IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 1, January 2024

DOI: 10.17148/IJARCCE.2024.13118

Multi Document Summarisation Using Rule Engine

Virendra Yadav[1], Anshul Gedam[2], Sanskar Korekar[3], Pranay There[4], Sujal Bitle[5], Tarang Bhaisare[6]

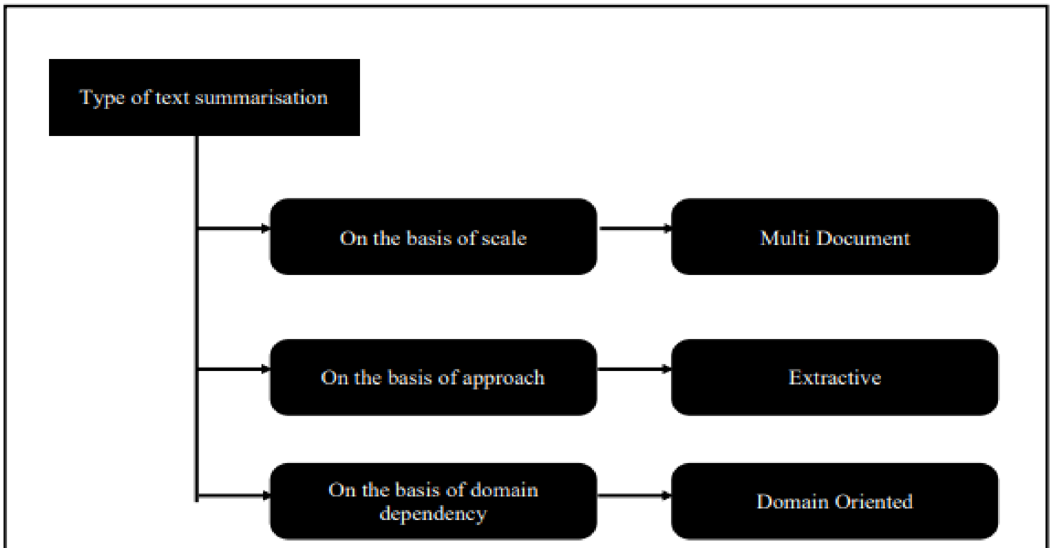
Department of Computer Science and Engineering, Priyadarshini College of Engineering, Nagpur, India[1-6]

Abstract: In today's digital generation, we have access to an enormous amount of information. However, it can be a time-consuming task for users to read, sort, and summarise all this information. To address this challenge, a method called multi-document summarisation has been developed using a rule engine. Multi-document summarisation involves the extraction of valuable and relevant information from a collection of uploaded documents. It aims to create concise and coherent summaries of the content within these documents. This approach is essential in various applications, including market reviews, search engines, and business analysis. Summarising multiple documents in this manner enables users to quickly obtain the necessary information from the entire set of referenced documents. The specific approach used in multi-document summarisation is extractive, which means it selects and combines existing content from the source documents to create a summary. The goal of the research paper mentioned is to employ a rule engine to summarise text from multiple documents using an extractive approach. This will result in a coherent and meaningful output, benefiting users in managing and extracting information from a large volume of documents.

Keywords: Document summarization, Extractive approach, Rule Engine, domain oriented

I. INTRODUCTION

In contemporary academic and research contexts, the increasing volume and diversity of scholarly publications and information sources present formidable challenges for researchers in terms of information assimilation and synthesis. In response to these challenges, the field of text summarisation and document summarisation has emerged as a potential solution. The primary goal is to contribute for the advancement of document summarisation through the utilisation of rule-based engines. Specifically, the research seeks to develop a system that automates the generation of concise document summaries from a diverse range of documents accessible through a web application. These summaries will be of a generic nature, encompassing the entire content of the documents uploaded and presenting it in a condensed form. Multi-document summarisation is notably more challenging than its counterpart, single-document summarisation, which is designed to extract summaries from individual documents. Single-document summarisation is inherently more straightforward in terms of document.



Type of Summerisation Used

1.1 On the basis of scale: Multi documents

The process of extracting data and information from multiple uploaded documents to create a comprehensive summary is commonly referred to as "multi-document summarisation." It is important necessary to remember that arrangement and sequence due to its isolated focus.

The underlying rationale behind the pursuit of multi-document summarisation lies in its ability to facilitate the needs of researchers, search engines, market analysts, and other pertinent stakeholders in the efficient analysis of substantial volumes of data. This process enables the conversion of vast datasets into concise and succinct summaries, all in reference to the multiple documents that have been uploaded and aggregated for analysis.

