IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

AUTOMATED NLP BASED SEARCH SYSTEM FOR FILES

[1]Duddu Sree Harsha, [2]Aditya Raj, [3]Abdulla Yusuf Khan, [4]Harshit Singh, [5]Chaitra YR

[1]Student, [2]Student, [3]Student, [4]Student, [5]Assistant Professor
[1]Department of Computer Science and Engineering
[1]Dayananda Sagar Academy of Technology and Management, Bengaluru, Karnataka, India

Abstract: AutoNBS is a search system aiming to retrieve documents from a storage device based on a user's input and his/her search history. We have File Browser at the domestic scale and Search Engines for Corporate and Internet Applications to search for documents. Basic filters based on file type, size, and date of modifications are the minimalistic features supported while browsing files, but while dealing with a large number of files, the former technique falls short of retrieving the desired documents. Search engines, on the other hand, vary based on the domain for which they have been implemented. For example, the Evidence Management System used by Indian Police have been specifically designed to handle CCTV footage and store any type of pre-verified files as evidence, irrespective of their digital readability and value for analysis purposes. We propose a search solution that has a generalized implementation for any domain. Files of any format (text, image, audio) can be stored automatically after being verified for duplication and sufficing the basic quality requirements. Based on the user's input and search history, a bias will be generated to retrieve files that are deemed the most relevant as per the user’s expectations.

***Index Terms* - Indexing, Keyword Extraction, NLP, Search engine, Evidence management.**

I. INTRODUCTION

In this digital era, we all have felt the need to get reliable information based on our expertise at the right time. Applying filters and sorting documents using the file manager of our local devices seemed to be working fine. But with the ever-increasing generation of data and complex contents, the former approach is not enough. We aim to create a system capable of automatically assessing what documents to retrieve based on the input keywords and the past search done by the user & automating the storage of new files based on their distinctness and reliability. While existing information management systems provide data handling features, their functionalities are limited. There remains a high risk of document redundancy which can, in the future, lead to anomalies.

There has been an explosive increase in the amount of data in use compared to the last decade. There has been a further increase in the number of documents. So rather than indexing according to documents having which keywords, we can use inverted indexing that maps the keyword with the document’s id (single or multiple) where that keyword is present. An inverted index is an easy-to-use and effective technique for searching text, pictures, and multimedia files. An inverted index allows us to do a fast text search and can also be applied widely, like in search engines.

We are also using NLP techniques, wherein a computer tries to understand the natural language used by humans. NLP majorly involves the conversion of speech data into text or vice versa. The audio files are converted to text using a speech-to-text module. A model known as Vosk is used to carry out the above-listed task. While doing so, we get varied accuracies. The audio file is divided into chunks of data with thirty seconds duration, and each of these chunks is then processed to achieve higher precision and consistency.

In our fast-paced lifestyle, the need to automate the function of valuable data retrieval is not just a luxury but a commodity. For example, in the police evidence management system, there is a large number of documents exists that has been spanning for a long time. They contain a large number of multimedia files with different formats. If a person wants to access a particular document, there must be a system that helps in the retrieval of relevant documents which takes as less time as possible. Our proposed method can stand out in this field, as it will be able to provide access to the desired documents in a fast retrieval time. They contain the evidence files in an unstructured way, so manual filtering and accessing a particular file may become difficult. So, there comes a need to organize it in a structured way to decrease access time and automate such processes. This system involves various stages like Input, Parsing, Keyword generation, Indexing, and Search. We take data of different types like text, images, and audio. After keyword extraction using various tools and libraries, we index them, storing their document identifier, and ranking scores. According to the keywords provided as search parameters, the possible documents are listed and sorted with their relevancy. A bias is introduced to show relevant files according to the user’s search history. The bias ensures that pattern identified in the user’s search pattern is given priority. This also allows for the effect similar to fuzzy search by begin forgiving to mistakes in user's search parameter.

About 2.5 quintillion bytes of data were created by people every day in the year 2021. More than 80% of the above data generated is unstructured, making it challenging to analyze and process. Keyword extraction becomes quite handy for finding the terms that are relevant to the document. Keywords are unique, relevant, and semantically vital words in a document and give valuable insights. So, automating the process of selecting keywords is essential.

