2022). While abstractive models seem like the ideal approach for any purpose of automatic text summarization, they are more computationally complex and require more resources to train and develop. Thus, it is common practice for researchers to fine-tune a pre-trained large language model for their specific task rather than creating a model entirely from scratch (Liu & Lapata, 2019; H. Zhang et al., 2019). Additionally, abstractive models may not fully capture the meaning of their input texts and produce factually wrong statements in their outputs, called hallucinations (Azamfirei et al., 2023). These hallucinations plague even the most advanced large language models today, posing major setbacks and reservations for this type of summarization.

To overcome many of the challenges that extractive and abstractive methodologies face when used alone, hybrid approaches have been a popular solution predominantly through the extract-then-abstract framework (Bajaj et al., 2021). As the name suggests, models with the extract-then-abstract structure first conduct extractive summarization on the input text and pass this output into the selected abstractive summarization method to generate the final summary. With this combination, the hybrid output is relatively more accurate and true while also sounding natural and coherent (Pandya, 2019).

Though much research has been conducted in this problem space, the problem has been less explored in summarizing legal documents. Some specialized extractive models enhanced with knowledge of law include LetSum (Farzindar & Lapalme, 2004), CaseSummarizer (Polsley et al., 2016), GIST (Liu & Chen, 2019), and DELSumm (Bhattacharya et al., 2021). Specifically, DELSumm embeds legal knowledge by optimizing an objective function that is built upon factors that are identified to be key to matters of law (e.g. legal experts providing a selection of "content" terms that the model should look out for and weigh more heavily) (Bhattacharya et al., 2021).

Fine-tuned BART and Pegasus have become popular options for abstractive summarization (Shukla et al., 2022). Because it is extremely crucial to have summaries that are factually correct for legal purposes, extractive methods historically have been utilized due to their natural strengths in preserving accuracy. Nevertheless, research in applying abstractive models to the legal domain has been a matter of great investigation as of recently. Especially, when paired with extractive models in the extract-then-abstract framework, these approaches have shown increasingly promising results (Pandya, 2019; Shukla et al., 2022).

2.2 | Automatic Text Simplification

Previous attempts at automatic text simplification have been applied in various domains, ranging from biomedical texts to English newspapers (Ondov et al., 2022; Carroll et al., 1998). A recent survey of automated text simplification research highlighted the major techniques and uses (Al-Thanyyan & Azmi, 2021). The work having been done in text simplification is extensive and no consensus has been reached as to what are the most effective methods. Most of the literature today breaks the problem into two sectors: lexical simplification and syntactic simplification (Al-Thanyyan & Azmi, 2021).

Lexical simplification is the process of replacing individual words or phrases with simpler synonyms. In order to find simpler synonyms, various methods propose different tactics in measuring complexity. Some rely on frequency counts in the corpi, such as the Kucera-Francis frequency which uses the Brown Corpus to measure complexity and other examples include using the Google Web Corpus (De Belder, 2010; Leroy et al., 2012) or relying on dictionaries, such as the plain language thesaurus (Ondov et al., 2022). There are various ways to address the question of finding sensical synonyms. Potential methods include measuring perplexity of potential synonyms, calculating the cosine similarities between word vectors (see Figure 2 above), or calculating probabilities of potential synonyms in original contexts (Truică et al., 2022). SimpLex, a very recently proposed architecture from Truică et al., focuses on perplexity and cosine similarity (2022). Based on recent approaches and experiments, the most effective automatic lexical simplifers use a combination of synonym features when selecting the optimal option (Al-Thanyyan & Azmi, 2021).

Syntactic simplification has also proven to be a particularly effective way of simplifying text for younger audiences (Caplan, 1992). To automate this process, it is common to establish a set of rules for changing the sentence structures to create shorter and more manageable structures (Chatterjee & Agarwal, 2022). DEPSYM is one approach that looks specifically at appositive clauses, relative clauses, conjoint clauses, and passive to active voice (Chatterjee & Agarwal, 2021). Since there are many cases to account for, the number of rules found in rule-based methods must be large so that the simplifier is versatile. However, it has proven to be a challenge to account for every single edge case (Al-Thanyyan & Azmi, 2021). More recently, due to the increased availability of more robust computational models such as transformers, there have been some proposals for data-driven approaches that learn, and thus, can apply syntactic simplification paradigms (Al-Thanyyan & Azmi, 2021).

3 | Data and Resources

Our main dataset that we use as input for summarization and simplification is the set of officially published oral transcripts from supremecourt.gov. Since 2006, the transcripts from oral arguments have been made available to the public for free and are updated as cases are completed (Wylie, 2022). Because of the limitations of the computing power we had access to for this project, we were only able to scrape and analyze 87 different transcripts.

When we needed to reference a large corpus to calculate word frequencies and *n*-gram probabilities for our lexical simplification methods, we initially used the Brown corpus. However, we quickly noticed some issues with applying the classic Brown corpus as it did not use simple language at a level of frequency that would suggest successful changes according to our replacement goals. Due to the nature of our task in simplifying text to the American middle school level, a corpus that mainly contains a relatively simple lexicon and syntactic structure was crucial. As such, we curated our own corpus from randomly selected pages on Simple Wikipedia, a version of Wikipedia where content is explained in extremely simplified terms and sentences. To do this, we used the MediaWiki API in tandem with the well-known BeautifulSoup library, which allowed us to clean the HTML tags from the scraped text. Our goal was to reach a corpus size that was comparable to the Brown corpus, which meant scraping over 10,000 Simple Wikipedia pages for a rough total of 1.1 million tokenized words. We used this corpus for selected methods as a part of our simplification step.

To implement our approaches, we relied on many free, open-source Python libraries and applications. The most notable and their use cases are shown in Table 1 below.

