

Flowchart Generation and Mind Map Creation using Extracted Summarized Text

Aditya Kulkarni*

Dept. of Computer Engineering DJSCE
Mumbai, India
adityakulkarni244@gmail.com

Lynette D'Mello*

Dept. of Computer Engineering DJSCE
Mumbai, India
lynette.dmello@djsce.ac.in

Hetansh Shah*

Dept. of Computer Engineering DJSCE
Mumbai, India
hetanshshah28@gmail.com

Krish Shah*

Dept. of Computer Engineering DJSCE
Mumbai, India
krishshah981@gmail.com

Abstract—The technology discussed in this research study aims to transform text into a variety of visual representations, including mind maps, flowcharts, and summaries. The research underlines the usefulness of graphical representations and summaries for greater retention, starting with the challenges of remembering and digesting information from numerous sources. The project uses machine learning and deep learning algorithms to extract text from user-provided images or PDFs and transform it into mind maps or flowcharts based on user preferences. The capability of this research spans both the business and academic areas, allowing for the concise summarization of lengthy, complex documents. An overview of the literature on text extraction, summarization, and conversion techniques is given in the article. These techniques covered include OCR, network text analysis, deep learning-based summarization, keyword extraction, and data flow diagrams. The suggested method involves extracting text using OCR, summarising text using the T5 transformer model, creating flowcharts based on phrase relationships, and creating mind maps using NLTK and NetworkX modules. The conclusion of the study discusses the advantages and implementation specifics of each technique. The project offers a powerful way for converting text into visual representations in both academic and professional contexts, encouraging better comprehension, information retention, and improved learning and problem-solving skills.

Keywords—text extraction, text summarization, tesseract, NetworkX, flowchart conversion, TopicRank, NLTK.

I. INTRODUCTION

In the modern world, there are many resources accessible to an individual who wants to acquire knowledge in numerous domains, but it can be difficult to recollect every piece of data from different sources. People have been seen to remember some kind of pictorial representation over plain text in such situations. A summary also makes it simpler to remember lengthy, dull, and straightforward text. In order to suit the user's preferences, the technology converts the text it has retrieved from a picture or PDF into mind maps or flowcharts. Both the corporate sector and the educational sector might be included to the scope. A concise and accurate summary can replace a long and complex paper. A helpful tool for learning, information recording, illustrating connections between ideas

and facts, hastening memory recall, and cultivating creative problem-solving skills is mind mapping. Visual learning has also been shown to boost classroom learning by 400%. In the corporate world, mind maps that are simple to read help employees better understand lengthy agreements. Additionally, a summary of the company's rules and procedures that working professionals can readily recall can be created. To provide the desired outcomes, it applies machine learning and deep learning principles. This literature review on text extraction, summarization, and conversion into mind maps or flowcharts is included in section 2 of this paper. The proposed strategy for carrying out the concept is the primary focus of Section 3. The results and comments of the summarising and translation into mind maps or flowcharts are the subject matter of Section 4. Finally, part 6 provides a summary of the study and section 5 draws attention to the software's uses.

II. LITERATURE REVIEW

In this section, a literature review of the existing methods for text extraction, text summarization, flowchart conversion, mind mapping, and keyword extraction is discussed.

In order to determine if a given text (such as a news item) falls under the category of being related to a specific topic or not, the study by Livia et al. [4], provides a novel two-way classification strategy for network text analysis that integrates graph-based and text-based elements. The suggested method is tested against two datasets, and the findings demonstrate that it is more accurate at classifying objects than the standard approaches. The research has a flaw in that it only tests the suggested method using two datasets, which might not be representative of all circumstances.

A deep learning-based method for extracted text summarization is presented in the research study by Arun et al. [7]. The suggested approach uses neural networks to select and extract the most crucial sentences or phrases from a given document, producing a succinct summary that effectively summarises the vital details. In order to capture semantic and contextual information for efficient summarization, the study investigates alternative deep learning architectures, such as

Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs). The approach's effectiveness in producing relevant and coherent summaries from big textual datasets is shown through experimental findings, highlighting its potential for automated document summarization tasks.