II. Literature Survey

The following is the analysis of the work done on the topics concerning the methodology of our proposed solution. The study on the existing models for multimedia-to-text extraction, keyword identification in different document types, indexing, and searching approach undertaken by fellow researchers will be discussed here.

Kumar, Anubhav [1] presented an effective method for extracting text from images using the Gabor filter. The proposed methodology involves text extraction using the Gabor filter method for text identification in complex images. This technique has three steps. Edge detection uses the Gabor filter transform properties of edges to find edge maps from all images in different orientations. The localization process uses the Morphology Open Binary process to transform the binary image into an augmented image. For text recognition, existing OCR systems require an input image from which characters can be analyzed and recognized. Experimental results show that this method can achieve a recall rate of 99.11% and a precision rate of 94.67%. Given the complexity of these images, the algorithm performs well, suggesting that such techniques may result to be useful in obtaining information from complex images.

Vinnarasu A. and Deepa V. Jose [2] describe an efficient method for speech recognition in their research. The speech is converted to the corresponding summarized text. This white paper uses the simple logic of adding a '?' to the end of the statements having interrogative semantics when there is a sizable gap in the audio signal. Otherwise, a '.' is added to mark the end of the sentence. The proposed research work also reduces the time and effort associated with manually documenting lengthy speech at events. Speech recognition and text summarization facilitate documentation tasks. This model can be used when a long lecture needs to be condensed into a concise document as an automated system converts the speech to text and summarizes the content. Since, the audio model can't handle large text, segmentation along with introduction of semantics would be beneficial for text extraction.

Piskorski, J., Stefanovitch N., Jacquet G., & Podavini, A. [3] presented a comparative study on different keyword extraction algorithms based on various techniques including KPMiner, Total Keyword Frequency, TF-IDF, RAKE, YAKE, KeyBert, and variants of TextRank-based algorithms. They applied these methods to keyword extraction from news articles in different languages and calculated the scores. They prioritized robustness, multilingualism, and efficiency in their keyword selection criteria. In their analysis, they appointed two human experts who were tasked with selecting keywords from the news articles for each language. They then tabulated and analyzed the data and observed the differences in annotators' inconsistencies across the languages. Furthermore, on applying those algorithms with some predefined parameters, their performance across languages is evaluated for exact, partial, and fuzzy matching. It was found that two algorithms had the highest score, YAKE, and KPMiner. Also, some hybrid combinations of YAKE variants and KPMiner also gave high scores. It was concluded that YAKE-KPMINER-R gave the highest F1 score of 20.1%, 46.6%, and 47.2% for the exact, partial, and fuzzy matching respectively. To further improve YAKE-KPMINER-R an additional deduplication step was added which gave only a small gain.

Luburić, N., Ivanović, D. [4] in their work made a comparative study between Solr and Elasticsearch, two well-known open-source search engines or information retrieval systems in terms of their efficiency, ease of use, functioning, speed, pros, and cons. Since both systems use Apache Lucene as their base library, they act similarly and provide nearly identical features, functionalities, and performance. Both are written in Java and provide a REST-like interface. Both technologies are very easy to get started with. But in terms of the query language, scalability, deployment, and some other functionalities, they are very different. Solr offers great information retrieval capabilities, but Elasticsearch is much easier to deploy and scale in production. Both systems are viable options when it comes to document indexing and searching. Choosing one among them depends on the requirements and use case.

Abhishek Das and Ankit Jain [5] discuss the evolution of the indexing carried out for the world wide web over the years. From the inception of the first full-text search engine to the present real-time social search engines. They present the improvements in understanding the user intent from term-based indexes to phrase-based indexes. They illustrate the memory and storage requirements for the same balancing cost to retrieval time using inverted indexes. The infrastructure required for the same with the ever-growing world wide web is predicted. Design trade-offs are mentioned to minimize resource utilization without affecting the results too much to continue scaling. The use of distributed file systems to better utilize the index giving priority to the high-impact terms first using partitions. The best data structures for the same are discussed also covering the re-ordering of the index for faster retrieval. For future work, they have proposed a real-time social graph to cater to the exponentially increasing user interaction with the world wide web. Support for personalized search with newer natural language processing algorithms is mentioned to enrich social context for query resolution.

