Global Transitions Proceedings 3 (2022) 2–7

Contents lists available at ScienceDirect

Global Transitions Proceedings

journal homepage: http://www.keaipublishing.com/en/journals/global-transitions-proceedings/

Summarization tool for multimedia data

Swarna Kadagadkai*, Malini Patil, Ashwini Nagathan, Abhinand Harish, Anoop MV

Department of Information Science & Engineering, JSS Academy of Technical Education, Bangalore 560062, India

ARTICLE INFO

Keywords:
Natural language processing
Optical character recognition
Rogue
Speech recognition
Text analysis
Term frequency-inverse document frequency

ABSTRACT

Text summarization is an important Natural Language Processing problem. Manual text summarization is a laborious and time-consuming task. Owing to the advancements in the field of Natural Language Processing, this task can be effectively moved from manual to automated text summarization. This paper proposes a model named Term Frequency-Inverse Document Frequency (TF-IDF) Summarization Tool which implements a text analytics approach called TF-IDF to generate a meaningful summary. TF-IDF is used to identify the topic or context of the text statistically. As data today is mostly unstructured in nature, this paper aims to explore a combination of NLP techniques such as Speech Recognition and Optical Character Recognition to summarize multimedia data as well. The TF-IDF Summarization Tool is seen to produce summaries with Jaccard's Similarity value of 67% and Rogue-1 of 64.9%, Rogue-2 of 48.2%, and Rogue-L of 56.4% based on a self-developed dataset.

1. Introduction

Summarization is the technique of condensing data to a form that is quick to peruse, easy to understand and, the technique of conveying the crux of the content whilst maintaining the grammar and semantics of the language [1]. The tool proposed in this paper is the Term Frequency-Inverse Document Frequency (TF-IDF) Summarization Tool which makes use of the Term Frequency-Inverse Document Frequency algorithm to summarize data. The data used for this tool is multimedia data which includes text (.txt, .pdf) and audio (.mp3, .wav, .m4a) files [2]. The pdf files are pre-processed using Optical Character Recognition (OCR) [[3],[4]] and the audio files are preprocessed using the Speech Recognition (SR) module of python [5]. Post-pre-processing, the TF-IDF algorithm is applied for summarization [6]. The tool is built to coherently carry out OCR, SR, and text summarization seamlessly [7]. The tool also allows users to record live audio online and summarize this audio content in real-time [8]. To achieve this, python's Speech Recognition module is used. Finally, the summary is returned in the form of a string [9].

Optical Character Recognition is the process of identifying text from scanned images or documents [10]. If the user uploads a pdf file to our tool, the file is first converted into an array of images using the popular pdf2image python module . The pdf2image module is a wrapper around pdftoppm and pdftocairo command-line tools to convert PDF to a PIL Image list [11]. This image list is passed to the OCR engine. Each image is scanned for readable text data. A combination of pattern recognition and feature detection is used to identify letters, therefore, words from the image [12]. The text identified is returned in the form of a string.

Speech Recognition is the interdisciplinary field that makes use of certain technologies to recognize spoken language and to translate it into text [[13], [14]]. The audio file uploaded by the user is first pre-processed to be transcribed. Most of the popular Speech Recognition modules make use of the Hidden Markov Models to recognize speech [15]. Python's Speech Recognition module is a package that acts as a wrapper around popular speech recognition APIs. This module returns the recognized speech in the form of a string [16]. Text Summarization is a common Natural Language Processing problem and several proposed solutions that produce coherent and fluent summaries for the given text are present [17]. Text can be summarized in four ways - Single Document Summary, Multi-Document Summary, Query Focused Summary, and Informative Summary. The proposed tool successfully carries out Multi-Document Summary. Text Summarization is categorized as extractive and abstractive text summarization.