The problem of information overload is exacerbated by the development of information technology, including social media applications and mobile Internet. Useful information can be filtered and found for users from large amounts of text thanks to keywords. Improved recommendations and keyword-based information retrieval are made possible by the automatic extraction of keywords from text for use as text tags. Zhenzhen et al. [9], have provided some study which suggests a novel method for extracting keywords from text that incorporates factors like association and word frequency. Results of the experiment reveal that TextRank and TF-IDF had lower precision rates, recall rates, and F-measure.

Sehrish et al. [13], incorporate a Data Flow Diagram (DFD) to model the data and functions in wide-ranging computer science applications. DFD was created utilising open-source software tools that give users access to a wide range of environments and shapes. The output's authenticity is still in question, and the current methods have never been successfully implemented. They also require a great deal of human labour. According to the characteristics of the desired system, the objective of the research is to create a partly automatic tool that can quickly construct complex Data Flow Diagrams. A Natural Language Interface (NLI), which enables users to construct questions and establish the capabilities and limitations of the system, was developed prior to the assembly of DFD. Keywords are culled from scraped content using Natural Language Processing (NLP) methods in order to build a data repository. In order to check for isomorphism, output DFDs had been turned into conceptual digraphs using adjacency and permutation matrices. The empirical findings show that the DFDs produced by the system are accurate, extensive, and noteworthy.

K. R. Singh et al. [14], give an overview of the most recent methods for text recognition from photographs in the paper proposed in her study. It covers a wide range of text recognition-related subjects, including as feature extraction, pre-processing methods, and classification algorithms. The study offers a thorough overview of the current status of the field and explores the advantages and disadvantages of various strategies. The accuracy of OCR on noisy or poor-quality photos is one area that must be improved in future studies.

III. PROPOSED METHODOLOGY

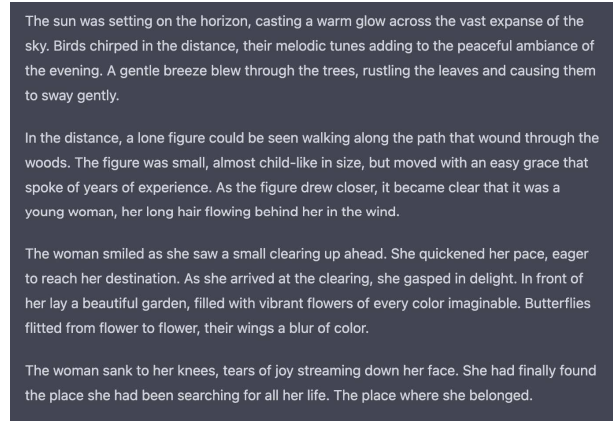
In this study work, a way has been supplied to transform the text into a summary or flowchart or thoughts maps. First, the text is extracted from the given input, then it's miles summarized and ultimately it's converted into the favored shape of visible illustration.

A. Text Extraction Using Tesseract

The approach of Optical Character Recognition (OCR) is utilised to show the text right into a virtual layout. The function makes use of a range of technologies and techniques to do

that. The "OpenCV" library is then used to examine it from the picture and convert it to RGB format. Reading, writing, and processing facilities for nevertheless pics and moving pix are to be had through the laptop vision library referred to as OpenCV.

The "Pytesseract" package is then used to OCR-extract the text from the image. The Tesseract OCR engine's Python wrapper, Pytesseract, makes it simple to do OCR on images. Overall, the function offers a quick and simple approach to converting an image to digital text format by utilising a number of well-known and practical Python modules. These technologies can be used with Python's text and image processing methods because of their many advantages, including their simplicity, speed, and accuracy.



The sun was setting on the horizon, casting a warm glow across the vast expanse of the sky. Birds chirped in the distance, their melodic tunes adding to the peaceful ambience of the evening. A gentle breeze blew through the trees, rustling the leaves and causing them to sway gently.

In the distance, a lone figure could be seen walking along the path that wound through the woods. The figure was small, almost child-like in size, but moved with an easy grace that spoke of years of experience. As the figure drew closer, it became clear that it was a young woman, her long hair flowing behind her in the wind.

The woman smiled as she saw a small clearing up ahead. She quickened her pace, eager to reach her destination. As she arrived at the clearing, she gasped in delight. In front of her lay a beautiful garden, filled with vibrant flowers of every color imaginable. Butterflies flitted from flower to flower, their wings a blur of color.

The woman sank to her knees, tears of joy streaming down her face. She had finally found the place she had been searching for all her life. The place where she belonged.

Fig. 1. Sample Text.