Wang, Yang & Gong, Yeyun & Huang, and Xuanjing [6] in their work have designed a model to determine key phrases and rank keywords based on context information. The data acquisition from tweets on Twitter is justified as it has length limitations on the tweets. Existing models suffer from such shorter text while performing keyphrase extraction. They discuss the use of a Recurring Neural Network to overcome this issue for the benefit of processing sequential data. The use of different types of RNNs and their performance in comparison is done to determine the best architecture for the hidden layers of the RNN. The various semantic extraction techniques are illustrated for automating the task to support training the RNN for higher accuracy. The joint processing of keyword ranking and key phrase extraction worked well on the join-layer RNN indicating the best results for the tweets.

Devlin, Jacob & Chang, Ming-Wei & Lee, Kenton & Toutanova, and Kristina [7] introduce a new language representation model BERT that can be utilized for a multitude of tasks from query resolving and understanding language context. It has pushed the General Language Understanding Evaluation (GLUE) benchmark to 80.5% (7.7% improvement). The pre-training has proven effective for several natural language processing tasks. Pre-training with untagged data and fine-tuning with tagged data are discussed extensively in this framework. BERT is identified to have unified architecture across different tasks which simplifies the build process for different tasks. The benefits of the feature-based approach are observed for computation. The benefits of pre-computing an expensive representation to overcome the difficulty of representation of some tasks using Transformer encoder architecture are shown. Their major contribution is the generalization of these bi-directional architectures allowing further improvements across tasks.

Dario Amodei, Rishita Anubhai, and Eric Battenberg [8] have designed an end-to-end Deep Learning RNN model for speech recognition for varying languages aiming to achieve a performance comparable to the recognition ability of a human. The convolution layer, bidirectional recurrent layer, and fully connected layer constitute the model while implementing the CTC loss function. Using HPC techniques, they witnessed a 7x speedup from their previous model. A total of 11,940 hours of labeled data are loaded to train the model for the English language. This data may contain direct speech or conversational speech. Noise, the speaker's tone, and voice clarity are influential factors for the transcription of the input audio. Long audio files are segmented, augmented, and scaled to increase the robustness to noise and for faster processing of the data. There is a 43% reduction in Word Error Rate in English Speech from their previous model. In contrast to its predecessor, its WER is equivalent or closely comparable to a human's performance for recognizing speech in different environments.

Gerritse, Emma & Hasibi, Faegheh & de Vries, Arjen. [9] in their work, have discussed how bias has both positive and negative impacts on the functioning of a conversational search system. A Personalized Knowledge Graph (PKG) is a suitable method for storing user information in a structured form and personalizing results per their ranking. Information can be Search History, most used documents, etc. Time is a vital factor for getting consistent, and personalized results consistently. Bias can be of three types- Bias from an algorithm, from people, and from an algorithm and people. The third form can be used either for accelerating personalization or for detecting a biased user and to help in diversifying the result. It has been observed that neural models, especially those embedded with BERT, rank documents with gender bias. Strategies have been developed to mitigate bias from severely impacting a system's functionality. User in Control is a UI based technique where a they are given access to view either diversified results or a biased result. The reflective method to mitigate the negative impact of bias suggests alternative search results for the biased result. Measuring user satisfaction is of utmost importance in order to adjust the bias effect on the system. This paper gives an insight into how bias for personalizing search can have impacts on the overall user experience, and how to manage its negative impact based on user decisions.

III. METHODOLOGY

Every person ever has needed to retrieve the most relevant information from some data store be it physical, mental, or digital. Minimizing the time for retrieval allows us to concentrate more on analyzing and extracting value from the information rather than spending more time finding the right one. The advances in the natural language processing field allow us to automate the process of language understanding, allowing us to extract value from relatively unstructured data like audio clips of conversations, scanned documents, or text in otherwise unrelated images.