Extractive text summarization is an automatic text summarization process that involves identifying the most important sentences from the text and classifying them as the summary. The first step in extractive text summarization is representing the topic of the content in the text or the indicators that define the features of the sentences in the text. To represent the topic of the content, various approaches are used such as Latent Semantic Analysis, Bayesian Models, Frequency Based Approach, and Topic Word Approach [18]. The Frequency-based approach involves determining the importance of the word based on its frequency in the text. Word probability and TF-IDF are two approaches used here.

The Topic word approach is the process of evaluating the frequency of each word, comparing the frequency against a frequency threshold, and extrapolating if the word can successfully represent the topic of the

* Corresponding author.
E-mail address: swarnakadagadkai1@gmail.com (S. Kadagadkai).

https://doi.org/10.1016/j.gltp.2022.04.001

Available online 2 April 2022
2666-285X/© 2022 The Authors. Publishing Services by Elsevier B.V. on behalf of KeAi Communications Co. Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

given text. These words are called topic words. The sentence is granted importance based on the frequency of these topic words in the sentence.

Indicator representation involves ranking the importance of a sentence based on its features rather than the frequency of the words. To achieve this, graph-based methods and machine learning models can be used. The graph-based approach is built on top of the Page Rank algorithm. The sentences are considered to be the nodes of a graph and based on the measure of similarity between any two sentences, an edge is constructed between the two nodes representing these sentences. These edges are also assigned weights based on this measure of similarity. Thus, the sentence importance metric is determined.

In the machine learning approach, text summarization is modeled as a classification problem. Based on the features of the sentences, the model classifies the sentence as a summary or a non-summary sentence.

Abstractive text summarization is constructed using deep learning and it involves using LSTMs and attention models to generate a summary [19]. This type of text summarization involves manipulating the semantics of the text to maintain consistency and to make the summary discernible. It recognizes the context of the given text, retains the content while re-constructing the text in a simplified manner. This area of research is still progressing towards increasing the accuracy and precision of the machine learning models used to generate abstractive summaries.

The proposed tool has devised a solution using the TF-IDF Approach and it consists of tokenization, Parts of Speech (PoS) Tagging, and lemmatization.

Tokenization is essentially breaking down a long sentence or phrase into smaller units called tokens. These tokens can be words, characters, or even punctuation marks based on the requirement. In-text summarization, it is important to understand the parts of the text that are words and the parts that are special characters. The proposed tool uses the Natural Language Toolkit (NLTK) of python to tokenize the text into words and sentences using the built-in word_tokenize() and sent_tokenize() functions of NLTK.

PoS tagging is the process of classifying the words of a sentence as parts of speech. It determines if the word is a noun, verb, adjective, adverb, punctuation, or a number. In text summarization, nouns, and verbs improve the accuracy and precision of the summary generated. Thus, this process is a crucial stage in text summarization. The NLTK module in python uses stop words to categorize the words of a sentence as parts of speech.

Lemmatization generates the root word from the derived word in the given text. Stemming is also another way of achieving this. However, the root word generated by lemmatization is an actual word, whereas the one generated by stemming need not be. Understanding the meaning of the text is an important process of text summarization, thus, lemmatization is used. The natural language toolkit in python allows us to lemmatize the words of a sentence.

2. Related work

Information on the internet is scaling at an exorbitant rate and according to studies conducted, people consume more content in one hour now than 100 years ago. Due to the volume of information available and the rate of its consumption, the time spent on understanding, reading, and consuming it must be reduced. This led to text summarization being a significant problem in Natural Language Processing.

Rani, R., et al [20] define extractive text summarization as the process of producing a summary that is the result of extracting the vital portions of the original text. In this type of summary, the vital portion of the original text is identified by utilizing various text analytics techniques thereby including it in the final summary. The paper talks about topic modeling to summarize the text. It also discusses single document and multi-document text summarization and novel document summarization.

Shi, T., et al [21] employs abstractive summarization by implementing RNN seq2seq models to carry out text summarization. The paper states why abstractive text summarization is gaining traction. It is corroborated by describing how abstractive text summarization uses novel words in the summary, is verbally innovative, and can easily incorporate external knowledge.