B. Text To Summary

The implemented technology creates a summary from the input text using the Hugging Face Library's T5 transformer model. It is a state-of-the-art language model that has been pre-trained on a substantial corpus of text using a variant of the Transformer architecture.

Utilising the pipeline function from the Hugging Face library, the code loads the T5 model for summarization. The summarization parameter specifies how the pipeline should be set up for text summarization. The model parameter specifies which pre-trained T5 model to use for summarization, while the tokenizer option specifies the tokenizer to use for the model. The framework argument (TensorFlow in this case) specifies the deep learning framework to be used.

Once the model has loaded, the summarizer pipeline receives the input text (body). The max length and min length parameters are used to specify the maximum and minimum lengths of the generated summary text. The do sample option is set to False in order to ensure that the model always generates the same summary text when the function is called. The summary text generated by the model is then output by the function.

The T5 model and the Hugging Face collection offer several advantages over traditional summarising techniques. The T5 model is a neural network-based approach that learns to build summaries based on a large amount of training data, resulting

in high-quality and coherent summaries. The Hugging Face library's user-friendly interface for pre-trained models like T5 may make it simpler to add contemporary natural language processing abilities into applications.

```
artificial intelligence (AI) has revolutionized various industries and aspects of
everyday life . it involves the use of algorithms and
data to enable machines to learn, reason, and make decisions
. ai has found applications in fields such as healthcare,
finance, transportation, and entertainment . there are also concerns regarding
ethics, privacy, and job displacement .
```

Fig. 2. Summarized Text.

C. Text To Flowchart Conversion

The function flowchartMaking creates a flowchart from the provided text. The program breaks the text down into separate sentences using natural language processing and machine learning. Then, based on the relationships between the sentences, edges are drawn between the sentences to represent the nodes in a graph. The result is then represented graphically as a flowchart.

The function makes use of a variety of technologies, such as:

Transformer Summarizer: It is a machine-learning version for textual content summarization that makes use of the GPT-2 transformer structure. The advantage of utilising this method is that it could produce high-quality summaries that accurately reflect the textual content's primary concepts.

PyDot: Graphviz's Python interface for graph visualization is known as PyDot. Graphs can be created and modified the usage of this library.

Matplotlib: A Python library used for records visualization is referred to as Matplotlib. It is used to show the flowchart picture on this function.

The advantages of adopting those technologies consist of the potential to summarize statistics effectively and accurately, create tricky graphs, and display the graph in a clear and comprehensible manner.

The steps involved within the characteristic's manner are as follows:

- 1) Use the GPT-2 model to summarize the furnished textual content.
- 2) Separate the material from the summary into separate sentences.
- 3) Make a graph with each sentence's nodes as its nodes.
- 4) Create a flowchart by way of becoming a member of the nodes in step with their connections.
- 5) Display and shop the generated flowchart using Matplotlib.

The flowchart Making function is a useful tool for locating connections between sentences in textual content and may be used for a number of obligations, along with report summaries, method modelling, and facts visualisation.

D. Text to Mind Maps

The method uses the Python Natural Language Toolkit (NLTK) and NetworkX modules to generate a thoughts map from a given paragraph of text. The code plays the following: Use the sent_tokenize() and word_tokenize() strategies from the NLTK package deal, to tokenize the enter prose into sentences and subsequently into words.

Use the functions set() and stopwords.words(), to eliminate stop words from the word list from the NLTK library and the module of punctuations from the string library.

Use a for loop and a list comprehension to construct a filtered list of words devoid of certain stop words and punctuation. Utilise the NLTK library's FreqDist() function to determine the word frequency distribution of the filtered words. Then, employ a list comprehension that looks for a minimum frequency of three occurrences to separate the key terms from the frequency distribution.