However, considering the limitations in reliability after a certain noise level in these data sources we must establish a benchmark requirement for ingestion into the system. Thus, images that are too blurred or audio clips with too much background noise must be discarded. This is because having no answer is better than relying on wrong insights. The proposed system caters to all domains and adapts to the need of the end user identifying the user's patterns and re-allocating priority. To this effect, the system will be available and value-making for all walks of people from individuals to large organizations. To ensure proper keyword extraction from text originating from images and audio which will lack semantics relative to regular text documents, we must train a purpose build RNN. To this end to create an RNN biased toward our needs, we must build a labeled dataset adhering to our requirements for keyword extraction. Scraping the web for short text and associated keywords is a good way to achieve this. Post training, the results of the RNN must be evaluated after training to determine its usage reliability.

The indexing phase must optimize storage space, read speed, and write speed. For general-purpose usage, the write speed can be slower than the read speed. The overall retrieval time to determine the most relevant file must be under 10 seconds. The use of distributed computing frameworks allows scaling as the Index grows for users which is directly proportional to the data stored. The concept of auto-sharding in the search engine Solr allows us for independent and isolated usage of the index partitions. The REST-like API provided by Solr must be easy to use keeping in mind all user interface design best practices. For the integrity of the research, the test data will be diverse enough to validate the above-stated functionalities. The building-block approach of each module allows us to alter individuals' modules completely without affecting the others. For example, if a better Keyword extraction and ranking algorithm are made available then it can be swapped out for the existing one even in an already running system by simply normalizing the same with the current models.