Text Summarization can be carried out using various techniques. The ones described in this section are, the graph-based approach, LSTM-NN, K-Means clustering, and Bidirectional Encoder Representations from Transformers (BERT) model, Text Rank algorithm, and Term Frequency-Inverse Document Frequency (TF-IDF).

Belwal., et al [22] describe a new graph-based extractive summarization model using keywords or topic modeling. In the graph-based approach, the sentences are represented as nodes, and a relationship is established among the nodes using edges. The important step here is to assign a weight to each edge. The weight assigned to the edge is determined by a similarity measure parameter. Then, the text Rank algorithm is employed where nodes are assigned a rank. Depending on the desired length of the summary, the top most ranks and the associated nodes are chosen to produce the final summary.

Mutlu, B., et al [23] make use of LSTM-NN models for sentence selection in summarization based on three parameters, semantic features, syntactic features, and ensembled features. The model stated here consisted of two LSTMs in the first layer that simultaneously process the semantic and syntactic feature spaces. The outputs of these two LSTMs were concatenated in the next layer, and finally, the resulting enhanced feature space was given to a 2-layer neural network to carry out classification. The resulting summaries were evaluated using the ROUGE metric.

Miller, D utilizes the K-Means algorithm and the BERT model to carry out extractive text summarization. The source text is tokenized into clean sentences and fed as input to the BERT model to generate embeddings. The embeddings are then clustered using K-Means and the embeddings closest to the centroid of the cluster are selected for a summary.

Rani, U., et al [24] describe the TF-IDF method to carry out text summarization. In this paper, TF-IDF is compared with the Text Rank algorithm approach and the LDA approach. TF-IDF is a statistical approach that compares the frequency of a given word in a given document with the inverse proportion frequency of this word in other documents. The term frequency (TF) score and the Inverse Document Frequency (IDF) score is calculated separately first and then coalesced to generate one final score that ranks all the sentences in the document. Based on the desired length of the summary, the top most sentences are selected.

3. Methods and models

This section highlights the objectives of the experiment, describes the process of preparing the dataset and elucidates the algorithm employed.

3.1. Objectives

The objective of this experiment is to summarize text, pdf, and audio files and generate the summary in a text form and to

1. reduce the manual computational time needed to summarize large volumes of data
2. increase the speed and accuracy of the TF-IDF model used to generate summary
3. allow the user to upload a raw text (.txt, .pdf) file, an audio (.mp3, .m4a, .wav) file, or to record audio online
4. ensure that the context, grammar, and semantics of the original document/text are maintained.

Table 1
Input and Output Specifications.

Sr. No	Input/Output	Specifications
1.	Input	A text (.txt, .pdf) file or an audio (.mp3, .wav, .m4a) file, or audio recorded online using the user system's microphone.
2.	Output	Summary of the given content in the form of a string.

Table 2
Results of the TF-IDF Summarization Tool.

Sr. No.	Metrics	Value (%)
1	Jaccard's Similarity	67
2	Word Error Rate	34
3	Compression Ratio	45
4	Rogue-1 (Recall, Precision, F-Measure)	63, 60, 56
5	Rogue-2 (Recall, Precision, F-Measure)	43, 50, 47
6	Rogue-L (Recall, Precision, F-Measure)	63, 59, 56
7	Rogue-W-1.2 (Recall, Precision, F-Measure)	37, 77, 50
8	Rogue-Su4 (Recall, Precision, F-Measure)	49, 66, 62

3.2. Dataset preparation

The dataset used for this experiment is in the form of multimedia data viz., audio, and text (.pdf, .txt) files. The dataset required has to be a collection of text (.pdf, .txt) files and audio (.mp3, .wav, .m4a) files. To meet the aforementioned objective and due to the paucity of data in the required form, the dataset was curated especially. This enabled us to build a tool that would improve user experience in terms of the data used. Since the TF-IDF algorithm does not involve training or testing, the data can be simply uploaded in its raw form. The entire collection sums up to 150 files used as input files.