1.2 On the basis of approach: Extractive Summarisation

The methodology employed for multi-document summarisation utilising a rule-based engine follows the "Extractive-Based Approach." This approach is a key feature of the web application designed for document summarisation, enabling the extraction of relevant content from the entirety of the data contained in the multiple uploaded documents.

In the extractive-based approach, the document summariser operates by extracting information based on several key factors. These factors include the sentence's importance, the frequency of sentence repetition, the distinctions among different sentences, and a comprehensive analysis to identify which content is the most pertinent, informative, and valuable. In essence, the extractive-based approach aims to condense extensive volumes of information derived from various user-uploaded documents into concise and relevant summaries. This process is designed to save time and effort, providing users with a streamlined means of obtaining essential information from their documents.

1.3 On the basis of domain dependency: Domain Oriented

In the context of this research paper, the approach to multi-document summarisation is oriented toward specific domains. This implies that the domain of the resulting summary will be inherently linked to the content of the uploaded documents themselves. The fundamental purpose of this domain-oriented approach is to facilitate users in comprehending and rectifying the information contained within the uploaded documents through concise and relevant summaries that pertain to the subject matter of the documents.

It is crucial to emphasise that the domain is intimately connected to the specified topic delineated within the uploaded documents. As an illustration, if the multiple documents contain information of the domains of medicine, research & law the multi-document summariser will generate summaries within the respective domains of medicine, research & law. This domain-oriented approach is poised to significantly enhance the usability and relevance of the summarisation process, aligning the summaries with the womanising content of the uploaded documents.

II. LITERATURE REVIEW

1) Extractive Text Summarisation
Extractive summariser is used for the extraction of the most repetitive words , it also decides that which sentence and points are more important depending upon their uses in the uploaded documents.

2) Term Frequency - Inverse Document Frequency (TFIDF) approach:-
TF-IDF is used to rectify the frequency of the words in the uploaded documents. The higher the frequency in the words the more chances to select the sentence and the words due to the question words occurrence.

3) Clustering based approach
The main purpose of the clustering approach is to determine the topics and the summerize the text from the multi documents uploaded in proper sequences and subtopics. The clustering approach forms the topics for the domain and forms sentences in proper format for the following topic.

4) Rule based Method - (Rule Engine)
In multi-document summarisation the rule based is used to apply rules to the extracted sentences from the uploaded documents. The rules are used to phrase the sentences in proper sequence and generation of the paragraphs in the required topic.

5) Domain-oriented Summarisation

The Domain-oriented Summarisation approach is applied for the output to be obtained in a particular domain of multi documents uploaded. The domain is the field to which the title is related. For example :- The domain of the multi document uploaded is the medical domain then the output summary will be in the medical domain only without any external domain application.

6) Applications of Multi-document Summarisers

The application of Multi-Document summarisation in current era is to summarise the data from the multiple documents to overcome the time and efforts of the user. As it is very time consuming to read , understand and then summarise the data from the multiple sources. It will be a time saving and advance technology for the researchers.

III. METHODOLOGY

A. Rule Engine

Rule engines, in the context of this study, operate through the establishment of a structured and intelligible framework for rule definition. These rules are conventionally formulated as "if-then" statements, wherein conditions undergo assessment, and corresponding actions are executed upon their satisfaction. Rule engines serve as a means to automate intricate decision-making processes, oversee workflows, and enforce organisational policies. Within the scope of this research, the rule engine is harnessed to mechanise the process of generating pertinent summaries from a collection of uploaded documents. The rule engine employs a rule-based approach for the extraction of sentences, paragraphs, or key phrases, with a focus on content relevance within the uploaded documents.

The central objective of the rule engine lies in the curation of significant and valuable information within the resulting summary. In the context of this research paper, the rule engine leverages cross-document coherence rules to discern connections among the concepts, events, and entities present in the provided documents. It also dynamically adjusts the length and structure of the summary in response to the complexity of the source material.

The incorporation of the rule engine into the summarisation process has contributed to its systematisation, flexibility, and adaptability in handling multi-document summarisation tasks. This paper delves into the utilisation of the rule engine to enhance the efficiency and effectiveness of multi-document summarisation by harnessing rule-based techniques and cross-document coherence rules to capture the essence of the source documents, thereby facilitating a more refined and coherent summarisation process.