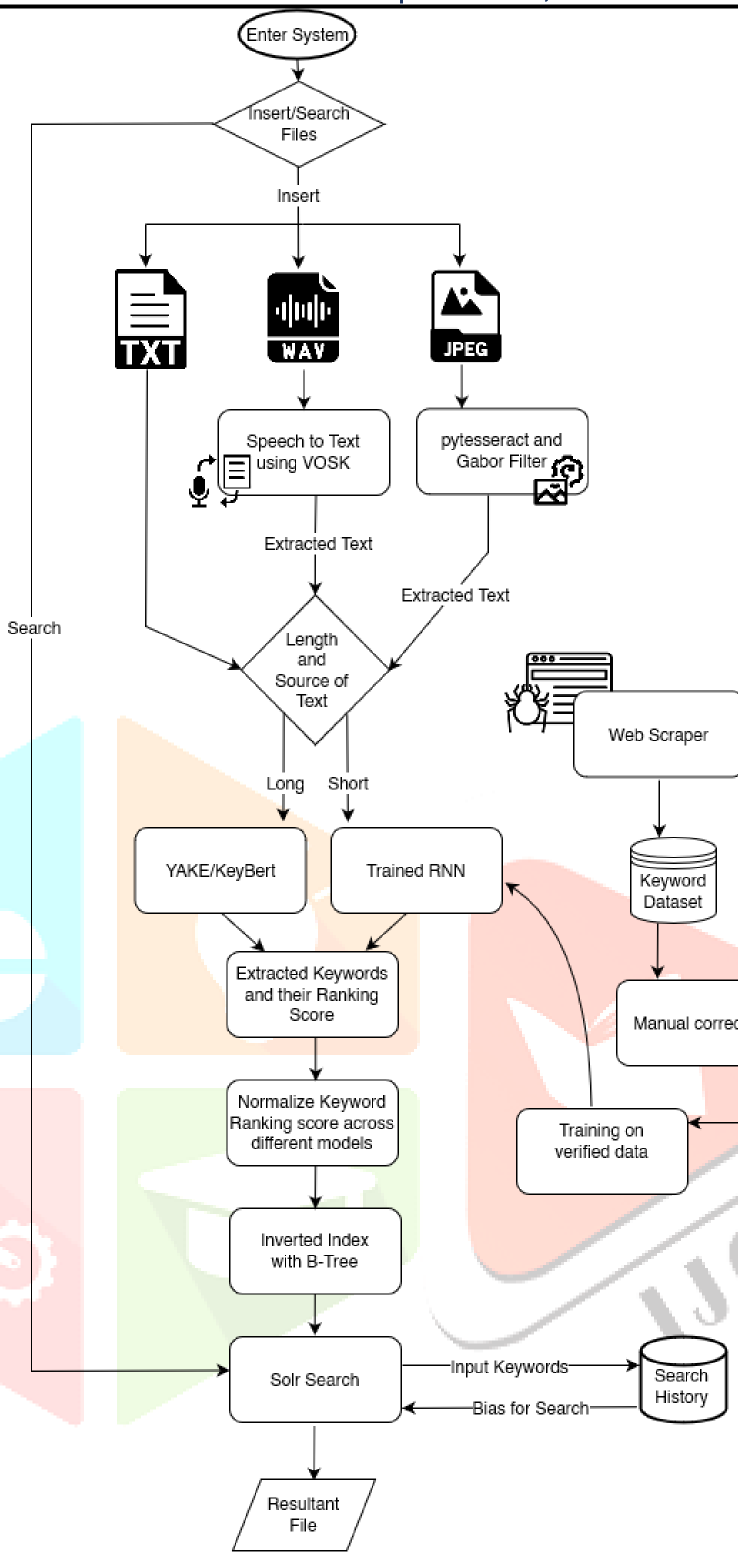

Fig-1: Proposed Flow of Application

3.1 Input Media Processing

In the first phase of the proposed system which is the input pre-processing phase, all types of files such as text, image, and audio are taken from the user using a single upload button. After this when the actual file is procured by the system, the contents of the file are compared with the existing file system for any matches using the filecmp() function of python which is widely used in file handling, it compares the given file with a directory which contains many files, if the system finds that there are any matches then the new file is discarded, and if it happens the other way around i.e. if there is no match of contents with the existing file system, then the format of the file is checked using a specialized regular expression, which splits the filename into two parts first one being the name of the file and second one, the file extension. When the file extension is obtained it becomes easier to classify to which group the uploaded file belongs i.e., either a text file or an image file, or an audio file. Upon getting the exact file format of the uploaded file certain steps are followed for each type of file. If the uploaded file is a text document, then the text is extracted using basic python file handling functions.

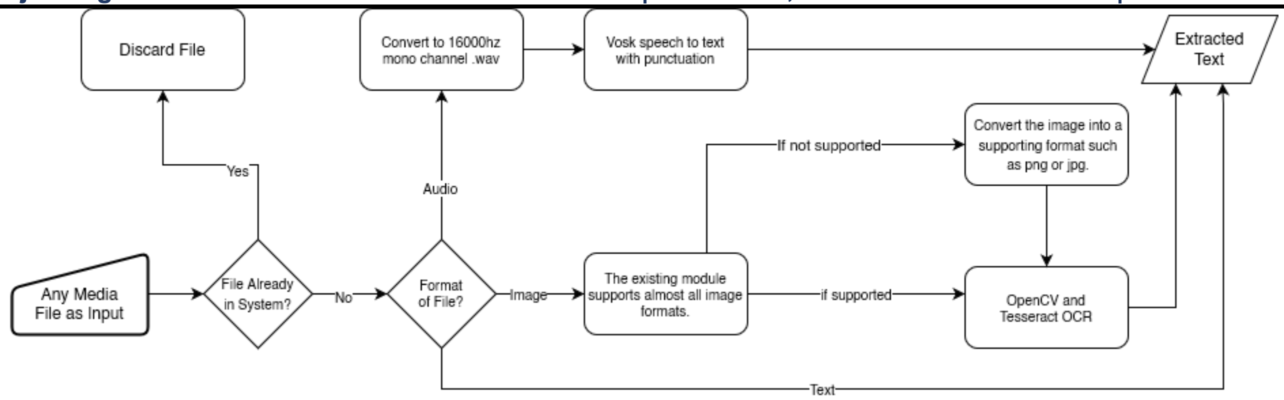

Fig-2: Input Processing

3.1.1 Text Extraction from Images

If the uploaded file is an image, then appropriate functions of OpenCV and tesseract OCR are used to extract the text. There are very few cases where for some unique image formats OpenCV cannot extract text, thus in such cases, an additional step is taken where the existing image is converted into a format that is supported by OpenCV such as png, jpg, etc. After it has been converted into a supported format then OpenCV and tesseract OCR is used to effectively extract the text from the image. While performing the text extraction from the images the images are first converted into grayscale after which thresholding is done.