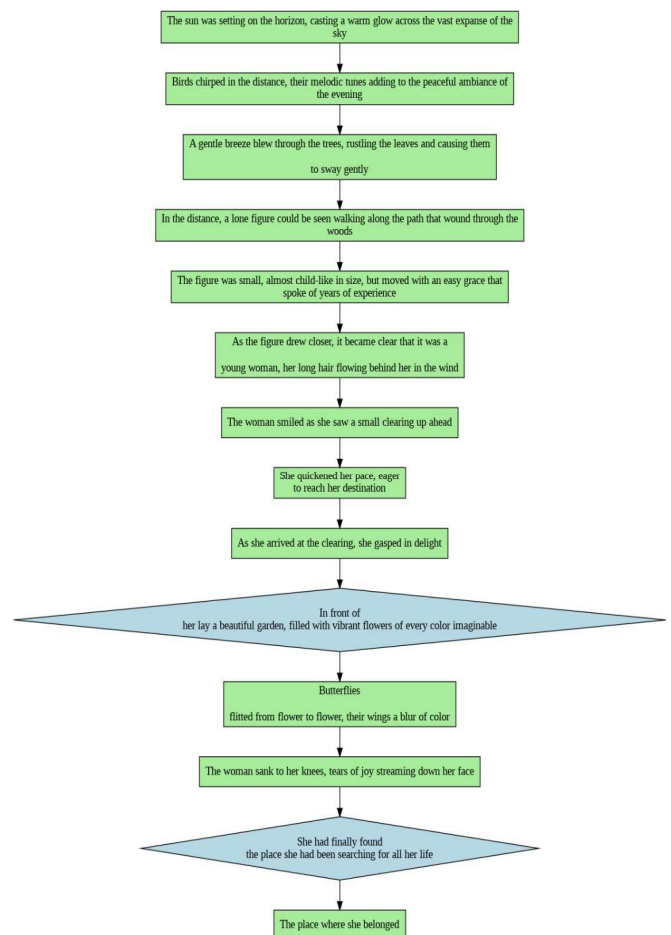


Fig. 3. Flowchart I

Use the filtered words as nodes and the relationships between them as edges, to construct the graph. The NetworkX library's Graph() function is used to initialise the graph. The filtered terms are then iterated over in each sentence using a nested for loop. The term is added to the graph as a node if it qualifies as an essential word. The words before and after the word are connected as edges with the label "follows" if it is not the first or last word in the phrase. A connection is made with

the word in the same position in the prior sentence with the label "related to" if the term is not in the first sentence. Finally, a connection is added to the word in the same location in the following sentence with the label "related to" if it is not in the last phrase.

In order to place the nodes in a way that reduces the overall edge length, the graph is drawn using the `spring_layout()` function from the NetworkX library. Then, using the `draw()` function, the graph is displayed with the node labels in a font size of 12 and a bold font weight. Additionally, the edge labels are added to the plot using the `get_edge_attributes()` and `draw_networkx_edge_labels()` functions from the NetworkX Library, with a font size of 8, a blue background colour, and a bold font weight. The `show()` function from the Matplotlib library is then used to display the plot with a figure size of 15x15 graph-based ranking and random walks to identify the most important terms in the content.

The limit, the maximum number of key phrases to return, and the body, the input text from which key phrases are to be extracted, are the two arguments needed for the function. The extractor object is initialised with the TopicRank algorithm after the input text has been loaded and preprocessed using the spacy library. While the candidate_selection technique chooses potential key phrase candidates based on noun and adjective sequences, the candidate_weighting approach employs a random walk algorithm to assign weights to the candidate phrases. The `get_n_best` method is then used to retrieve the maximum number of the top keywords.

One advantage of using the TopicRank algorithm for key phrase extraction is its ability to handle lengthy texts, support multi-word phrases, and use of graph-based ranking algorithms to identify the most important terms. The pke package offers a user-friendly interface for building various key phrase extraction models, including the ability to train supervised models on distinct datasets. In general, the feature provides a quick and simple method for locating and extracting significant keywords from text sources.

$$t_f(t, d) = \frac{f_d(t)}{\max_{w \in d} f_d(w)} \quad (1)$$

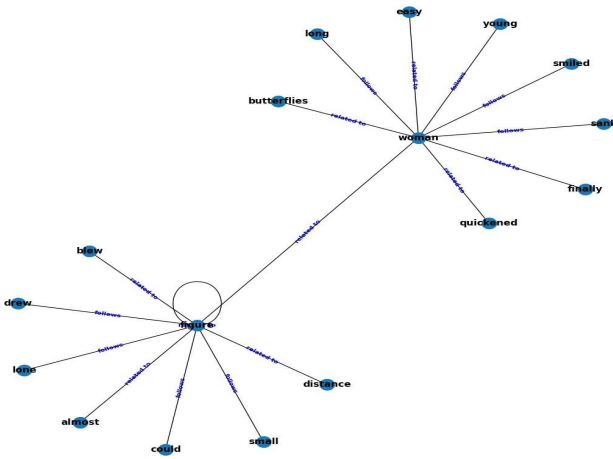


Fig. 4. Mind Map I

E. Keywords Extraction from Text

ExKeywords is a function that extracts important phrases from a given text document using the TopicRank algorithm implemented in the pke package. The free and open-source Python package pke offers keyphrase extraction models for both supervised and unsupervised keyphrase extraction. This function uses the unsupervised TopicRank method, which uses