In essence, the rule engine autonomously determines the most suitable arrangement of tasks, adapting to the specific requirements of the situation. This distinctive characteristic signifies a departure from the conventional linear sequence of operations, as it introduces a dynamic and rule-driven approach to task orchestration within the multi-document summarisation process.

This research underscores the significance of this innovative approach, which allows the rule engine to optimise the summarisation process by rearranging the order of Clustering, TF-IDF, Topic Modelling, and Cosine Similarity operations based on the inherent needs of the documents and the rules it employs. This adaptive and rule-centric methodology contributes to the rule engine's ability to enhance the efficiency and effectiveness of multi-document summarisation, offering a novel perspective in the field of automated document summarisation.

B. PDF Text Extraction using PyPDF2

1. Library: PyPDF2 is a Python library used for working with PDF files and it allows you to extract text and manipulate PDF documents.
2. Functionality: In the code provided, the PdfFileReader class from PyPDF2 is used to extract text from PDF files.
3. Here's how it works:

a. The PdfFileReader is initialised with the path to a PDF file using PdfFileReader(open(file_path, "rb")). The "rb" mode indicates that the file should be opened in binary read mode.

b. The purpose of the getNumPages() method is for determining the total number of pages in the PDF.

c. A loop is used to iterate through each page of the PDF with pdf.getPage(page). For each page, extractText() is called to extract the text content. The extracted text is then concatenated to create a complete text representation of the PDF.

Usage: This technique is essential for extracting text content from PDF documents, which may contain text, images, and various other elements. Extracted text can then be processed, analysed, and summarised as needed.

C. Text Extraction from DOC and DOCX using extract:

Library: textract is a Python library that simplifies the extraction of text from various document formats, including DOC and DOCX files.

Functionality: In the code, textract is used to extract text from Microsoft Word documents (both .doc and .docx). Here's how it works:

textract.process(file_path) is used to extract text from the specified document file. The file_path is the path to the document file you want to process.

The extracted text is further processed and decoded using UTF-8 encoding to ensure that it is in a usable text format.

Usage: This method is employed for extracting text from Microsoft Word documents, which are commonly used for various types of reports, articles, and documents. Extracting text from these formats allows the code to work with textual content from a wide range of sources, enabling summarisation and analysis.

D. Text Preprocessing

Text preprocessing is a critical step in natural language process and text analysis tasks. It involves cleaning and transforming raw text data to make it more suitable for further analysis, such as text classification, sentiment analysis, or information retrieval.

Here's a more detailed explanation of the mentioned text preprocessing techniques:

1. Lowercasing**:**
*Purpose***:** Convert all text to lowercase to ensure uniformity and reduce the impact of letter case on text analysis. This makes "Hello" and "hello" identical for analysis.

2. Removing Special Characters and Punctuation:
*Purpose***:** Eliminate special characters and punctuation marks, such as !@#$%^&*()_+[]{}|:;"'<>,.?/~, which may not contribute significantly in the sentence meaning. Removing them helps simplify the data and focuses on the core text.

3. Removing Numbers:
*Purpose***:** Exclude numerical digits from the text. This is useful for tasks where numerical values are not relevant, such as sentiment analysis or topic modelling.

4. Removing URLs and Email Addresses:
*Purpose***:** Omit web links (URLs) and email addresses, as they are typically not informative for many text analysis tasks. Removing them helps reduce noise in the data.

5. Handling Non-Breaking Spaces and Extra Spaces:
Purpose: Replace non-breaking spaces with regular spaces and remove extra spaces to standardise text formatting. This ensures that spaces are consistent and do not impact text analysis.

6. Removing HTML/JSON Tags:
*Purpose***:** Strip away HTML and JSON tags to obtain plain text from web pages or structured data. This is essential when dealing with text extracted from web content or JSON files.

7. Removing Non-Alphabetic Characters:
Purpose: Eliminate characters that are not letters of the alphabet, such as digits, symbols, or other non-alphabetic characters. This step helps to focus on the textual content.

8. Handling Emojis:
Purpose: Depending on the analysis task, you can choose to remove, replace, or retain emojis. Emojis can convey sentiment and add meaning to text, so handling them depends on your specific goals.

9. Handling Non-ASCII Characters:
Purpose: Remove or replace non-ASCII characters to handle text in different character encodings. This is important for working with diverse text data in various languages and scripts.

10. Removing Diacritics:
Purpose: Use the “unicodedata” library to remove accents and diacritical marks from characters. This simplifies the text and ensures that words with and without diacritics are treated the same way.