3.2.1. Text data preparation

The text files in the dataset are in .txt and .pdf formats. The files are in the standard English language. To summarize the text of a different language, a language translator needs to be employed. The collection of text files was downloaded from corpusdata.org. The collection provided by this website included text content from ~95,000 websites. Three different formats of the data are present in this collection – relational databases, word/lemma/Parts of Speech (PoS), and words (paragraph format). This experiment made use of the format of the words and is not pre-processed or cleaned in any manner. A text identifier (textID) is present for each paragraph which signifies from which website the text has been taken. The textID is followed by the linear text. The words of the linear text are not annotated for word, lemma, or PoS. The dataset used for this experiment contained 80 text files in total.

3.2.2. Audio data preparation

The audio files in the dataset are of the .mp3, .m4a and .wav form. The collection of audio files was downloaded from LibriSpeech ASR corpus. It consists of 1000 hours of 16kHz read English speech. The audio content is taken from various audiobooks. All the files present in this collection are of the form .mp3. To extend the experiment to other formats of audio files, 600 seconds of 16kHz read English speech in the .m4a and .wav form were manually downloaded. The collection of audio files is not pre-processed or cleaned in any manner. The dataset used for this experiment contained 70 audio files each of length 30 seconds in total.

3.3. Methodology

The original file is given as input by the user and its type is checked. The length of the summary is decided by the user and is referred to as the Retention Ratio (RR). Along with the provision of uploading a text or an audio file, the proposed tool allows users to record audio online and get the summary generated in real-time. The audio is recorded using the user's system's microphone and it is transcribed to text using python's Speech Recognition module. The Speech Recognition module generates the transcript for this audio content and returns it as a string. This transcript along with RR is passed as parameters to the text summarization function. If the file type is a text file with a .txt, it is passed as a parameter along with RR to the text summarization function. If the file type is a pdf file, it is passed as a parameter to the OCR function. The OCR function passes the pdf file as a parameter to a python library called pdf2image. This library converts each page of the pdf file into an image and returns an array of images. The array of images is passed to the easyocr reader object and the text content is extracted into a text file and is passed as a parameter to the text summarization function along with RR. If the file type is an audio file with a .mp3, .wav, or .m4a extension, the file is passed as a parameter to python's Speech Recognition module and the module returns the transcript of the audio content in the form of a text file. This text file along with RR is passed as parameters to the text summarization functions. The flowchart representing the methodology is depicted in Fig. 1.

3.4. Algorithm

The text summarization function has an algorithm as follows. The input and output requirements for this algorithm are given in Table 1.

1 Upload input file. Determine the type of input file. If it is a text file, go to Step 2. If it is an audio file, go to Step 3.
2 Determine the type of text file. If it is in .txt format, go to Step 4. If it is in .pdf format, go to Step 5.
3 The audio file is passed to the speech_recognition() function which uses python's Speech Recognition model. This model returns the transcript of the audio file and writes it to a .txt file. The .txt file is passed as a parameter to the text_summarization() function. Go to step 6.
4 The .txt file is passed as a parameter to the text_summarization() function. Go to Step 6.
5 The .pdf file is passed to a pdf2image library which converts each page of the .pdf file to an image. The image is passed to the ocr() function, which uses the easyocr reader object to scan the image for text. It returns the text present in the image as a string. The string is then written to a .txt file. This .txt file is passed to the text_summarization() function.
6 The text_summarization() function takes two parameters – Retention Ratio (length of the summary) and a .txt file with the given text. This function is constructed to accept the source text in a text file. Therefore, both speech_recognition() and the ocr() function return the text in a text file.
7 The content of the .txt file is tokenized into sentences using the sent_tokenize() function and into words using the word_tokenize() function.
8 The word_tokenize() tokenizes the sentences into words and returns an array of tokens.
9 The array of word tokens is scanned for stop words and is ousted from the array. The remainder of the array is passed to the word_net_lemmatizer() function
10 The word_net_lemmatizer() calculates the root word from each derived word and the frequency of these root words are stored as key-value pairs in a dictionary.