PyPDF2	Library to convert oral argument transcripts (given in .pdf form) to .txt files for easier processing
Legal Pegasus	A Pegasus (transformer) model fine-tuned on U.S. SEC litigation files for abstractive summarization
BERT Extractive Summarizer	A BERT-based model used for extractive summarization
spaCy	Library to parse sentences and identify where relative clauses are in given text
textstat	Library to provide readability scoring metrics
openai	Library to access the state-of-the-art GPT-3 model to compare with our outputs
matplotlib	Library to produce visuals for our test statistics
NLTK	Library for basic text manipulation (tokenization, word sense disambiguation via Lesk algorithm)

Table 1. Python libraries and applications used.

4 | Methods

Below, we explain our broad choices and general approach to tackling our goal of summarizing and simplifying text for middle school-aged children. Instead of treating the problem as one, we split the problem up into two distinct steps: summarization then simplification. We realized that this was necessary after only attempting summarization and observing that the output summaries were condensed but not approachable for readers at the English-speaking readers at the middle-school level. Many of the summaries retained their complex language and structure. Therefore, we decided to implement a second step of simplification to adjust the readability level to that of a middle-schooler. Splitting the problem up like this ended up being an ideal approach because it allows us to isolate and test various techniques at both the summarization and simplification levels to find optimal solutions.

Once we split the problem, it became clear that all of our subproblems could be reframed as a series of translation problems. The question of summarization can be reformulated as a translation question by asking: how can we map an extensive set of words and sentences to a less extensive set that still preserves the essence of the original set? The two simplification questions (lexical and syntactic) can be reframed as: how do we translate a word to its less complex form and how do we translate a sentence into simpler syntax all while preserving meaning?

The former question uses a direct machine translation (MT) approach because we map directly from one word to its counterpart. The latter uses a transfer-based MT approach because we parse a sentence based on a set of grammar rules and then transform that parse of syntactic structure into our target output. The rule is the interlingua language. Shown below, Figure 3 outlines the major design choices and overall roadmap of the project.

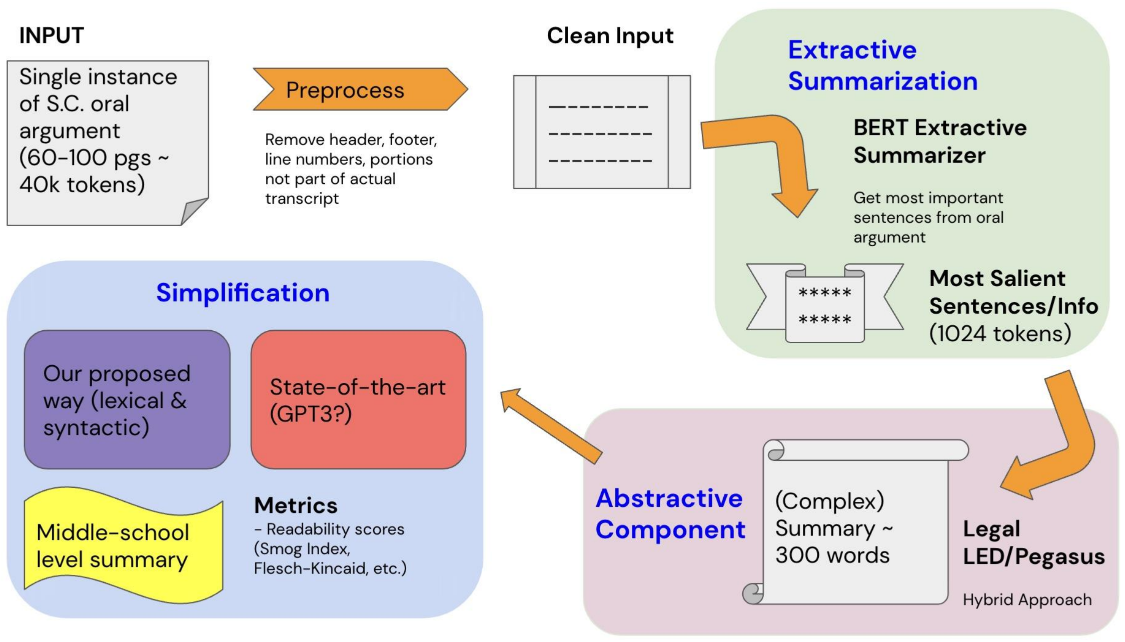

INPUT
Single instance of S.C. oral argument (60-100 pgs ~ 40k tokens)
Preprocess
Remove header, footer, line numbers, portions not part of actual transcript
Clean Input
Extractive Summarization
BERT Extractive Summarizer
Get most important sentences from oral argument
Most Salient Sentences/Info (1024 tokens)
Abstractive Component
(Complex) Summary ~ 300 words
Legal LED/Pegasus
Hybrid Approach
Simplification
Our proposed way (lexical & syntactic)
State-of-the-art (GPT3?)
Middle-school level summary
Metrics
- Readability scores (Smog Index, Flesch-Kincaid, etc.)

Figure 3: We follow the extract-then-abstract framework for summarization. We approach summarization and simplification as two discrete steps.

4.1 | Automatic Text Summarization

As stated above, we pursue three main methods for the automatic text summarization problem. To accomplish this, we explore various extractive and abstractive approaches independently, and eventually compare two hybrid approaches with a single abstractive approach.

We first propose and create our own extractive method, which relies on the intuition that sentences that contain more of the higher frequency content words are more important sentences to the overall text. In other words, the presence of more popular words in a sentence would result in that sentence having a higher relative rank. To implement this specifically, we count the frequency of stemmed content words in the original transcript. Then, we utilize a dictionary for each sentence in the input and rank them based on the total count of these stemmed content words. We call this the naive word frequency approach.

The other extractive approach we considered was BERT (Bidirectional Encoder Representations from Transformers). BERT generally had the best performance in both the BLEU and ROUGE scores from previous studies (Shukla et al., 2022).

In addition to extractive methods, we looked into state-of-the-art abstractive methods. Our research found countless recommendations of a fine-tuned version of Google's Pegasus model, called Legal Pegasus. Legal Pegasus was trained on SEC litigation files, which provided useful (however far from optimal) context when analyzing the supreme court oral argument transcripts. Because we have limited computational power and no human or golden reference summaries, we are unable to fine-tune our own large language model.