11. Handling Repeated Characters:
Purpose: Reduce repeated characters to their single occurrence to avoid skewing the analysis. For example, "coooool" becomes "cool."

12. Using Regular Expressions (re library):
Purpose: Regular expressions are powerful tools for pattern matching and text manipulation. It is used to find and replace specific patterns or characters in the text. This is especially helpful for complex text transformations.

IV. TOPIC MODELLING

A. Latent Dirichlet Allocation (LDA)

1. Purpose:
LDA is a probabilistic model used for topic modelling in text data. It assumes that documents are mixtures of topics, and topics are mixtures of words. LDA aims to uncover these underlying topics and their distributions within a given corpus of text.

2. How it works:
a) LDA overview the entire documents as the mixture of topics in which each and every topic is categorised by the selection of words and sentences.
b) It decides and assigns the words in such a way that it forms a proper paragraph of the domain of the multi documents uploaded.
c) The result is the topic set where each topic is the representation of the sentences and each sentence is the representation of the topic.

3. Applications: LDA is used in various NLP tasks, such as document clustering, information retrieval, content recommendation, and content summarisation.

B. Gensim

1. Purpose: Gensim is an open-source Python library designed for topic modelling and document similarity analysis. It provides efficient tools for working with large text corpora and applying LDA and other topic modelling algorithms.
2. Key Features:
a) Implementation of LDA: Gensim provides a user-friendly API for training LDA models on text data.
b) Scalability: It can handle large corpora efficiently, making it suitable for real-world applications.
c) Word Embeddings: Gensim also includes tools for training Word2Vec models for word embeddings.
d) Usage: Gensim is widely used in academia and industry for various NLP tasks, including document clustering, content recommendation, and information retrieval.

C. Typical Steps in Using LDA with Gensim

1. Preprocessing: Prepare your text data by tokenising, lowercasing, and removing stop words and unwanted characters. This ensures that the text data is clean and ready for modelling.
2. Creating a Dictionary and Corpus:
a. Use Gensim to create a dictionary that maps words to numerical IDs.
b. Create a corpus that represents each document as a bag-of-words, where the word IDs are associated with their frequency in the document.

3. Training the LDA Model:
a. Specify the number of topics you want to discover.

b. Train an LDA model on the prepared corpus, specifying the dictionary and the number of topics.
c. The LDA model will learn topic-word distributions and document-topic distributions.

4. Interpreting Topics:
a. Examine the top words associated with each topic to understand the themes it represents.
b. Analyse the document-topic distributions to see which topics are prevalent in each document.

5. Visualisation and Evaluation:
a. Visualise the results, often using tools like pyLDAvis or Matplotlib, to gain insights into topic distributions and relationships.
b. Evaluate the quality of topics using metrics like coherence scores.

V. TF-IDF ANALYSIS

A. Term Frequency-Inverse Document Frequency (TF-IDF) Analysis
1. Purpose: TF-IDF is the abbreviation of Term Frequency-Inverse Document Frequency, is a critical technique in NLP used to assess the useful words in a document relative to a collection of documents (corpus). It is employed to identify significant terms in a document while reducing the less number of repetitive words having less importance in the summary.

2. How it works:
a. Term Frequency (TF): It calculates the number of frequency of the occurrence of the sentences and the words in the multi documents uploaded. It gives the ranking scores to the frequency of the repetition.which means maximum the number of repetition of the sentences and words the higher will be the score of the sentence and will help to decide the importance of the sentence in the summarisation.
b. Inverse Document Frequency (IDF): IDF accounts for how common or rare a term is across the entire corpus. Rare terms are given higher scores, emphasising their uniqueness.
c. Combining TF and IDF: The TF and IDF values are multiplied to obtain the TF-IDF score for each term in a document. This score reflects the importance of the term in that document relative to the entire corpus.

3. Applications: TF-IDF is widely used in information retrieval, text mining, and document classification. It helps identify keywords, assess document similarity, and improve the performance of various NLP tasks.
B. TfidfVectorizer (scikit-learn)
1. Purpose: The TfidfVectorizer is a component of the scikit-learn library, a popular Python library for machine learning and data analysis. TfidfVectorizer is specifically designed to convert a collection of text documents into a matrix of TF-IDF features.
2. Key Features:
a. Document-Term Matrix: TfidfVectorizer takes a collection of text documents and transforms it into a document-term matrix, where each row represents a document, and each column represents a unique term.
b. Customisation: Users can customise various parameters, such as adjusting TF-IDF weightings, handling stop words, and specifying n-grams (word combinations).
c. Efficiency: The scikit-learn implementation of TfidfVectorizer is efficient and can handle large corpora with ease.
d. Usage: TfidfVectorizer is employed in NLP for tasks like text classification, clustering, and information retrieval. It transforms textual data into a numerical format that can be used as input for machine learning models.