$$\text{idf}(t, D) = \ln \frac{|D|}{|\{d \in D : t \in d\}|} \quad (2)$$

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D) \quad (3)$$

$$\text{tfidf}(t, d, D) = \frac{\text{idf}(t, D)}{|D|} + \text{tfidf}(t, d, D) \quad (4)$$

Where, $f_d(t)$ = frequency of term t in document, d D = corpus of documents

IV. Results and Discussion

The function uses the Hugging Face library's T5 transformer model to summarise text. The system creates succinct summaries of the input text by configuring the model with the pipeline function. The T5 model provides accurate and well-organized summaries because it is a neural network-based method that was trained on a sizable corpus. The user-friendly interface of the Hugging Face library improves the T5 model's accessibility and incorporation into apps.

The flowchartMaking function described in the paper makes use of a variety of tools, including Matplotlib, PyDot, and the TransformerSummarizer. It divides the input text into sentences and builds a flowchart by figuring out how the sentences relate to one another. The diagram created as a result of this process offers a graphical representation that makes the connections and text structure easier to comprehend. Data visualisation, process modelling, and document summaries are possible uses for this function.

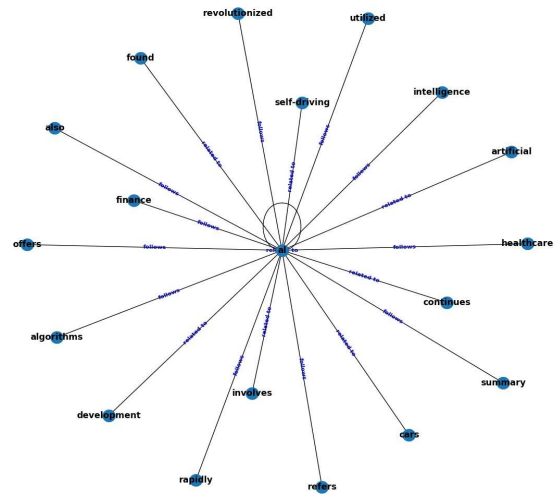


Fig. 5. Mind Map II

The NetworkX and Python NLTK modules are used in the suggested method for creating mind maps. A mind map is created by tokenizing the input text into sentences and words, removing stop words and punctuation, and creating a graph based on the relationships between words. The major ideas and relationships within the text are explored and understood more easily thanks to the mind map's visualisation using NetworkX and Matplotlib.

The paper introduces the exKeywords function, which extracts key phrases from text using the TopicRank algorithm from the pke library. This unsupervised technique uses graph-based ranking and random walks to rank key phrases. With support for long texts and multi-word phrases, the function offers a simple method for extracting significant keywords from the input text. The ability to create and train supervised models for key phrase extraction is another feature of the pke library.

