IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.53

Volume 5, Issue 2, January 2025

Legal Document Summarizer using Spacy and BART

Achutam Varpe[1], Omkar Ugale[2], Rahul Ugale[3], Saurabh Pagar[4] and Prof. Narendra Joshi[5]

Students, Department of Cloud Technology and Information Technology[1-4]

Guide, Department of Cloud Technology and Information Technology[5]

Sandip University, Nashik, India

Abstract*:* *Legal professionals face the growing challenge of managing large amounts of complex legal documents. This paper introduces a new solution that combines machine learning (ML) and natural language processing (NLP) to change the way legal documents are summarized and analyzed. Our approach leverages the ability of ML algorithms to extract important information from a variety of legal documents including contracts, court decisions, laws, and legal opinions and then uses NLP algorithms to transform that data this has been filtered into a clear and concise summary that captures the main content of original documents. This summary is not only highly accurate but also scalable to a wide range of legal documents, saving legal professionals time and resources. The authors experimented with domain-independent model for legal text summarization, called BART. Summarized documents are evaluated by registered experts against various criteria and using the ROUGE metric this shows that the text summarization is effective in legal texts with independent domain-model.*

Keywords: NLP, Text Analysis, Extractive Summarization, Text Classification, Abstractive Summarization, Legal Terminology

I. INTRODUCTION

In the legal profession, the task of managing an ever-increasing complexity of legal documents has become a daunting challenge. Lawyers must contend with a variety of documents including contracts, court decisions, laws and legal opinions, which often require a great deal of time and effort to sift through This challenge requires innovative solutions that can increase efficiency, productivity and decision making in the law.

Our paper presents a sophisticated approach that leverages the power of machine learning (ML) and natural language processing (NLP) to transform how legal documents are analyzed and compiled. This approach uses ML algorithms to extract important insights from a wide range of legal documents, then the NLP algorithm processes these insights into a comprehensive summary that faithfully captures the essence of the original documents needs and time and storage. Our approach covers and also includes basic features such as document prepossessing, named entity (NER), information classification, sentence reduction, dummy summary generation, and use of legal keywords allows users the flexibility to create longer summaries by focusing on specific aspects or topics in documents. The benefits of this new approach extend to lawyers and organizations, including time efficiency, improved decision making, reduced investigative risk, cost savings, potentially has made changes to a wide range of regulatory documents, and improved performance. By automating and optimizing the document summary process, our system empowers lawyers to be more productive and make informed decisions faster. The author of this paper has tried to do summarization in two parts first is extractive summarization and second one is abstractive summarization. For extractive summarization we are using pytextrank algorithm and for abstractive we are using

BART. Section 3 includes the literature survey i.e., prior work related to proposed methodology. Section 4 discusses the detailed explanation of the proposed methodology. Section 5 contains evaluation of our model. Section 6 includes future scope of our project and Finally, section 7 concludes the paper.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 5, Issue 2, January 2025

II. PROPOSED METHODOLOGY

Our methodology for legal document summarization leverages a web-based platform developed using Django, integrated with a MySQL database for effective data storage and retrieval. The system employs two core methods for document summarization:

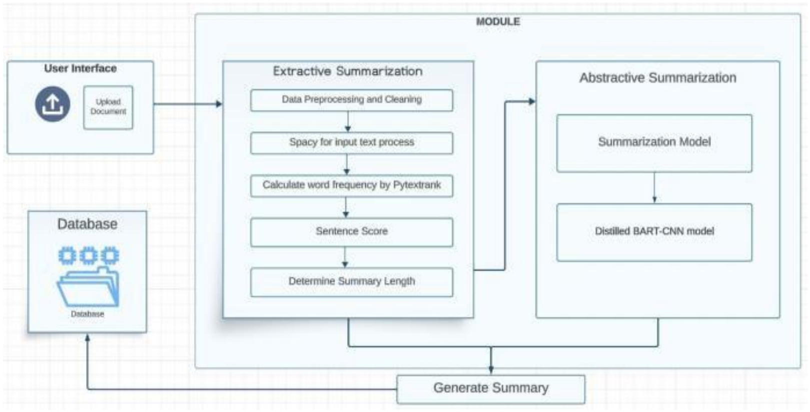


Fig 1. Block Diagram of The Proposed Methodology

A. Extractive Summarization

The initial step is the Data Preprocessing Module, which involves thorough text cleaning to remove unnecessary characters, formatting, and whitespace. Moreover, we eliminate common stop-words that do not significantly contribute to the document’s meaning.

spaCy

spaCy is an open-source natural language processing (NLP) library designed for industrial-strength use. It is developed by Explosion AI and provides efficient tools for processing and analyzing textual data.