C. Cosine Similarity (scikit-learn):

1. Purpose: Cosine similarity is a technique used to recognise the sentences which are similar from the multiple documents. From this cosine similarity the data collected will be more accurate and precise in the summary.
2. How it works:
a. For each pair of documents, their TF-IDF vectors are treated as vectors in a high-dimensional space.
b. The cosine of the angle between these vectors is computed. A smaller angle (cosine value closer to 1) indicates higher similarity, while a larger angle (cosine value closer to -1) suggests dissimilarity.
c. Cosine similarity values range from -1 (perfect dissimilarity) to 1 (perfect similarity).

3. Applications**:** Cosine similarity is widely used for tasks like document retrieval, recommendation systems, and clustering. It allows for the measurement of how closely documents are related based on their content.

VI. CLUSTERING

A. K-Means Clustering:
1. Purpose: K-Means clustering is a fundamental unsupervised machine learning algorithm used to group similar data points into clusters.On the other hand of text analysis, K-Means can be applied to group similar documents based on their content.

2. How it works:
a. Initialisation: K-Means begins by selecting a specified number of clusters (K) and initialising the cluster centroids randomly or using other methods.
b. Assignment: Each document is assigned to the nearest cluster depending on similarity metric, cosine similarity.
c. Update: The mean of the documents within each cluster is the recalculation of the clusters centroids.
d. Iteration: The assignment and update steps are repeated until convergence, where data points no longer change clusters significantly.
e. Result: The result is K clusters, each containing a group of documents with high similarity within the cluster.

3. Calculating the Number of Clusters: Selecting the optimal frequency of clusters (K) is a crucial step. Common techniques, such as the elbow method or using domain knowledge, can be used. In your context, you mentioned determining K based on the square root of the number of documents, but other approaches may also be suitable, depending on the specific use case.

4. Applications: K-Means clustering is widely used in document clustering, image segmentation, customer segmentation, and other unsupervised learning tasks. In text analysis, it can be employed to group similar documents, which is useful for information retrieval, recommendation systems, and content organisation.

B. NumPy:
1. Purpose: NumPy is a fundamental Python library for numerical and array operations. It is essential for various mathematical and statistical computations, making it valuable for tasks like clustering and data analysis.

2. Key Features:
a. N-Dimensional Arrays: NumPy provides support for N-dimensional arrays, allowing for efficient storage and manipulation of large datasets.
b. Mathematical Functions: NumPy offers a wide range of mathematical functions and operations, such as mean, maximum, minimum, and more.
c. Linear Algebra: It includes functions for linear algebra operations, making it a valuable tool for mathematical and statistical analysis.
d. Interoperability: NumPy is compatible with many other libraries and tools commonly used in the scientific and data analysis communities.

3. Usage: In the comparison of document clustering, NumPy can be used for various tasks, including calculating the similarity between documents (e.g., cosine similarity), determining cluster centroids, and performing numerical operations to assess cluster quality, such as calculating the maximum similarity within clusters. NumPy is particularly helpful when working with large datasets or when performing complex mathematical operations within the clustering process. It is often integrated with other libraries like scikit-learn for a comprehensive analysis of textual data.

VII. EXTRACTIVE SUMMARISATION

A. Extractive Summarisation
1. Purpose: Extractive summarisation is a technique used in natural language processing to create a summary of a longer document by selecting and extracting sentences or passages that are deemed the most important or representative of the content. Unlike abstractive summarisation, which generates summaries in a more human-like manner, extractive summarisation directly extracts and arranges existing sentences from the uploaded document.

2. How it works:
a. Sentence Scoring: In extractive summarisation, sentences within the source document are scored based on their relevance, importance, or informativeness. Various algorithms and features can also be used to determine the frequency.
b. Selection: The sentences with the highest frequency of repetition are selected for summarisation in the summary. These selected sentences are then arranged in a coherent manner to form the extractive summary.

c. Content Retention: Extractive summarisation aims to retain the most crucial information from the uploaded documents while maintaining the original wording.