3.1.2 Text Extraction from Audio

The input audio files are converted into .wav format as there is a module in python which provides lots of functionalities for the handling of .wav file, then the audio file will have its frame rate set to 16000 Hz along with the usage of a single channel which is known as mono channel, after doing the initial pre-processing on this audio file it is passed onto Vosk model which provides the functionality for efficient extraction of text. Vosk is generally preferred by everyone as it provides functionality not only for English but specific to the dialect of Indian English along with 20+ other languages. This Vosk model when implemented in this system will output the text from the audio file and the same will be copied into a text file which is further stored in the existing file system. All these files which have extracted text in them will be given as input to the next phase which is keyword extraction.

3.2 Keyword Extraction

The second phase of the proposed system utilizes advancements in NLP to create value even from unstructured data. The previous phase gives us extracted text originating from various sources namely text, audio, and images. Thus, the semantics of the extracted text vary from each other in terms of structure, size, and flow of words. The same keyword extraction algorithm will not work on different sources of text.

3.2.1 Models for Keyword Extraction

Before keyword extraction, we can apply stopword removal and the process of stemming and lemmatization to obtain only root words. Piskorski, J et.al [3] have done a comparative study on the various available keyword extraction algorithms and determined that YAKE and KPMiner have the highest score. A hybrid of these two called YAKE-KPMINER-R is seen to have the best results. Keeping this in mind the keyword extraction module aims to use YAKE and thereafter the hybrid for this process. However as discussed by Wang, Yang et.al [6] existing systems suffer when dealing with shorter texts as building context from them is harder. The use of a Recursive Neural Network to train using a purpose-built keyword dataset is required. As YAKE requires semantics for keyword extraction in the case of large text which lacks sufficient semantics i.e., text originating from images or audio, the use of KeyBert can be made. As it is based on BERT embeddings the lack of semantics does not affect it adversely.

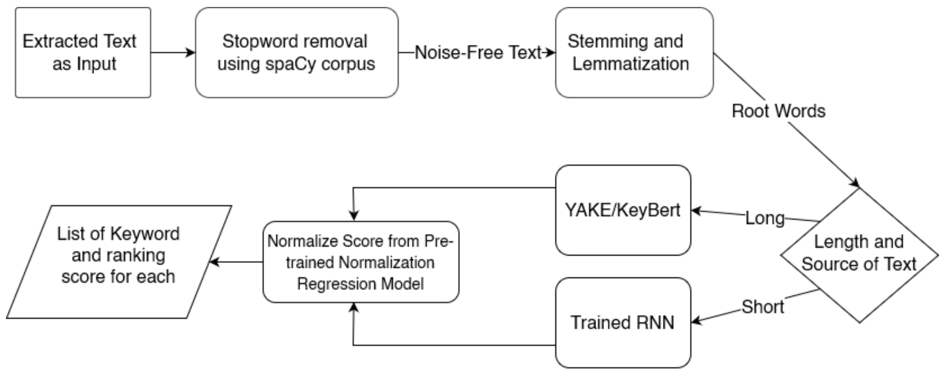

Fig-3: Keyword Extraction Process

3.2.2 Dataset Generation using Web Scraper

For this, we have created a web scraper that extracts the titles from ted talk articles along with meta-tag keywords placed for search engine optimization. Using lemmatization and stop-word removal, we can maximize output from the scraper. Further, we must manually verify these over 2000 entries to correct and enrich the dataset for our specific purpose. All nouns must be a keyword in the dataset as noun identification is weak currently. As all nouns are potential search parameters, they must rank high in relevancy and should be included in the result set. Other considerations, such as the ratio of text to be considered for keyword ranking to cater to our needs have been set to 45:100. Testing must be done to fine-tune this ratio based on results.