Tokenization in spaCy is a process of breaking a text into individual units, such as words or sentences. SpaCy provides a pre-trained statistical model for tokenization that you can use to efficiently tokenize text.

PyTextRank

PyTextRank is a Python library that implements text ranking algorithms for extractive summarization and keyphrase extraction. It is built on top of spaCy, a popular natural language processing (NLP) library. PyTextRank uses graph-based ranking algorithms to identify important sentences and phrases in a text document.

Key functionalities of PyTextRank include:

- Graph based ranking: PyTextRank constructs a graph representation of the document, where nodes represent sentences or phrases, and edges represent the relationships between them.
- TextRank Algorithm: PyTextRank applies the TextRank algorithm to the constructed graph to rank the nodes (sentences or phrases) based on their importance. TextRank is a graph-based ranking algorithm inspired by PageRank, which was originally designed for ranking web pages.
- Sentence Extraction: The library extracts the most important sentences from the document based on their rankings. These sentences collectively form an extractive summary of the document.
- Key-phrase Extraction: PyTextRank can also identify key-phrases in the document by selecting important phrases based on the graph-based rankings.
- Integration with SpaCy: PyTextRank is designed to work seamlessly with spaCy. It extends spaCy's functionality by providing additional capabilities for text summarization and key-phrase extraction.

Steps of implementation nlp = spacy.load("en_core_web_lg") nlp.add_pipe("textrank")
PyTextRank is added to the spaCy pipeline. *word.text.lower().strip("\n") not in stop_words and word.text.lower() not in punctuation*

IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.53

Volume 5, Issue 2, January 2025

Checks if the lowercase version of the word (stripped of newline characters) is not in the list of stop words and if the lowercase version of the word is not in the list of punctuation marks.

word_freq[word.text] = word_freq.get(word.text, 0) + 1

Updates the frequency count for each word in the word_freq dictionary. The get method is used to retrieve the current frequency of the word. If the word is not present in the dictionary, it defaults to 0, and 1 is added to it. *if word_freq[word.text] > max_freq: max_freq = word_freq[word.text]*

Tracks the maximum frequency (max_freq) encountered while processing the document. If the current word's frequency is greater than the current maximum frequency, the max_freq variable is updated.

sent_tokens = [sent for sent in doc.sents]

This part extracts sentences from the processed text (doc). It uses a list comprehension to create a list (sent_tokens) containing spaCy Span objects, each representing a sentence in the document.

sent_scores[sent] = sent_scores.get(sent, 0) + word_freq[word.text]

This part calculates sentence scores based on the normalized word frequencies. It iterates through each sentence in sent_tokens and then through each word in the sentence. If the word is present in the normalized word_freq dictionary, its normalized frequency is added to the sentence score in the sent_scores dictionary.

The sentence scores are calculated by summing the normalized frequencies of words that are present in each sentence. Then we set the variable select_len based on the length of the input text (text). The value of select_len is determined by different conditions depending on the length of the text. If the length of the text is greater than large_text_threshold and less than 500,000 characters, select_len is set to 3% of the number of sentences. If the length of the text is greater than 500,000 characters, select_len is set to 4% of the number of sentences. If the length of the text is between 30,000 and 50,000 characters, select_len is again set to 4% of the number of sentences. If none of the above conditions are met, select_len is set to 10% of the number of sentences.

summary = nlargest(select_len, sent_scores, key=sent_scores.get)

Here, the nlargest function from the heap module to select the top select_len sentences from the sent_scores dictionary based on their scores.

final_summary = [word.text.replace("\n", "") for word in summary]

Creates a list (final_summary) by iterating over each sentence in the selected summary and removing newline characters ("\n") from each word in the sentence.

This step is done to clean up the formatting of the summary.

if word_freq[word.text] > max_freq: max_freq = word_freq[word.text]

Tracks the maximum frequency (max_freq) encountered while processing the document. If the current word's frequency is greater than the current maximum frequency, the max_freq variable is updated.

sent_tokens = [sent for sent in doc.sents]

This part extracts sentences from the processed text (doc). It uses a list comprehension to create a list (sent_tokens) containing spaCy Span objects, each representing a sentence in the document.

sent_scores[sent] = sent_scores.get(sent, 0) + word_freq[word.text]

summary = " ".join(final_summary)

Joins the cleaned words from final_summary into a single string, separated by spaces. This creates the final summarized text.