3. Applications: Extractive summarisation is widely used for summarising long documents, news articles, research papers, and other text sources. It simplifies content consumption and provides users with concise overviews of lengthy documents.

B. LexRank Algorithm and LexRankSummarizer (sumy):

1. Purpose: LexRank is an extractive summarisation algorithm designed to rank sentences based on their similarity and importance. The LexRankSummarizer, implemented in the "sumy" library, leverages the LexRank algorithm to create extractive summaries.

2. How it works:

a. Graph-Based Approach**:** LexRank is a graph-based algorithm. It first constructs a similarity graph where nodes and edges. where nodes and edges are used to represent sentence and similarity between the sentences respectively.

b. Sentence Similarity**:** Sentence similarity is typically computed using techniques like cosine similarity, Jaccard similarity, or other measures.

c. Ranking Sentences: LexRank applies the PageRank algorithm, which was originally developed for ranking web pages, to rank the sentences in the graph. This ranking reflects the repetition and necessity of each sentence in the document.

d. Selection: Sentences with the highest LexRank scores are selected for inclusion in the extractive summary.

3. Applications: LexRank, and consequently the LexRankSummarizer, is particularly useful for summarising documents where sentence importance is determined by their relationships and similarities to other sentences in the document. It has applications in content summarisation, information retrieval, and document summarisation tasks.