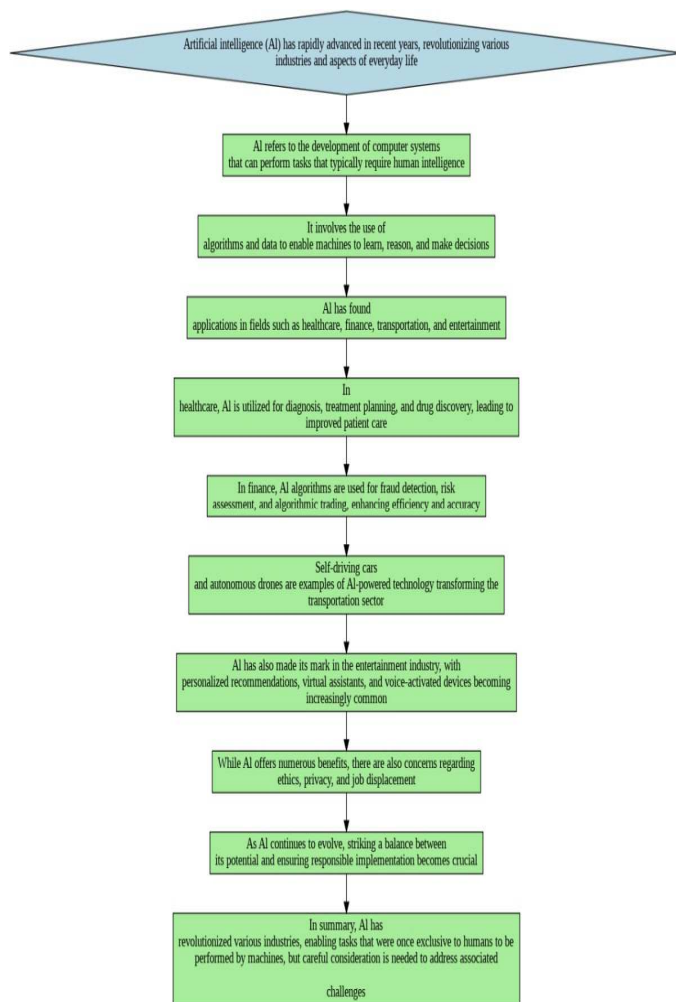


Fig. 6. Flow Chart II

The provided methodology demonstrates useful techniques for turning text into summaries, flowcharts, and mind maps overall. The system makes use of OCR, transformer models, graph-based algorithms, and visualisation libraries to provide useful tools for information processing, analysis, and comprehension. The suggested methodology has the benefits of being

straightforward, quick, accurate, and adaptable to different text sources. The system is suitable for a variety of natural language processing and data visualisation applications thanks to the integration of Hugging Face, NLTK, and pke, three user-friendly libraries that enhance accessibility and usability.

V. APPLICATIONS

There are many uses for text to mind map and text to flowchart conversion in educational and professional fields. Here are a few instances:

A. Education

Teachers and students may find it useful to take notes and organise thoughts by converting text to mind maps. By converting text to mind maps, which enable learners to perceive and connect multiple concepts, knowledge is retained and remembered by learners more readily. Concept maps are informational diagrams that help students understand and organise links between distinct concepts. Text to mind map conversion can be used to create concept maps. It is possible to promote active learning by using text to flowchart conversion to involve students in the process of diagramming and visually displaying their understanding of a subject. Research Tools Study tools can be used by students to assess and reinforce their subject understanding. You may create these study tools by turning text into mind maps and text into flowcharts.

B. Business Application

Workflows, business processes, and decision-making processes can all be modelled and managed by businesses using text to flowchart conversion. Text to flowchart translation can be used to create visual representations of project schedules, dependencies, and key routes, which makes managing difficult projects easier. You may create training materials and user manuals that take you through tasks and procedures step-by-step by converting text to flowcharts. Businesses can use text to mind map and text to flowchart conversion to present complex information in an alluring and visual way that will help stakeholders understand it and make decisions. Additionally, flowcharts and mind maps may be used to analyse and analyse the significant data that firms have obtained after conducting market research in a way that is clearer and more understandable.

Text to mind map and text to flowchart conversion can be useful tools in academic as well as professional settings, which will improve communication and comprehension of complex information.

VI. CONCLUSION

The technology developed as a result of the proposed study allows text to be transformed into visual representations such as mind maps, flowcharts, and summaries. The proposed approach employs a wide range of tools and techniques, such as optical character recognition (OCR) with Tesseract, the T5 transformer model for text summarization, the creation of flowcharts using machine learning and natural language processing, and mind maps using NLTK and NetworkX modules. Additionally, it is simpler to see key concepts and relationships within the text when using the suggested method for creating mind maps using NLTK and NetworkX modules. The detailed

and well-organized depiction provided by the mind maps helps with comprehension and memory retention. Overall, the suggested methodology offers practical and affordable ways to turn text into representations that are memorable and aesthetically beautiful. Utilising state-of-the-art technology and libraries enhances the approaches' accuracy and usefulness, making them valuable tools in a range of industries including education, corporate settings, and information processing. Future research can focus on enhancing and expanding these techniques in order to meet challenges and applications in the field of text analysis and visualization.