With these models selected, we chose three approaches to summarization. In the first, we began with our word-frequency extractive process, and passed in a parameter $n = 20$, which was then given to LegalPegasus. In the second, we used a chunking method to break up the original tokenized transcript into the max token size for the model, 1024. After receiving our concatenated output from these chunks, we recursively ran the summary back through the model until we were left with an output of roughly 300 words. For the last approach, we used the BERT extractive model, which again extracted a total of n=20 sentences from the original transcript. This step was followed by a single iteration of abstractive summarization with the Legal Pegasus model.

4.2 | Automatic Text Simplification

When addressing the question of simplification, we divided the process into lexical and syntactic simplification. Our method for lexical simplification involves finding the most simple synonym that matches the given context of the original word. To obtain our list of possible synonyms, we used NLTK. More specifically, we imported WordNet, a mapping of words in the English language based on meaning. By obtaining the synsets of a given word, we were able to extract all words that WordNet classifies as a synonym. This was the fastest option, however, we found that it had some limitations and did not always provide the most comprehensive list of synonyms, which may have affected results. We provide possible alternatives in the "Future Work" section of our paper.

After obtaining a list of words, we tested methods for finding which word was most complex by testing both frequency of a word in a corpus and the length of a word. We originally used the Brown corpus to test for word frequencies, but found that frequent words were not necessarily less complex. This led to our conclusion that the Brown corpus does not feature very simple vocabulary. Instead, we web scraped from SimpleWikipedia to get a corpus of mostly simple English words. By obtaining the frequency of a given word, we could figure out how complex it was since we make the assumption that less frequent words are more complex.

As part of the lexical simplification problem, we had to address the issue that some words have multiple senses for the same form, a problem commonly known as the Word Sense Disambiguation (WSD) problem. A sense is one meaning of a word (Ruggeri & Caro, 2013). For example, "bank" can refer to both a financial institution or the side of a river. Our methods for picking synonyms that match senses involve using bigram probabilities and existing methods such as the classic Lesk algorithm. The Lesk algorithm for approaching the WSD problem was proposed by Michael Lesk in 1986. See section 5.2 for a more thorough explanation of our methods.

We use bigram probabilities to test the likelihood of seeing each synonym by finding the bigram probability of seeing the synonym after the word before the original word and then the bigram probability of the synonym with the word following the original word. These probabilities are multiplied together as a score for each potential substitution word (see Figure 4 below for how we calculate the score with bigram probabilities). This ensured us that we picked the synonyms that fit the context best, assuming the best option occurs with the highest probability. As another approach, we use the NLTK library to access and use the Lesk algorithm to retrieve only the synonyms that match the sense of the word. The library implements Lesk by comparing the whole sentence as context to the WordNet definition of the synonym.

$$\text{Score for Word } X\text{:}\quad p(X|w_{prev}) * p(w_{after}|X)\text{, where}$$

$$w_{prev} \text{ is the word before where } X \text{ would be}$$

$$w_{after} \text{ is the word after where } X \text{ would be}$$

$$p(X|w_{prev}) = \frac{freq(w_{prev},X)}{freq(w_{prev})}$$

$$p(w_{after}|X) = \frac{freq(X,w_{after})}{freq(X)}$$

Figure 4: The score for a synonym replacement is the product of two bigram probabilities.

For syntactic simplification, our final method includes creating a set of rules that map complex sentences into simpler syntax. We debated other syntactic simplification techniques, such as removing adjectives or prepositional phrases, but found that these changes, which resulted in a simpler set of sentences, also resulted in a loss of meaning. Instead, we chose to focus on relative clauses because these were found to be common in the summaries that we generated and we were able to create our own set of rules. Relative clauses are dependent clauses that begin with a relative pronoun such as “who,” “who,” “whom,” “in which,” or “that.” For each relative pronoun, we created a set of rules to split up a sentence containing that pronoun into two different sentences. See Figure 5 below for an example rule.

Original sentence construction: *Subject*, “who” *Relative Clause*, *verb Object*.

Example of original: “Sam, who likes pie, says hi.”

New Sentence construction: Subject relative clause. Subject Verb Object.

Example of new construction: “Sam likes pie. Sam says hi.”

Figure 5. Example rule and procedure for syntactic simplification.

5 | Experiments and Results

5.1 | Automatic Text Summarization Experiments

As previously mentioned in the methods section, our various approaches to automatic text summarization (whether extractive, abstractive, or hybrid), helped inform countless design decisions for the project. First, we found that we had to rule out a purely extractive approach, as having a summary made entirely from excerpts of the text was difficult to follow and lacked cohesion. Our own extractive method that focused on content word frequency, which we called the naive approach, not only had the problem of creating a summary purely built from the transcript dialogue, but the sentence ranking itself was not highly effective. First, the most popular words drove similar sentences to receive similarly high scores, decreasing the variety of information extracted from the transcript. Second, highly frequent words were not inherently the best indicator of an “important” sentence, especially in dialogue, where conversations were less structured and topics could leave and return to attention quickly. Determining the most important sentences for each topic covered was our primary goal for an extractive method. The success of BERT in our research instead drove us to try this method. Using BERT, our group determined a noticeable improvement in the variety of sentences extracted from the transcript, whether it be a different person speaking or a separate subject entirely. This variety will be particularly important for our abstractive methods.