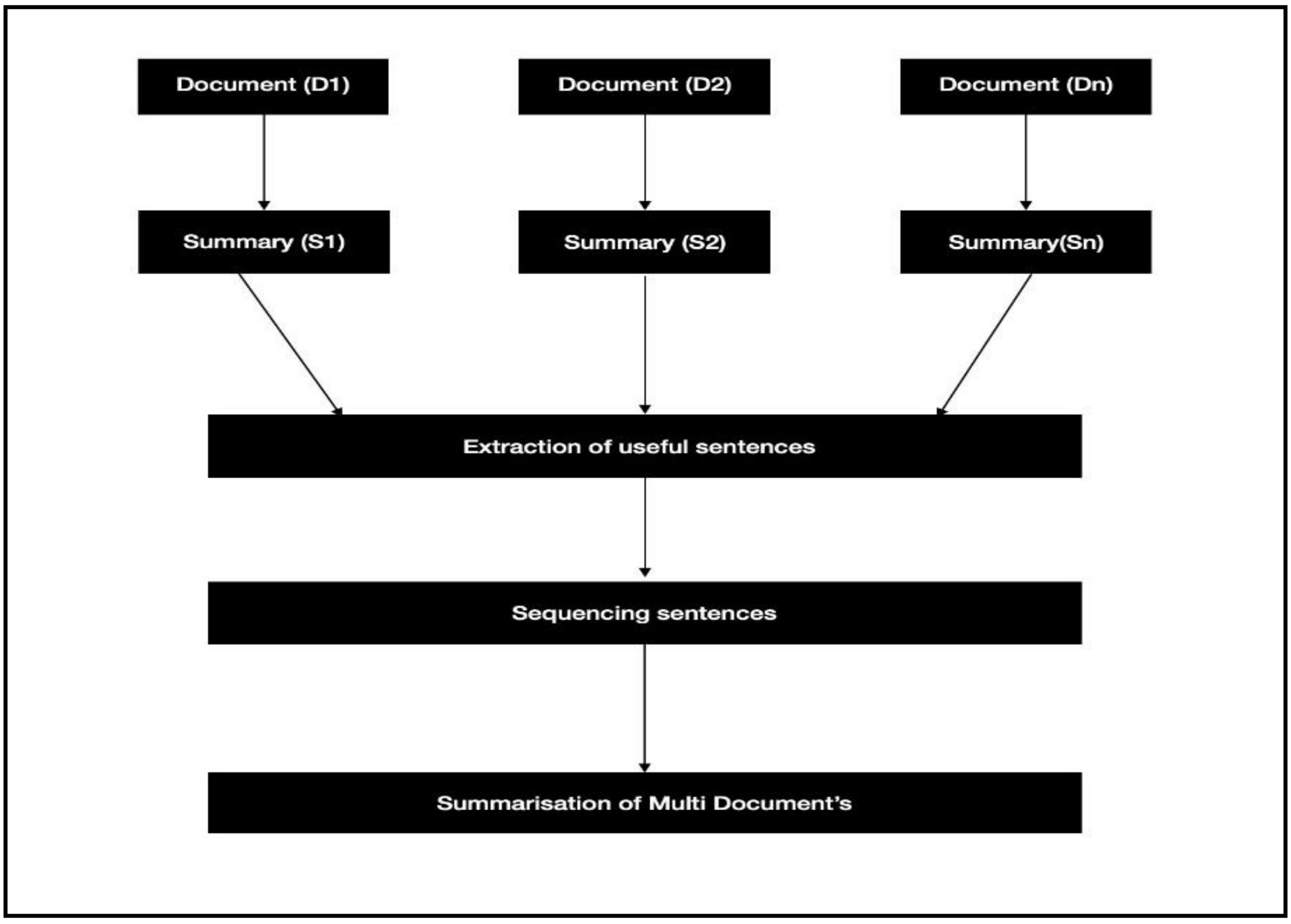


Fig. 2 Outline the sequence of steps involved in summarising multiple document

VIII. RESULT & ANALYSIS

A. Results

1. Precision and Relevance: Rule-based systems tend to provide precise summaries by focusing on specific rules. If the rules are well-defined and accurately capture relevant information, the summaries are likely to be highly relevant to the source documents.
2. Coverage: The coverage of the summarisation system refers to its ability to include important information from various parts of the input documents. A good rule engine should cover a wide range of topics and details from the source documents.
3. Redundancy: Redundancy in summaries occurs when similar or identical information is repeated. Careful rule design can minimise redundancy and ensure that the summary contains diverse and valuable content.
4. Coherence: Coherence assesses the logical flow and smooth transition between sentences in the summary. While extractive approaches preserve the original sentences, crafting rules that ensure coherent sentences in the obtained summary is crucial for readability.

B. Analysis

1. Rule Effectiveness: Evaluate the effectiveness of the rules in capturing relevant information. Analyse which rules contribute most significantly to the standards of the summaries and refine or modify them as needed.
2. Comparative Analysis: Compare the rule-based approach with other summarisation techniques, such as abstractive methods or machine learning-based models. Determine the advantages and disadvantages of the rule-based approach in different scenarios.
3. Scalability: Assess how well the rule engine scales with an increasing number of source documents. Evaluate its performance with small-scale and large-scale document sets to ensure efficiency and accuracy in different contexts.
4. Limitations: Identify the limitations of the rule-based approach, such as its inability to handle ambiguous language or complex contexts. Understanding these limitations can guide future improvements and research directions.

IX. FUTURE SCOPE

A. Enhancing Rule Engine Capabilities
Improving the rule engine itself is crucial. Researchers can work on creating more sophisticated and adaptable rule-based systems that can handle a wider range of document types, languages, and domains. These systems could employ machine learning techniques to automatically generate or refine rules.

B. Scalability
Making rule-based multi-document summarisation systems scalable to handle large volumes of documents is essential. Future research can focus on optimising performance and efficiency to process and summarise large datasets in real-time or near-real-time.

C. Customisation and User-Defined Rules
Allowing users to define their own rules or preferences for summarisation can enhance the usability of such systems. Future research can explore ways to make summarisation engines more customisable, so they can generate summaries tailored to specific user needs.

D. Hybrid Approaches
Combining rule-based summarisation with other techniques like extractive or abstractive methods, reinforcement learning, or deep learning can lead to more robust and effective summarisation systems.

X. CONCLUSION

The incorporation of a rule engine into the process of extractive multi-document summarisation has demonstrated its remarkable effectiveness and versatility in managing a wide array of intricate textual content. This rule engine functions independently of any predefined sequence, instead dynamically restructuring the order of operations in response to its own decision-making processes.

This unique attribute endows the rule engine with the capacity to autonomously determine the most appropriate arrangement of tasks, tailored to the specific demands of the situation.

Through the utilisation of engine rule technique and cross-document coherence rules, the rule engine has yielded substantial improvements in the efficiency and efficacy of multi-document summarisation. It excels in distilling the core essence of source documents, resulting in more polished and coherent summarisation.

In contrast to sequential methodologies such as Clustering, TF-IDF, Topic Modelling, and Cosine Similarity, the rule engine's adaptability in orchestrating the sequence of operations introduces a dynamic and rule-driven approach. This innovative approach optimises the summarisation process by reordering the operations based on the inherent requirements of the documents and the rules it employs. This flexibility proves particularly advantageous in scenarios characterised by significant variations in content complexity and structural composition.