REFERENCES

- [1] Character Recognition using Machine Learning and Deep Learning - A Survey. (2020, March 1). IEEE Conference Publication — IEEE Xplore. <https://ieeexplore.ieee.org/document/9167649/keywords#keywords>
- [2] Liao, S., Yang, Z., Liao, Q., & Zheng, Z. (2023). TopicLPRank: a keyphrase extraction method based on improved TopicRank. *The Journal of Supercomputing*, 79(8), 9073–9092. <https://doi.org/10.1007/s11227-022-05022-0>
- [3] Yan, Y., Tan, Q., Xie, Q., Zeng, P., & Li, P. (2017). A graph-based approach of automatic keyphrase extraction. *Procedia Computer Science*, 107, 248–255. <https://doi.org/10.1016/j.procs.2017.03.087>
- [4] Celardo, L., & Everett, M. G. (2020). Network text analysis: A two-way classification approach. *International Journal of Information Management*, 51, 102009. <https://doi.org/10.1016/j.ijinfomgt.2019.09.005>
- [5] Korab, P. (2023, September 12). Text Network Analysis: Theory and Practice - towards Data science. Medium. <https://towardsdatascience.com/text-network-analysis-theory-and-practice-223ac81c5f07>
- [6] Pham, P., Nguyen, L. T. T., Pedrycz, W., & Vo, B. (2022). Deep learning, graph-based text representation and classification: a survey, perspectives and challenges. *Artificial Intelligence Review*, 56(6), 4893–4927. <https://doi.org/10.1007/s10462-022-10265-7>
- [7] Yadav, A. K., Singh, A. P., Dhiman, M., Vineet, Kaundal, R., Verma, A., & Yadav, D. (2022). Extractive text summarization using deep learning approach. *International Journal of Information Technology*, 14(5), 2407–2415. <https://doi.org/10.1007/s41870-022-00863-7>
- [8] Vilca, G. C. V., & Cabezedo, M. a. S. (2017). A study of abstractive summarization using semantic representations and discourse-level information. In *Lecture Notes in Computer Science* (pp. 482–490). https://doi.org/10.1007/978-3-319-64206-2_54
- [9] Xu, Z., & Zhang, J. (2021). Extracting Keywords from Texts based on Word Frequency and Association Features. *Procedia Computer Science*, 187, 77–82. <https://doi.org/10.1016/j.procs.2021.04.035>
- [10] Hu, M., Guo, H., Zhao, S., Gao, H., & Su, Z. (2021). Efficient Mind-Map generation via Sequence-to-Graph and reinforced graph refinement. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. <https://doi.org/10.18653/v1/2021.emnlp-main.641>
- [11] Meddeb, A., & Romdhane, L. (2022). Using topic modeling and word embedding for topic extraction in Twitter. *Procedia Computer Science*, 207, 790–799. <https://doi.org/10.1016/j.procs.2022.09.134>
- [12] Hou, J., Xie, L., & Zhang, S. (2022). Two-stage streaming keyword detection and localization with multi-scale depth-wise temporal convolution. *Neural Networks*, 150, 28–42. <https://doi.org/10.1016/j.neunet.2022.03.003>
- [13] Cheema, S. M., Tariq, S., & Pires, I. M. (2023). A natural language interface for the automatic generation of data flow diagrams using web extraction techniques. *Journal of King Saud University - Computer and Information Sciences*, 35(2), 626–640. <https://doi.org/10.1016/j.jksuci.2023.01.006>
- [14] Wang, B., & Giabbanelli, P. J. (2023b). Identifying informative features to evaluate student knowledge as causal maps. *International Journal of Artificial Intelligence in Education*. <https://doi.org/10.1007/s40593-023-00329-2>
- [15] Arai, K. (2023). Method for training and white boxing DL, BDT, Random Forest and mind maps based on GNN. *Applied Sciences*, 13(8), 4743. <https://doi.org/10.3390/app13084743>
- [16] Zaman, F., Shardlow, M., Hassan, S., Aljohani, N. R., & Nawaz, R. (2020). HTSS: A novel hybrid text summarisation and simplification architecture. *Information Processing and Management*, 57(6), 102351. <https://doi.org/10.1016/j.ipm.2020.102351>
- [17] Suleiman, D., & Awajan, A. (2020). Deep learning based abstractive text summarization: approaches, datasets, evaluation measures, and challenges. *Mathematical Problems in Engineering*, 2020, 1–29. <https://doi.org/10.1155/2020/9365340>