Amongst the various abstractive models available, we tested Google's Pegasus model extensively. Initial testing with subsets of the transcript (in order to meet the maximum token constraint of the model) resulted in a large number of "hallucinations," or nonfactual sequences of text present in the final summary of the model. This problem persisted throughout all of our experiments, and informed our thought-process for the rest of the project towards minimizing these inaccuracies. Because of this, we were guided towards a fine-tuned branch of Pegasus, called Legal Pegasus. Legal Pegasus brought various advantages to Pegasus, including a drastic reduction in the generation of unrelated information in the final summary, and an improvement in extracting key named-entities and organizations. However, the 1024 token limit of Legal Pegasus posed an issue with our transcripts, which had anywhere from 10,000 - 15,000 tokenized words after minimal cleaning. This pushed us towards the classic chunking method, where we extracted "chunks" of tokenized transcript to feed into the Legal Pegasus model, and proceeded to concatenate each chunk's output together. This final summary, carefully chosen to remain under the token limit, was recursively fed back into Legal Pegasus for a final iteration, resulting in a final 300-word summary. Unfortunately the Legal Pegasus model was not very effective through chunking, for a few reasons. Firstly, a limitation in using the model was that Legal Pegasus treated each "chunk" as its own summary. Concatenating these smaller summaries together resulted in huge portions of regurgitated information. Additionally, to meet the token limit for the final summary, each 1024-token chunk was being summarized into a maximum of 100 words, which proved to be a difficult task for the model; many sentences were clearly truncated in this penultimate step. For the final summary, regurgitated information in our input took up valuable space that limited the amount of unique, varied input that the model could use to build its summary, encouraging higher levels of hallucination and a reliance on the model's pre-trained context.

However, the abstractive approaches were not without their clear advantages over the extractive models. One important realization was that the concise and standard structure of the summaries produced by the abstractive methods were far easier to understand, however inaccurate they may be. With more factual information, these summaries were straightforward and featured sentence structures that would be advantageous for our simplification steps. Our group concluded that a rich and varied input to Legal Pegasus would produce the best model for oral argument transcripts. This idea led us to a hybrid extractive-abstractive approach, where we could accomplish multiple goals with each step of the combined process. For the extractive step, we chose to use BERT to extract the 20 most important sentences of the transcript, a number which would consistently provide maximum information without inducing truncation when fed into the Legal Pegasus model. Then, this extractive summary was fed once through Legal Pegasus, leaving us with our final summary. This generated summary was far more accurate than previous methods, but was still prone to smaller hallucinations; it generally performed better in following the events of the transcript itself, rather than mentioning random details of the case as a whole. Through further testing, we found additional benefits with increasing the depth of Legal Pegasus' beam search, enabling bigram-blocking, and turning off truncation for the final summary.

5.2 | Automatic Text Simplification Lexical Experiments

For text simplification, we attempted to address both lexical and syntactic simplification. To address lexical simplification, we proposed and tested six different methods. These six methods can be further divided into three groups: determining complexity, matching word sense, and combinations. Because complexity and word sense are both important factors for finding successful replacement synonyms, we hypothesized that our combination methods would perform relatively better. Below we will explain the relative successes and limitations of each method when we tested them as part of our experiments for finding the best lexical simplification method.

5.2.1 | Group 1: Determining Complexity

Our first two methods seek to explore two different techniques for determining the complexity of a given synonym.

Method 1

Our first method relies on frequency counts to determine complexity. We go through the list of synonyms for a word to check the frequency of a synonym in our Simple English corpus and replace the original word with the synonym that occurs most frequently in the corpus, since we assume more frequent words correspond to simpler words. This proved to find simpler words for substitutions, but problems arose because sometimes the chosen words did not fit the context. For example, this method changed “character limit” to “part limit.” While it is likely true that “part” is simpler than “character”, it doesn’t make sense in context. This method addresses issues of complexity but not of sense, which reaffirms our preconceived notions.

Method 2

Our second method provides an alternative to determine the complexity of synonyms based on length. For the sake of this method, we assume that shorter words are more simple. For example, the thirteen-character word “perspicacious” can be replaced with the ten-character word “insightful.” However, after many trials using various oral arguments, we found that this method was an ineffective way to measure complexity; it is simply not always the case that shorter words are simpler. For instance, this method switched “inconsistency” to “repugnance,” which is arguably more complex based on current language usage.

5.2.2 | Group 2: Matching Word Sense

For our third and fourth set of methods, we attempt to address the problem of word sense disambiguation that arose in previous methods, using two new methods.

Method 3

We use unigram, bigram, and trigram probabilities calculated from the Simple Wikipedia corpus to determine the probability of a synonym being in the context of the original word. The intuition behind this method is that words that appear together in a published and readable text should already be in the right senses. To achieve this, we assign the *n*-grams featuring those words together a higher probability for selection. We pick the synonym with the highest probability after applying add-one smoothing so that the method is deterministic. Sometimes this approach is effective and chooses words that fit the context (i.e it switches “disagree” to “dissent”), but we find that the model often relied on add-one smoothing. This tendency can be attributed to the sparsity of the text data.

Method 4

We use the NLTK library to employ the classic Lesk algorithm enhanced by WordNet to retrieve synonyms that only match the sense of the context. As mentioned previously, this library works by comparing the whole sentence as context to the WordNet definition of the synonym. We found that this approach was relatively more effective in retrieving synonyms that matched the sense of the original word.

5.2.3 | Group 3: Combining Complexity and Word Sense

Having experimented with finding separate ways to find the most complex and sensible synonym, we seek to put intuition behind our previous methods together in a hybrid approach.

Method 5

For method 5, we used the lesk measure to get a list of synonyms that fit the sense of the original word and then used the Zipf frequency to pick the least complex synonym out of those options. The Zipf frequency is a library that uses a much larger corpus to find the frequency of a synonym based on how many times it shows up per billion words (*Wordfreq · PyPI*, n.d.). We were inspired to use the Zipf from our Method 1, but found that this was more effective than using our own corpus. This was fairly effective and often succeeded. For example, it switched "determine" to "see." However, this approach also performed arguably unnecessary word switches, like "argue" to "debate."