Thus, based on the extracted text's source and length we will use the YAKE/KeyBert hybrid or the trained RNN. As both these two models evaluate very differently, we must normalize the keyword ranking scores from them. The idea going forward would be to get them on a scale of 0 to 1. To normalize, we will segment text and pass thousands of entries to both models and compare both scores for each keyword to determine a net normalization factor using a regression model. The output from this module is the list of keywords for the document mapped to the normalized relevancy score and file id.

3.3 Indexing

The third phase of the AutoNBS is to establish an indexing format for faster and more efficient retrieval of files from a storage device. The input to this module will be keywords, along with a set of document identifiers and their relevancy score for the keyword. This input will be mapped into an Inverted index where each semantically consistent keyword will be a key. All the documents that contain the corresponding keyword will be the values of the keys. Document id with score will represent the relatability of the document content for the keyword. This combination will be treated as the value of a document with respect to a key. Example of the input as the inverted index is as follows:

```
{

'police': [(0.87,212), (0.86, 112), (0.91, 23), (0.46, 101)],
'blackmail': [(0.77, 23), (0.75, 48), (0.82, 206)],
'lakh': [(0.92, 23)],
'consent': [(0.84, 206), (0.78, 13), (0.77, 112), (0.54, 212), (0.12, 99)]

}
```

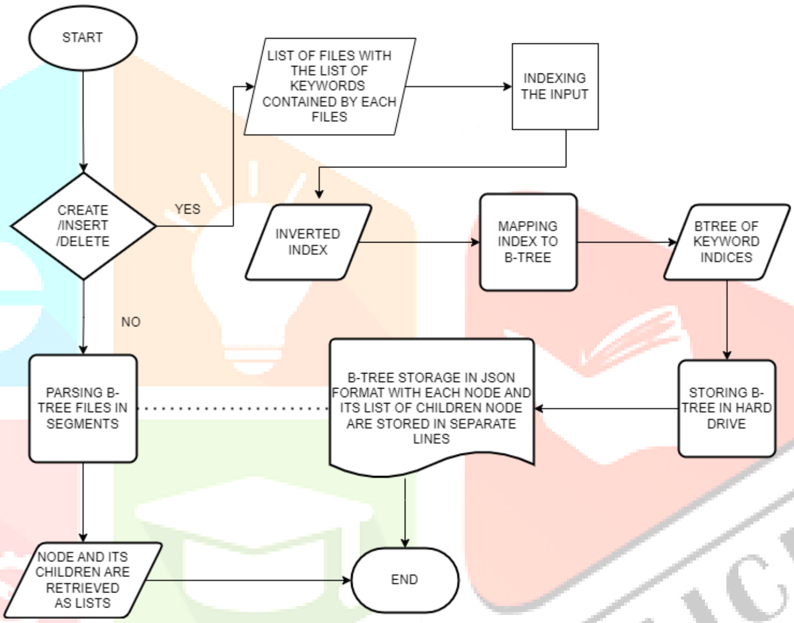

Fig-4: Flow of Indexing process

3.3.1 Inverted Index

The list of semantically usable keywords will be limited for each domain. But the number of documents to be stored and analyzed will never stop increasing. Conventional methods of indexing where each document is a separate key and its value are the lists of associated keywords will lead to the creation of a vertically scaling index file. As discussed by Abhishek Das and Ankit Jain [5], an inverted index can be used to balance the cost of retrieval time. To avoid vertical scaling and reduce the data retrieval process for big data, horizontally scaling the index file will be a better alternative. Inverted index represents the index with the keywords as the keys and the list of documents containing that keyword as its value. Hence, the number of keys in the index file will remain fixed while the list of values will keep increasing.