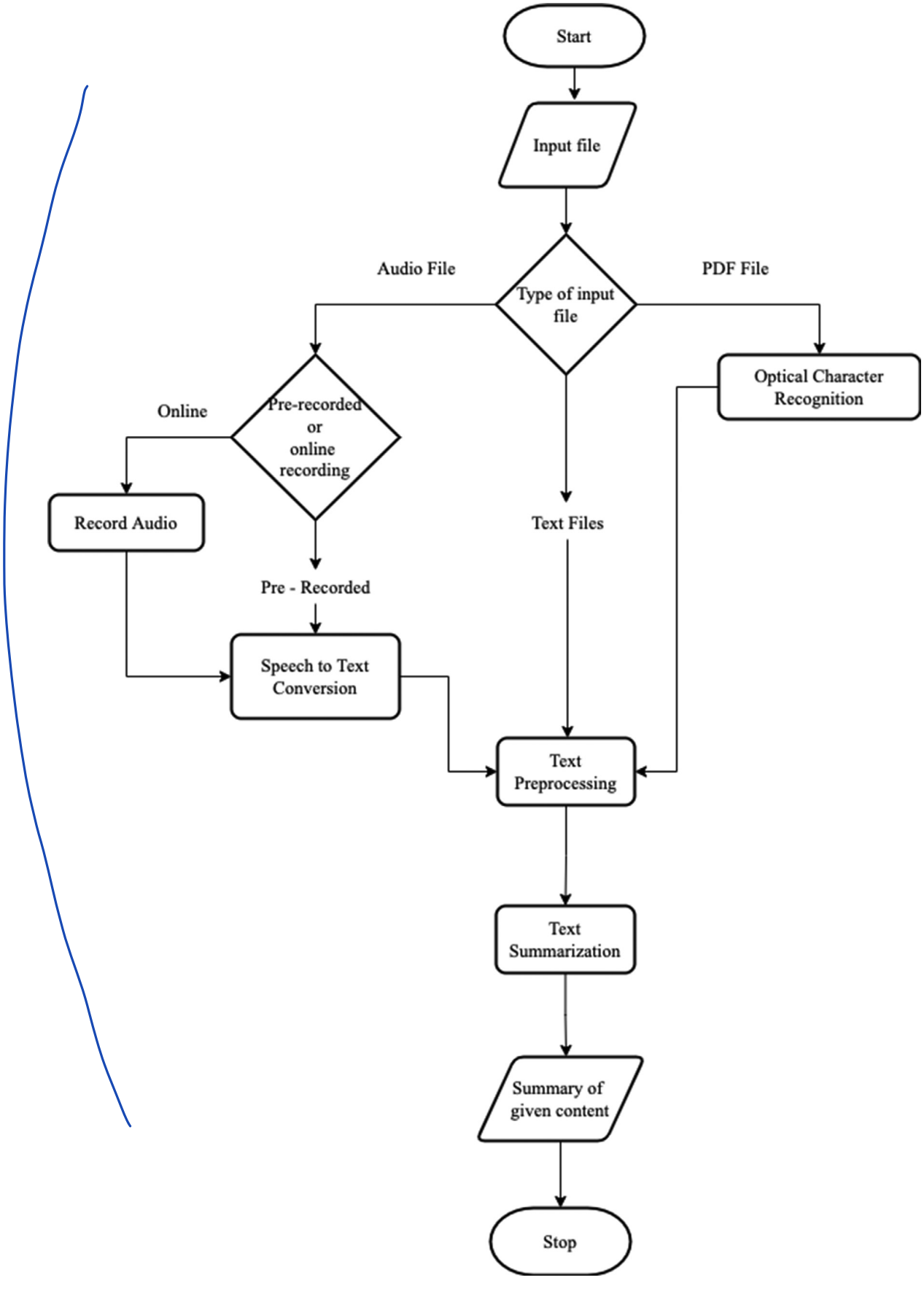

Fig. 1. Flowchart representing the methodology of the TF-IDF Summarization Tool.