Method 6

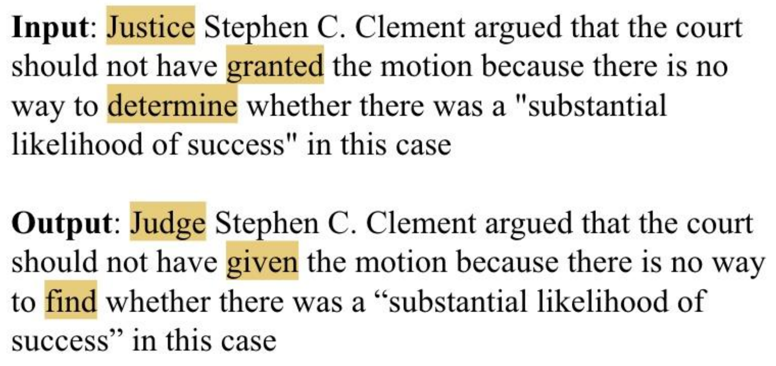
Instead, we found that the best method was method 6 that utilizes word length, frequency of the synonym in the Simple Wiki corpus, and the Lesk algorithm to pick the optimal synonym (see Figure 6 below for the exact equation). Based on the scoring metrics that we employed (see Results section for description of said metrics), this method simplified the sentence to the lowest grade level. See Figure 7 below for an example input-output sentence pairing.

$n =$ synonym (unigram) frequency in Simple Wikipedia corpus

$\ell =$ character length of synonym

synonym score = $\sqrt{n} - \ell^2$

Figure 6. We apply the square root and square operators in the combination equation to normalize the perceived ranges of *n* and ℓ (*n* generally ranged between 1 to 1000, and ℓ generally ranged from 3 to 10). The higher the score of a synonym, the better it is as a replacement.



Input: Justice Stephen C. Clement argued that the court should not have granted the motion because there is no way to determine whether there was a "substantial likelihood of success" in this case

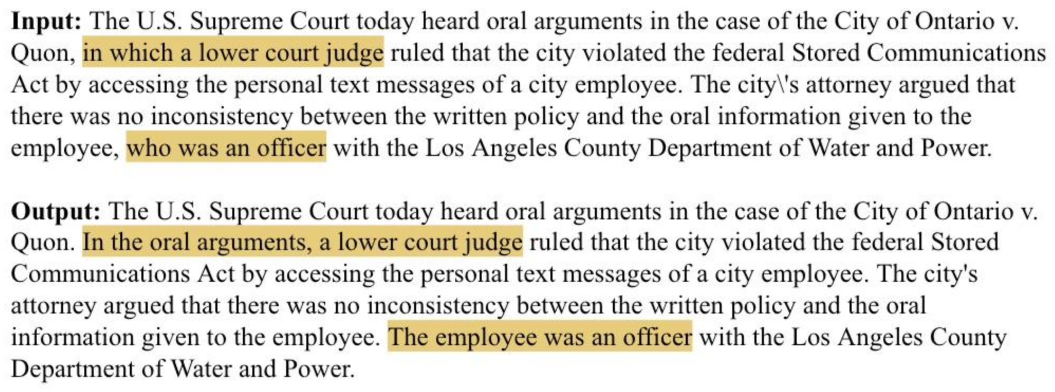
Output: Judge Stephen C. Clement argued that the court should not have given the motion because there is no way to find whether there was a “substantial likelihood of success” in this case

Figure 7. Example input sentence and output sentence using the combined method (method 6).

5.3 |Automatic Text Simplification Syntactic Experiments

As mentioned above in our Methods section, we created a set of rules in order to remove relative clauses from sentences by splitting the sentences. In general, our rule set effectively split the sentences into two new grammatical sentences (see Figure 8 below for an example).

However, we are certain that in the future, cases may arise that we were not able to account for in our rule set, which is a previously stated limitation of rule-based methods.



Input: The U.S. Supreme Court today heard oral arguments in the case of the City of Ontario v. Quon, in which a lower court judge ruled that the city violated the federal Stored Communications Act by accessing the personal text messages of a city employee. The city\'s attorney argued that there was no inconsistency between the written policy and the oral information given to the employee, who was an officer with the Los Angeles County Department of Water and Power.

Output: The U.S. Supreme Court today heard oral arguments in the case of the City of Ontario v. Quon. In the oral arguments, a lower court judge ruled that the city violated the federal Stored Communications Act by accessing the personal text messages of a city employee. The city's attorney argued that there was no inconsistency between the written policy and the oral information given to the employee. The employee was an officer with the Los Angeles County Department of Water and Power.

Figure 8. Example input and output sentence before and after removing relative clauses.

5.4 | Results: Model Scores

Five derived summaries were tested and compared: (1) the original summary, (2) the summary lexically simplified, (3) the summary syntactically simplified, (4) the summary both lexically and syntactically, and (5) the summary simplified by a state-of-the-art GPT-3 model. For lexical simplification, we used method 6 as described above because we empirically found that it generated the most favorable results after several trial runs. For syntactic simplification, we followed our rule-based relative clause breakup approach.

To assess the readability of the produced summaries, we use a variety of well-known and frequently used scoring metrics. The score values and corresponding statistics are shown in Figures 9 and 10 below. Because the Flesch-Kincaid Grade Level and Flesch Reading Ease metrics are arguably the most widely used and notable, we further provide boxplot comparisons for each measure in Figures 11 and 12, respectively (*Flesch Reading Ease and the Flesch Kincaid Grade Level – Readable*, n.d.). Note that for all measurements except the Flesch Reading Ease, the value represents the lowest grade of education that can understand the text. We call these seven different metrics, grade-level metrics. For example, a score of 7.28 signifies that the text is roughly comprehensible to 7th and 8th grade students from the American education system. In contrast, a higher value for Flesch Reading Ease represents easier readability, with scores of 80-90 signifying the American middle school level.