B. Abstractive Summarization

The heart of our summarization approach lies in the Abstractive Summarization module. Here, we implement a distilled BART-CNN model, which allows us to generate coherent and concise summaries by comprehending and rewriting the content. We also ensure that the generated summaries adhere to predefined length constraints, making them user-friendly and easily digestible. To provide a seamless experience for users, we develop a user-friendly User Interface in the project. This frontend enables users to interact with the summarizer effortlessly. Users can upload legal documents, and the system handles the input and output efficiently, presenting summaries in a userfriendly format.

C. User Interface Django

Python-based Django is a high-level web framework that promotes efficient development and simple, straightforward design. Although it adheres to the Model-View-Controller (MVC) architectural pattern, the ModelView-Template (MVT) pattern is used in Django. Django offers a collection of tools and protocols that facilitate the rapid and effective development of web applications by developers.

One key aspect of Django is its Object-Relational Mapping (ORM) system, which simplifies database interactions. The ORM allows you to define your data models using Python classes, and Django takes care of translating these classes into database tables and handling database queries. Components:

1. Models from django.contrib import
admin from .models import SummaryData class SummaryAdmin(admin.ModelAdmin):
list_display = ('id','file_name', 'upload_date','summary')

A model is a Python class that defines the structure of a database table. Each attribute of the class represents a field in the table. Models define the data schema and include information about fields' types, relationships, and constraints.

2. Migrations
Python manage.py makemigrations Python manage.py migrate

Django uses migrations to manage changes to the database schema over time.When you create a new model or modify an existing one, you create a migration that represents the changes. Migrations are then applied to the database to update its schema accordingly.

3. Database Configuration
DATABASES = {
'default': {
'ENGINE': 'django.db. backends.mysql', 'NAME': 'DocumentSummarizer', 'USER': 'root',
'PASSWORD': 'sonu@18rc', 'HOST': 'localhost',
'PORT': '3306',
}
}

Django supports various databases, including PostgreSQL, MySQL, SQLite, and Oracle but in this project we have used MySQL. Here we have configured the database connection in the settings.py file, specifying details like database engine, name, user, password, host and port.

4. Admin Interface

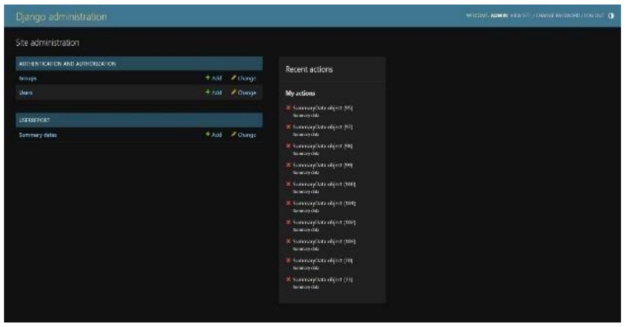
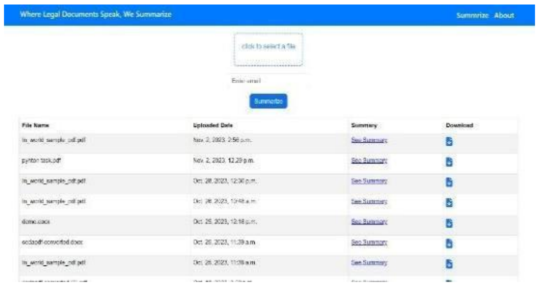


Fig 2: Admin Dashboard

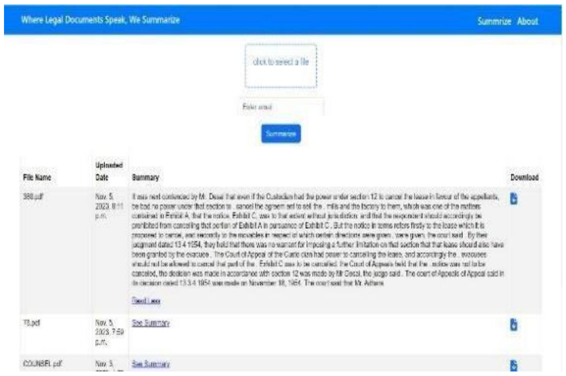
Django includes a built-in admin interface that automatically generates a user-friendly interface for managing database records. Developers can customize the admin interface to suit their application's needs