Table 3
Comparison of the TF-IDF Summarization Tool with the State of Art.

Sr. No	State of Art	Metrics (%)		
		Rogue-L	Rogue-1	Rogue-2
1.	Text Summarization using Abstract Meaning Representations	54.7	62	48.6
2.	Analyzing the capabilities of crowdsourcing services for text summarization	-	42	11.1
3.	Automatic Text Summarization of Legal Cases: A Hybrid Approach	33.50	27.88	5.82
4.	SUMMN: A Multi-Stage Summarization Framework for Long Input Dialogues and Documents	51.39	53.44	20.30
5.	Multi-document extraction-based Summarization	-	-	33
6.	**TF-IDF Summarization tool**	**56.4**	**64.9**	**48.2**

11 Each sentence token returned by the sent_tokenize() function is checked for special characters and is scored for importance by the sentence_importance() function.
12 The sentence_importance() function removes all special characters and performs Parts of Speech tagging using NLTK's pos_tag() function.
13 The pos_tag() function recognizes all the nouns and the verbs in the sentence after verifying if it is a stop word or not. The experiment requires us to identify only nouns and verbs and categorize them.
14 The array of words returned by the pos_tag() function is lemmatized and then passed to the tf_idf() function.
15 The tf_idf() function calculates the tf score using the tf_score() function and the idf score using the idf_score() function for each word returned by the pos_tag() function. The product of these two scores is returned by the tf_idf() function.
16 The tf_score() calculates the score using the frequency of the word in the sentence divided by the total number of words in the sentence.
17 The idf_score() calculates the score using the total number of sentences in the document divided by the number of sentences containing that term and evaluating this value to log10.
18 The tf_idf() returns the product of the tf_score() and the idf_score() as the score for sentence importance and each sentence token is assigned a score.
19 Based on the Retention Ratio, the top most sentences are selected in the descending order of the highest scores.

4. Results and discussion

A summarization system can be tested in two ways using: qualitative measures and quantitative measures. Qualitative measures are carried out by manually comparing the summary generated by a model with the summary generated by human experts. This is a laborious task. Therefore, this paper makes use of quantitative measures like Rogue, Jaccard's Similarity, Word Error Rate, and Compression Ratio. The results are depicted in Table 2.

This paper also compares the proposed TF-IDF Summarization Tool with the state of artwork by using the aforementioned metrics. The results are depicted in Table 3.

4.1. Jaccard's similarity

The Jaccard similarity index is used to compare elements of two sets to determine shared and distinct elements. Here, the elements are machine-generated summary and human-generated summary. It's a tool used to measure similarity between these two elements, within the 0\% to 100\% scale. The higher the percentage, the more similar the two elements are. The formula to calculate this is given in equation (1).

$$Jaccard\ Similarity\ =\ |X \cap Y|/|X \cup Y| * 100 \tag{1}$$

Here, X = machine generate summary, Y = human generated summary, $X \cap Y$ = common sentences in the two elements and $X \cup Y$ = sentences in either element.

4.2. Word Error Rate

Word Error Rate (WER) is a metric for measuring the accuracy of automatic speech recognition (ASR) systems. It measures accuracy based on the ability of the system to convert speech to text. The formula to calculate this is given in equation (2).

$$WER = \frac{S + I + D}{N} \tag{2}$$

Here, S = the number of substitutions, I = the number of insertions, D = the number of deletions, and N = the number of words in the reference.