	Flesch Reading Ease	**Automated Readability Index**	**Dale-Chall Index**	**Flesch -Kincaid Grade Level**	**Gunning FOG Index**	**Smog Index**	**Coleman -Liau Formula**	**Linsear -Write Index**
Original	56.300 ± 9.985	13.311 ± 3.312	10.073 ± 0.972	10.976 ± 2.767	12.905 ± 2.709	12.711 ± 1.800	10.576 ± 1.946	15.663 ± 4.585
Lexical Only	68.209 ± 9.408	11.722 ± 3.296	9.743 ± 0.924	9.407 ± 2.754	11.667 ± 2.701	10.921 ± 1.608	8.378 ± 1.764	14.721 ± 4.160
Syntactic Only	59.776 ± 8.561	11.718 ± 2.523	9.498 ± 0.912	9.815 ± 2.056	11.431 ± 1.921	12.137 ± 1.462	10.187 ± 1.744	13.416 ± 4.505
Both	70.694 ± 7.694	10.157 ± 2.516	9.174 ± 0.901	8.328 ± 2.034	10.260 ± 1.939	10.437 ± 1.388	8.126 ± 1.542	12.465 ± 4.584
GPT-3	67.821 ± 9.010	10.225 ± 2.484	8.844 ± 0.850	8.501 ± 2.035	10.487 ± 2.052	10.809 ± 1.693	9.183 ± 1.600	10.584 ± 2.993

Figure 9. Mean scores and standard deviations of *n* = 87 samples.

	Flesch Reading Ease	**Automated Readability Index**	**Dale-Chall Index**	**Flesch -Kincaid Grade Level**	**Gunning FOG Index**	**Smog Index**	**Coleman -Liau Formula**	**Linsear -Write Index**
Original	56.08	13.20	9.89	10.70	12.37	12.70	10.28	15.75
Lexical Only	69.11	11.70	9.55	9.00	11.19	10.90	8.14	14.50
Syntactic Only	58.11	11.70	9.38	10.00	11.32	12.30	10.02	13.00
Both	70.57	10.20	9.05	8.10	10.06	10.70	8.01	12.00
GPT-3	68.13	9.80	8.85	8.50	10.32	10.70	9.04	10.70

Figure 10. Median scores of *n* = 87 samples.

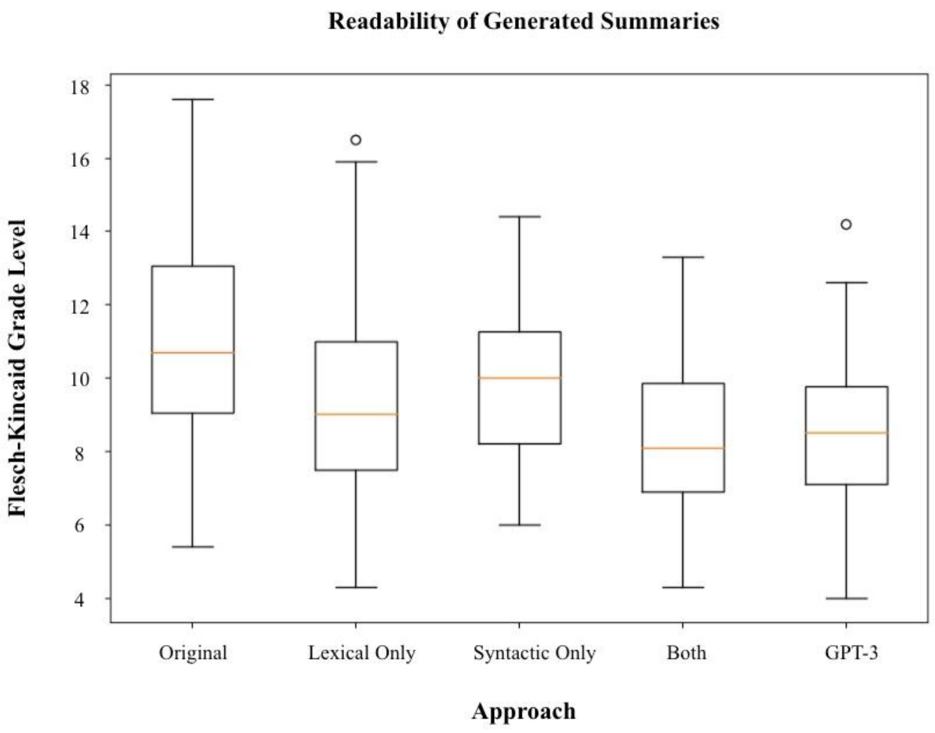


Figure 11. Boxplot of scores based on the Flesch-Kincaid Grade Level metric.

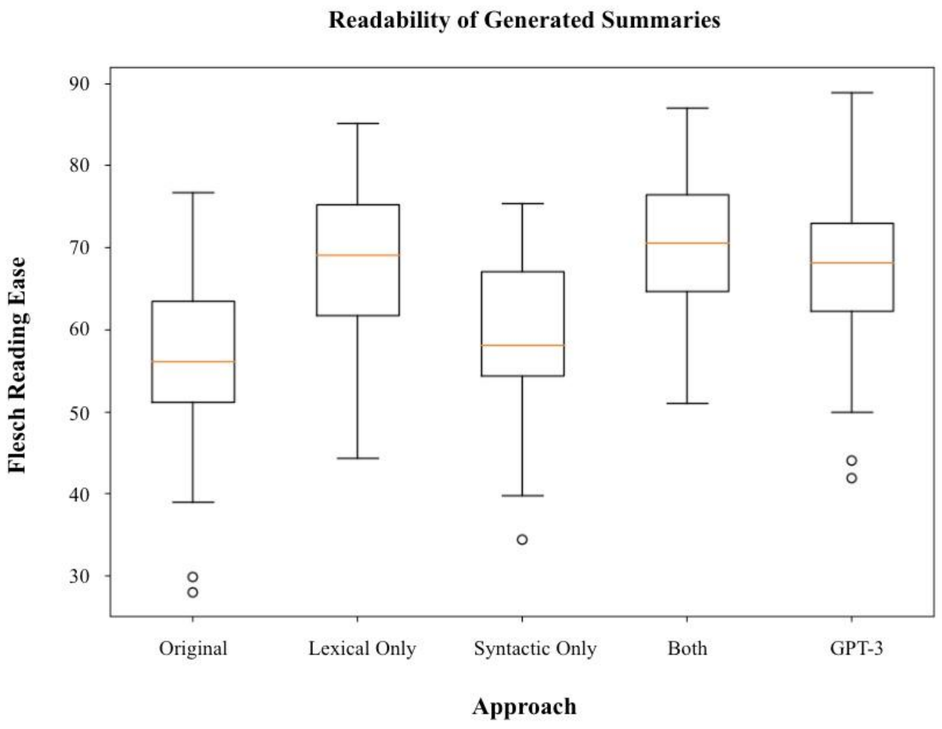


Figure 12. Boxplot of scores based on the Flesch Reading Ease metric.