REFERENCES

[1] R.S, Ramya, M. Shahina Parveen, Savitha Hiremath, Isha Pugalia, S.H. Manjula, and K.R. Venugopal. "A Survey on Automatic Text Summarization and its Techniques." International Journal of Intelligent Systems and Applications in Engineering IJISAE 11.1s (2023): 63–71.

[2] Zhang, Y., Ni, A., Mao, Z., Wu, C. H., Zhu, C., Deb, B., Awadallah, A. H., Radev, D., & Zhang, R. (2022). SUMMN: A Multi-Stage Summarization Framework for Long Input Dialogues and Documents. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Vol. 1: Long Papers, pp. 1592-1604). Association for Computational Linguistics.

[3] Joshi, A., Fidalgo, E., Alegre, E., & Fernández-Robles, L. (2019). SummCoder: An unsupervised framework for extractive text summarization based on deep auto-encoders. *Expert Systems with Applications*, 129, 200-215.

[4] Kurian, S., & Mathew, S. (2020). Survey of scientific document summarization methods. *Computer Science*, 21(2), 3356. https://doi.org/10.7494/csci.2020.21.2.3356.

[5] Satre, S. M., Patil, M., & Raju, S. (2019). Multi-Document Summarization using Fuzzy and Hierarchical Approach. *International Research Journal of Engineering and Technology (IRJET)*, 06(04), 2607. https://www.irjet.net/ISSN: 2395-0056.

[6] Agarwal, R., & Chatterjee, N. (2022). Improvements in Multi-Document Abstractive Summarization using Multi Sentence Compression with Word Graph and Node Alignment. *Expert Systems with Applications*, 190, 116154.

[7] Asa, A. S., Akter, S., Uddin, M. P., Hossain, M. D., Roy, S. K., & Afjal, M. I. (2017). A Comprehensive Survey on Extractive Text Summarization Techniques. *American Journal of Engineering Research (AJER)*, 6(1), 226-239. https://www.ajer.org/ISSN: 2320-0847 | p-ISSN: 2320-0936

[8] Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, Ł., & Shazeer, N. (2018). Generating Wikipedia by Summarizing Long Sequences. In *Proceedings of the International Conference on Learning Representations (ICLR)*, arXiv:1801.10198v1 [cs.CL].

[9] Tomer, M., & Kumar, M. (2021). Multi-document extractive text summarization based on firefly algorithm. Journal of King Saud University – Computer and Information Sciences, Advance online publication. https://doi.org/10.1016/j.jksuci.2021.04.004

[10] White, C. T., Molino, N. P., Yang, J. S., & Conroy, J. M. (2022). occams: A Text Summarization Package. Analytics, 2, 546–559. https://doi.org/10.3390/analy

[11] Afsharizadeh, M., Ebrahimpour-Komleh, H., Bagheri, A., & Chrupala, G. (2022). A Survey on Multi-document Summarization and Domain-Oriented Approaches. Journal of Information Systems and Telecommunication, 10(1), 68-79. DOI: https://doi.org/10.52547/jist.16245.10.37.68

[12] Koh, H. Y., Ju, J., Liu, M., & Pan, S. (2022). An Empirical Survey on Long Document Summarization: Datasets, Models and Metrics. arXiv preprint arXiv:2207.00939v1 [cs.CL].

[13] P, Keerthana. (2021). Automatic Text Summarization Using Deep Learning. EPRA International Journal of Multidisciplinary Research (IJMR), 7(4). https://doi.org/10.36713/epra2013

[14] Uckan, T., & Karci, A. (2020). Extractive Multi-Document Text Summarization Based on Graph Independent Sets. Egyptian Informatics Journal, 21(3), 145-157.

[15] White, C. T., Molino, N. P., Yang, J. S., & Conroy, J. M. (Year of publication). occams: A Text Summarization Package. *Analytics*, 2, 546-559. https://doi.org/10.3390/analytics2030030.

[16] Rao, P. R. K., & Devi, S. L. (2018). Enhancing Multi-Document Summarization Using Concepts. AU-KBC Research Centre, MIT Campus of Anna University, Chennai 600044, India. Email: t.pattabhi@gmail.com; sobha@au-kbc.org. Journal Name, Volume(Issue), Page range. [Published online: 10 March 2018]

[17] Prabhala, B. (2014). Scalable Multi-Document Summarization Using Natural Language Processing. Master's thesis, Rochester Institute of Technology, B. Thomas Golisano College of Computing and Information Sciences, Rochester, New York. Supervised by Dr. Rajendra K. Raj.