4.3. Compression ratio

Compression Ratio (CR) is a metric that compares the length of the summary with the length of the original text. It acts as a catalyst in producing an effective summary in less time and with the least redundancy. The formula to calculate this is given in equation (3).

$$C = \frac{S}{T} \tag{3}$$

Here, S = length of the Summary and T = length of the original text.

4.4. Rogue

Rogue stands for Recall Oriented Understudy for Gisting Evaluation and it is a set of evaluation metrics. It can be used to evaluate any data that is machine translated and that includes machine-generated summaries. It compares the machine-generated summary with a reference summary. The reference summary is usually generated by a human expert. In the conducted experiment, Rogue-N (for N=1 and 2), Rogue-L, Rogue-W-1.2 (Rogure_W_weight = 1.2), and Rogue-Su4 metrics were used. The Precision is calculated as shown in equation (4), Recall as shown in (5), and F-Measure as shown in equation (6) for these metrics.

$$\text{Pr}\,ecision = total\_now/total\_words\_machine \tag{4}$$

Here, total_now = total number of overlapping words between machine-generated and the reference summary and total_words_machine = total number of words in the machine-generated summary

$$Recall = total\_now/total\_words\_ref \tag{5}$$

Here, total_now = total number of overlapping words between machine-generated and the reference summary and total_words_ref = total number of words in the reference summary

$$F - Measure = \frac{\left(1 + \beta^{2}\right) R * P}{R + \beta^{2} * P} \tag{6}$$

Here, β = configuration parameter, the default value is 1.0 which gives equal weightage to precision and recall in the equation, R = Recall, and P = Precision

5. Conclusion

Due to the profound availability of data, text summarization is a vital problem statement in the world of computation and technology. Currently, this problem statement is one of the hot areas of research and attracts a lot of attention from different fields. To mitigate this problem, this paper proposes the TF-IDF Summarization Tool as a model to reduce users' time and effort. This model was able to efficiently produce a summary of varying lengths for text and audio files. Compared to most summarization tools in the related work, the proposed system can generate quick and meaningful summaries with relative ease for audio files using a combination of speech recognition and text summarization and for PDF files using a combination of OCR and text summarization.

6. Future scope

The system can be extended to accommodate features such as including other file extensions for audio files such as .mpeg, .aac, .flac, including handwriting analysis and summarizing handwritten text, and construct the model based on the abstractive approach using LSTMs and attention models.

Acknowledgments

The authors of this paper would like to thank JSS Academy of Technical Education for providing this opportunity.