6 | Conclusions

We draw upon the quantitative results from the last section as well as the qualitative results throughout our experimentation process to make these conclusions.

For the task of summarization, we find that a hybrid approach, specifically with the BERT and Legal Pegasus transformers, results in the most effective method for summarizing Supreme Court oral arguments. We arrived at this conclusion through qualitative and empirical analysis from reading the 87 summaries this approach generated. To test our simplification methods, we use these summaries from this approach as the "original" input.

For the task of simplification, we find that our method of combining lexical and syntactic approaches are productive. As shown in Figure 9 from Section 5, we see an average 25.56% increase in the Flesch Reading Ease score and an average of a 20.03% drop in all grade level metrics when compared to the original summary score when both simplification methods are applied. Even when lexical simplification is applied alone, we see a 21.15% increase in the Flesch Reading Ease score and an average of an 11.20% drop in all grade-level metrics. Based on these metrics, our model is competitive with a state-of-the-art model GPT-3's performance, where we see an average 20.46% increase in the Flesch Reading Ease score and an average of a 20.39% drop in all grade level metrics. In Figures 11 and 12, the boxplots displaying our simplified summaries show improvement with respect to the original summaries. In general, we see the boxplots for our simplification methods shifted relatively lower and higher in Figures 11 and 12, respectively.

A limitation with these results that is not expressed in the quantitative readability and grade-level scores is in real-world performance. While the simplification step of the model succeeds in making frequent lexical and syntactical improvements, it is also prone to outlandish lexical replacements, which reduces readability in real-world usage. As for syntactical changes, breaking up all relative clauses may in theory make summaries structurally simpler and “easier” to read, but some outputs resulted in choppier sentences that made a sentence more difficult to follow and arguably grammatically questionable with respect to current language usage.

Lastly, we found that further, implementing additional successful syntactic simplification would require an incredibly robust rule set to simplify all grammatical cases that could be seen in a paragraph of English text. However, as demonstrated by the average 6.17% increase (Flesch Reading Ease score) and average 9.29% drop in all grade level metrics over the baseline summary, even a relatively simple simplifier with a few rules can improve readability for less advanced English readers.

7 | Future Work

There is much work that can be done to build upon this project. In terms of summarization, we were lucky to have access to the Legal Pegasus transformer which already was trained on SEC litigation files. However, we still prefer to fine-tune our own transformer model on legal oral arguments specifically.

For lexical simplification, we were limited by the synonyms that WordNet offered. However, in the future, we hope to figure out how to obtain a more comprehensive list of synonyms that can give us better results. One way of doing this would be to web scrape from a website such as thesaurus.com. We also focused solely on lexical word replacement, but believe that we can obtain better results if we also implement phrasal replacement. This would require us to find synonyms on a phrasal level and then apply the same complexity and sense tests.

In general, we also understand that we want to achieve our goal of summarizing and simplifying for a middle school student in a single step to make this process more streamlined. Therefore, one future goal is to utilize a transformer model to perform summarization and simplification steps simultaneously, using an argument associated with grade level to specify the level of simplification.

Ideally, testing our model with more than our current limited set of 87 examples, having human-written summary references, and utilizing human interviews for a human-based metric would improve on the robustness of our results. These additions would demystify the difference in results between our group's human interpretation and the respective scores shown for the readability metrics. In particular, human reference would enable us to test our model with the standard BLEU and ROUGE scores, which would give a better understanding of the quality of our summaries, rather than relying solely on the simplicity of the provided summaries and our intuition.

8 | Honor Code and Contributions

8.1 | Honor Code

This report represents our own work in accordance with university regulations.

/s/ John Van Horn

/s/ James Zhang

/s/ Gillian Rosenberg

8.2 | Contributions

John Van Horn: wrote methods section on various summarization options, data section of SimpleWikipedia web scraping, conclusions, future work, experiments section on various summarization options, literature review on summarization, researched libraries and coded text pre-processing, pre-trained LegalPegasus model chunking implementations, GPT-3 implementation, wiki web-scraping implementation, final testing loop to extract scores, coded naive extractive implementation, coded BERT-LegalPegasus hybrid summarization function, coded simplification Method 6 (combination lexical simplification)

James Zhang: paper's literature review of automatic text summarization, data (and its collection), overview of methods, creating graphics, refined overall language; managed code base (clean up and formatting); coded relative clause rule-based simplification; found & implemented pre-trained models to use; created slides presented on showcase day; ran tests on local computer for all scores; proposed the scoring metrics; outlined and developed project roadmap

Gillian Rosenberg: Paper's introduction, literature review of simplification, general methods section, lexical and syntactic simplification methods section, experiments and results for simplification, future work; researched libraries for part-of-speech tagging and synonym list creator; coded five separate functions for lexical simplification; coded the testing client; assisted with syntactic simplification methods; created slides presented on showcase day.