Where Legal Documents Speak, We Summarize
Summrize About
click to select a file
Enter email
Summarize

File Name	Uploaded Date	Summary	Download
hi_world_sample_pdf.pdf	Nov. 2, 2023, 2:56 p.m.	See Summary	
pyhton task.pdf	Nov. 2, 2023, 12:29 p.m.	See Summary	
hi_world_sample_pdf.pdf	Oct. 28, 2023, 12:30 p.m.	See Summary	
hi_world_sample_pdf.pdf	Oct. 28, 2023, 10:48 a.m.	See Summary	
demo.docx	Oct. 25, 2023, 12:18 p.m.	See Summary	
ocdapdf-converted.docx	Oct. 26, 2023, 11:39 a.m.	See Summary	
hi_world_sample_pdf.pdf	Oct. 26, 2023, 11:38 a.m.	See Summary	

Fig 3: Homepage (before summarization)

Here on the homepage we can see the option of upload file, we have to upload file in the form of word/pdf. Click on summarize.


Where Legal Documents Speak, We Summarize
Summrize About
click to select a file
Enter email
Summarize

File Name	Uploaded Date	Summary	Download
388.pdf	Nov. 5, 2023, 8:11 p.m.	It was next contended by Mr. Desai that even if the Custodian had the power under section 12 to cancel the lease in favour of the appellants, he had no power under that section to . cancel the agreement to sell the . mills and the factory to them, which was one of the matters contained in Exhibit A, that the notice, Exhibit C, was to that extent without jurisdiction, and that the respondent should accordingly be prohibited from canceling that portion of Exhibit A in pursuance of Exhibit C . But the notice in terms refers firstly to the lease which it is proposed to cancel, and secondly to the movables in respect of which certain directions were given . were given, the court said . By their judgment dated 13.4.1954, they held that there was no warrant for imposing a further limitation on that section that the lease should also have been granted by the evacuee . The Court of Appeal of the Custodian had power to canceling the lease, and accordingly the . evacuees should not be allowed to cancel that part of the . Exhibit C was to be cancelled. The Court of Appeals held that the . notice was not to be canceled, the decision was made in accordance with section 12 was made by Mr Desai, the judge said . The court of Appeals of Appeal said in its decision dated 13.3.4.1954 was made on November 18, 1954. The court said that Mr. Asthana Read Less	
78.pdf	Nov. 5, 2023, 7:59 p.m.	See Summary	
COUNSEL.pdf	Nov. 5,	See Summary	

Fig 4: Homepage (After summarization)

After clicking on summarize the system will start the summarization and the summary will be displayed in the table as shown in image and we also provide the option of downloading the summary the summary will be downloaded the pdf format.

III. EVALUATION

We used the ROUGE metrics, which are commonly used to assess the quality of generated summaries by comparing them to reference summaries (human-written summaries). The metrics are typically presented in terms of Precision (P), Recall (R), and F1-score (F1) for various n-gram levels (e.g., unigrams, bigrams, trigrams, etc.), as well as for different summary lengths.:

TABLE I: ROUGE SCORE FOR THE SUMMARIZATION TEXT

Type	Precision	Recall	F-Score
ROUGE-1	41.29	39.30	40.26
ROUGE-2	12.32	11.75	12.02
ROUGE-3	5.08	4.91	5.00
ROUGE- L	32.96	31.67	32.30

ROUGE-1 (unigram) F1-score for Hypothesis 0 and Reference 0 is 41.29, indicating that the first hypothesis summary is quite similar to the first reference summary in terms of unigrams.
On the other hand, the ROUGE-2 (bigram) F1-score for Hypothesis 0 and Reference 0 is 12.32, which suggests a lower level of overlap for bigrams.
Similarly, other ROUGE metrics (ROUGE-3, ROUGE-4 and ROUGE-L) are provided for this specific hypothesis reference pair. It is difficult to obtain the perfect scores using ROUGE for the extractive summarization

so we evaluated our summarized text by legal experts. We send samples of our model and asked them to evaluate result based on accuracy and details in the summarization.
The following diagram shows the result,