References

[1] Minakshi Tomer, Manoj Kumar, Multi-document extractive text summarization based on firefly algorithm, J. King Saud Univ. Comput. Inform. Sci. (2021).
[2] T.N. Nguyen, N.P. Nguyen, C. Savaglio, Y. Zhang, B. Dumba, The Role of Artificial Intelligence (AI) in Healthcare Data Analytics, Int. J. Artif. Intell. Tools 30 (06 N 08) (2021).
[3] Zhang, Y., Ni, A., Mao, Z., Wu, C. H., Zhu, C., Deb, B., ... & Zhang, R. (2021). Summˆ N: A Multi-Stage Summarization Framework for Long Input Dialogues and Documents. arXiv preprint arXiv:2110.10150.
[4] B.D. Parameshachari, R. Banu, P. Rashmi, C.R. Rachana, B.R. Rao, Security mechanism for image authentication based on pixels adaption, Digi. Image Process. 8 (5) (2016) 146–149.
[5] J.N. Madhuri, R.G. Kumar, Extractive text summarization using sentence ranking, in: 2019 International Conference on Data Science and Communication (IconDSC), IEEE, 2019, March, pp. 1–3.
[6] M.K. Chowdary, T.N. Nguyen, D.J. Hemanth, Deep learning-based facial emotion recognition for human–computer interaction applications, Neur. Comput. Applica. (2021) 1–18.
[7] R. Elbarougy, G. Behery, A. El Khatib, Extractive Arabic text summarization using modified PageRank algorithm, Egypt. Inform. J. 21 (2) (2020) 73–81.
[8] N. Kumari, P. Singh, Automated hindi text summarization using Tf-Idf and textrank algorithm, J. Criti. Rev. 7 (17) (2020) 2547–2555.
[9] P. Kiran, B.D. Parameshachari, J. Yashwanth, K.N. Bharath, Offline signature recognition using image processing techniques and back propagation neuron network system, SN Comput. Sci. 2 (3) (2021) 1–8.
[10] C. Shah, A. Jivani, An automatic text summarization on Naive Bayes classifier using latent semantic analysis, in: In Data, Engineering and Applications, Springer, Singapore, 2019, pp. 171–180.
[11] A.P. Widyassari, S. Rustad, G.F. Shidik, E. Noersasongko, A. Syukur, A. Affandy, Review of automatic text summarization techniques & methods, Journal of King Saud University-Computer and Information Sciences, 2020.
[12] K. Yu, X. Qi, T. Sato, Z. Wen, Y. Katsuyama, K., ... Tokuda, T. Sato, Design and performance evaluation of an ai-based w-band suspicious object detection system for moving persons in the iot paradigm, IEEE Access 8 (2020) 81378–81393.
[13] R. Rani, D.K. Lobiyal, A weighted word embedding based approach for extractive text summarization, Expert Syst. Appl. 186 (2021) 115867.
[14] S. Madhu, B.D. Parameshachari, N. Joe, H.S. DivakaraMurthy, Color retinal image analysis for automated detection and severity of exudates, Int. J. Adv. Res. Comput. Sci. 4 (10) (2013).
[15] S. Thapa, S. Adhikari, S. Mishra, Review of text summarization in Indian regional languages, in: Proceedings of 3rd International Conference on Computing Informatics and Networks, Springer, Singapore, 2021, pp. 23–32.
[16] J. Zhang, K. Yu, Z. Wen, X. Qi, A.K. Paul, 3D reconstruction for motion blurred images using deep learning-based intelligent systems, CMC-Comput. Mater. Cont. 66 (2) (2021) 2087–2104.
[17] K.U. Manjari, S. Rousha, D. Sumanth, J.S. Devi, Extractive Text Summarization from Web pages using Selenium and TF-IDF algorithm, in: 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI, IEEE, 2020, June, pp. 648–652. (48184).
[18] F. Ding, K. Yu, Z. Gu, X. Li, Y. Shi, Perceptual enhancement for autonomous vehicles: restoring visually degraded images for context prediction via adversarial training, IEEE Trans. Intell. Transp. Syst. (2021).
[19] Liu, Y., & Lapata, M. (2019). Text summarization with pretrained encoders. arXiv preprint arXiv:1908.08345.
[20] R. Rani, D.K. Lobiyal, An extractive text summarization approach using tagged-LDA based topic modeling, Multim. Tool. Applica. 80 (3) (2021) 3275–3305.
[21] T. Shi, Y. Keneshloo, N. Ramakrishnan, C.K. Reddy, Neural abstractive text summarization with sequence-to-sequence models, ACM Transact. Data Sci. 2 (1) (2021) 1–37.
[22] R.C. Belwal, S. Rai, A. Gupta, A new graph-based extractive text summarization using keywords or topic modeling, J. Amb. Intell. Hum. Comput. 12 (10) (2021) 8975–8990.
[23] B. Mutlu, E.A. Sezer, M.A. Akcayol, Candidate sentence selection for extractive text summarization, Inform. Process. Manag. 57 (6) (2020) 102359.
[24] U. Rani, K. Bidhan, Comparative assessment of extractive summarization: textrank tf-idf and lda, J. Sci. Res. 65 (1) (2021) 304–311.