References

Abdel-Salam, S., & Rafea, A. (2022). Performance Study on Extractive Text Summarization Using BERT Models. *Information*, *2*, 67. https://doi.org/10.3390/info13020067

Al-Thanyyan, S. S., & Azmi, A. M. (2021). Automated Text Simplification. *ACM Computing Surveys*, *2*, 1–36. https://doi.org/10.1145/3442695

Azamfirei, R., Kudchadkar, S. R., & Fackler, J. (2023). Large language models and the perils of their hallucinations. *Critical Care*, *1*. https://doi.org/10.1186/s13054-023-04393-x

Bajaj, A., Dangati, P., Krishna, K., Kumar, P., Uppaal, R., Windsor, B., Brenner, E., Dotterrer, D., Das, R., & McCallum, A. (2021). *Long Document Summarization in a Low Resource Setting using Pretrained Language Models*. https://doi.org/https://doi.org/10.48550/arXiv.2103.00751

Beltagy, I., Peters, M., & Cohan, A. (2020). *Longformer: The Long-Document Transformer*. https://doi.org/https://doi.org/10.48550/arXiv.2004.05150

Bhattacharya, P., Poddar, S., Rudra, K., Ghosh, K., & Ghosh, S. (2021). *Incorporating Domain Knowledge for Extractive Summarization of Legal Case Documents*.

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, *1–7*, 107–117. https://doi.org/10.1016/s0169-7552(98)00110-x

Caplan, D. (1992). *Language: Structure, processing, and disorders*. MIT Press.

Carroll, J., Minnen, G., Canning, Y., Devlin, S., & Tait, J. (1998). Practical Simplification of English Newspaper Text to Assist Aphasic Readers. *Proc. of AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*.

Chatterjee, N., & Agarwal, R. (2021). DEPSYM: A Lightweight Syntactic Text Simplification

Approach using Dependency Trees. *CEUR Workshop Proceedings*.

Chatterjee, N., & Agarwal, R. (2022). Studying the Effect of Syntactic Simplification on Text Summarization. *IETE Technical Review*, 1–12. https://doi.org/10.1080/02564602.2022.2055670

De Belder, J. (2010). Text simplification for children. *KU LEUVEN Libraries*.

Erkan, G., & Radev, D. R. (2004). LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, 457–479. https://doi.org/10.1613/jair.1523

Farzindar, A., & Lapalme, G. (2004). LetSum, an Automatic Text Summarization system in Law field. *Jurix*.

Flesch Reading Ease and the Flesch Kincaid Grade Level – Readable. (n.d.). Readable; https://www.facebook.com/ReadableHQ. Retrieved May 7, 2023, from https://readable.com/readability/flesch-reading-ease-flesch-kincaid-grade-level/

Leroy, G., Endicott, J., Mouradi, O., Kauchak, D., & Just, M. (2012). Improving Perceived and Actual Text Difficulty for Health Information Consumers using Semi-Automated Methods. *National Library of Medicine*.

Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. *SIGDOC '86*.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2019). *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. https://doi.org/https://doi.org/10.48550/arXiv.1910.13461

Liu, C.-L., & Chen, K.-C. (2019). Extracting the Gist of Chinese Judgments of the Supreme

Court. *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law*. https://doi.org/10.1145/3322640.3326715

Liu, Y., & Lapata, M. (2019). Text Summarization with Pretrained Encoders. *ACL Anthology*. https://doi.org/https://doi.org/10.48550/arXiv.1908.08345

Mihalcea, R., & Tarau, P. (2004). TextRank: Bringing Order into Texts. *ACL Anthology*.

Miller, D. (2019). Leveraging BERT for Extractive Text Summarization on Lectures. *Arxiv.Org*. https://doi.org/https://doi.org/10.48550/arXiv.1906.04165

Ondov, B., Attal, K., & Demner-Fushman, D. (2022). A survey of automated methods for biomedical text simplification. *Journal of the American Medical Informatics Association*, *11*, 1976–1988. https://doi.org/10.1093/jamia/ocac149

Oral Arguments - Supreme Court of the United States. (n.d.). Home - Supreme Court of the United States. Retrieved May 10, 2023, from https://www.supremecourt.gov/oral_arguments/oral_arguments.aspx

Pandya, V. (2019). AUTOMATIC TEXT SUMMARIZATION OF LEGAL CASES: A HYBRID APPROACH. *Arxiv.Org*. https://doi.org/https://doi.org/10.5121/csit.2019.91004

Polsley, S., Jhunjhunwala, P., & Huang, R. (2016). CaseSummarizer: A System for Automated Summarization of Legal Texts. *ACL Anthology*.

Ruggeri, A., & Caro, L. (2013). *Linguistic Affordances: Making sense of Word Senses*.

Shukla, A., Bhattacharya, P., Poddar, S., Mukherjee, R., Ghosh, K., Goyal, P., & Ghosh, S. (2022). Legal Case Document Summarization: Extractive and Abstractive Methods and their Evaluation. *ACL Anthology*. https://doi.org/https://doi.org/10.48550/arXiv.2210.07544

Transcripts and Recordings of Oral Arguments - Supreme Court of the United States. (n.d.).

Home - Supreme Court of the United States. Retrieved May 10, 2023, from https://www.supremecourt.gov/oral_arguments/availabilityoforalargumenttranscripts.aspx

Truică, C.-O., Stan, A.-I., & Apostol, E.-S. (2022). SimpLex: a lexical text simplification architecture. *Neural Computing and Applications*, *8*, 6265–6280. https://doi.org/10.1007/s00521-022-07905-y

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. *31st Conference on Neural Information Processing Systems*. https://doi.org/https://doi.org/10.48550/arXiv.1706.03762

wordfreq · PyPI. (n.d.). PyPI. Retrieved May 10, 2023, from https://pypi.org/project/wordfreq/

Wylie, S. (2022, March 31). *The Supreme Court Must Continue To Provide Live Audio Broadcasts of Oral Arguments - Center for American Progress*. Center for American Progress. https://www.americanprogress.org/article/the-supreme-court-must-continue-to-provide-live-audio-broadcasts-of-oral-arguments/

Zhang, H., Cai, J., Xu, J., & Wang, J. (2019). Pretraining-Based Natural Language Generation for Text Summarization. *ACL Anthology*.

Zhang, J., Zhao, Y., Saleh, M., & Liu, P. (2019). *PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization*. https://doi.org/https://doi.org/10.48550/arXiv.1912.08777

Zheng, H., & Lapata, M. (2019). Sentence Centrality Revisited for Unsupervised Summarization. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/p19-1628