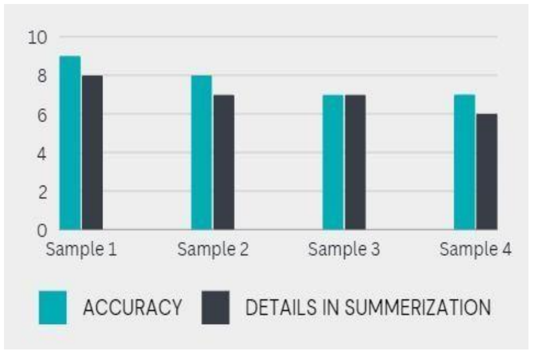


Fig 5. Result of Legal Expert

IV. FUTURE SCOPE

The utilization of the Pytextrank algorithm in conjunction with the distilled BART CNN model for a Legal Document Summarizer represents a notable advancement in the application of machine learning to the legal domain. However, it is essential to acknowledge certain limitations and consider potential future directions for this project. Summarizing lengthy legal documents while retaining essential details can be challenging, and the system may struggle with the specificity of legal terminology. Furthermore, ethical and legal considerations are paramount, especially when handling sensitive or confidential materials.
Looking ahead, future opportunities abound. Customizing the summarizer for specific legal subfields, multilingual support, enhanced semantic understanding, and integration of knowledge graphs or legal ontologies are avenues for improvement. The system could evolve to offer comprehensive legal research assistance, compliance with ethical and regulatory standards, user customization, standardized evaluation metrics, and integration with other Legal-Tech tools.
In conclusion, the future scope for a Legal Document Summarizer is promising, with ample room for development and refinement to better serve the needs of legal professionals and researchers in an ever-evolving legal landscape.

V. CONCLUSION

The use of the Pytextrank algorithm and the distilled BART-CNN model in a Legal Document Summarizer demonstrates a powerful tool for legal professionals, researchers, and anyone who needs to quickly grasp the content of complex legal documents. It streamlines the process of summarizing legal documents, saving time and effort, and has the potential to significantly improve the efficiency and accuracy of legal work. However, it's important to note that the effectiveness of the summarization system may vary depending on the quality and complexity of the input legal documents and the specific needs of the users. Ongoing refinement and evaluation of the system will be crucial to ensure its optimal performance.

REFERENCES

[1]. Ahsaas bajaj, and Pavitra Dangati, "Long Document Summarization in a Low Resource Setting using pretrained language models", March 2021.
[2]. Iz Beltagy, Matthew E. Peters, Arman Cohan, "Longformer: The Long-Document Transformer", Dec 2020.
[3]. Yue Dong, "A survey on Neural Network-Based Summarization Methods", April 2018.
[4]. Alexios Gidiotis, and Grigorios Tsoumakas "A Divide-and-Conquer Approach to the Summarization of Long Documents", Sep 2020.
[5]. Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, Phil Blunsom "Teaching Machines to Read and Comprehend"

[6]. Dandan Huang, Leyang Cui, Sen Yang, Guangsheng Bao, Kun Wang, Jun Xie, Yue Zhang "What Have We Achieved on Text Summarization?"

[7]. Philippe Laban, Tobias Schnabel, Paul N. Bennett, Marti A. Hearst "SUMMAC: Re-Visiting NLI-based Models for Inconsistency Detection in Summarization", Feb 2022.

[8]. Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, Luke Zettlemoyer "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension", Oct 2019.

[9]. Tharindu Ranasinghe, Constantin Orasan and Ruslan Mitkov "Enhancing Unsupervised Sentence Similarity Methods with Deep Contextualized Word Representation"

[10]. Paheli Bhattacharya, Kaustubh Hiware, Shubham Rajgaria, Nilay Pochhi, Kripabandhu Ghosh, and Saptarshi Ghosh" A Comparative Study of Summarization Algorithms Applied to Legal Case Judgments

[11]. Satyajit Ghosh, Mousumi Dutta, Tanaya Das "Indian Legal Text Summarization: A Text Normalization-based Approach", Sep 2022.

[12]. Rahul C Kore, Prachi Ray, Priyanka Lade, Amit Nerurkar "Legal Document Summarization Using NLP and ML Techniques", May 2020.

[13]. Khushboo S. Thakkar, Dr. R. V. Dharaskar, M. B. Chandak "Graph-Based Algorithms for Text Summarization"