3.3.2 Representation of B-Tree

The main concern while indexing is the risk of the index file reaching a size where loading the entire file into memory may not be feasible. Abhishek Das and Ankit Jain [5] have mentioned the need for design trade-offs to minimize resource utilization. A self-balancing tree eliminates the need for loading the entire contents of a file at once. A B-Tree is an extension of a Binary Search Tree (BST) where a node can have multiple keys. For an order 'N', a node can have a maximum of (2*N)-1 keys and each number of children in each node is 1 + the number of keys in the node. The tree is automatically updated by rearranging the keys and nodes to self-balance itself after the insertion or deletion of a key. The time complexity for the searching operation is O(log(n)).

While insertion or deletion may take time, the searching operation is faster than from conventional trees. The addition of new nodes/keys is in a bottom-up approach. A node may contain keys up to the size of a memory block. Thus, the depth of the key is maintained to an acceptable range even for large amounts of data with the help of horizontal branching. When a node is filled, the median key is taken and put into a new node. This new node becomes the parent node of the former node. The keys to the left and right of the median are stored as the left children and right children of the parent node respectively.

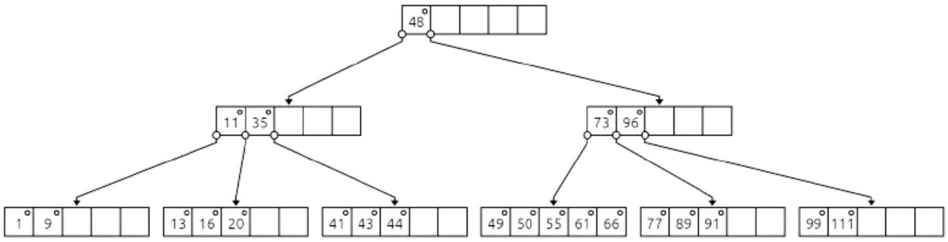

Fig-5: B-Tree of order n=3

3.4 Search System

The final module of AutoNBS is the searching phase, where Apache Solr is being used to provide a search platform that provides us with full-text search capabilities, and interaction with it is done using restful API services via XML, JSON, and HTTP. It is used to create index and search indices to get the required documents easily using rest API with simple get and post methods. As stated by Luburic, N. et. Al. [8], the inverted index technique is used to store content which also helps in reducing the time taken to locate a document. It also offers various features like Auto-completion, spell suggestions, and faceting.

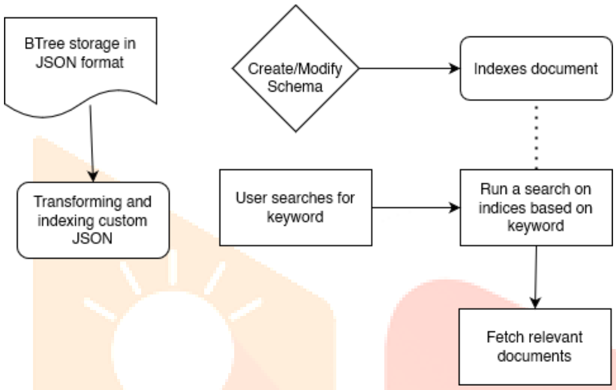

Fig-6: Flow of Search System

The input to this module is the B-Tree in JSON form where at each odd and even line, the node value and a list of children nodes are stored respectively. Uploading of this JSON data is done using index handlers. This JSON document is added to Solr with the update request. A single JSON file can be divided into different documents based on the parameters given with the request. The Solr schema API provides read and write access, which manages the schema's elements. To retrieve or modify the schema, a request is sent to Solr. Based on the schema, the different files are stored in the inverted index form. Now according to the user's search parameter, a Solr setup is done.

The processing of user requests is done by request handlers. Based on the keywords provided by the user, Solr runs a search on the indices containing the document ids corresponding to a keyword. A request handler invokes a query parser to process a search query, analyzing its parameters. A filter query can also be used in search parameters. Filter query builds caches against the entire index, as part of the response. All the relevant documents are fetched and displayed in JSON form. The top N retrieved documents can be re-ranked using a different query and by using a trained machine learning model by utilizing the Learning to Rank(ltr) module. The appropriate use of filter queries can enhance search performance and can also be used to store a user's history of searches. This data can be further used to include a bias to the user's future searches.

IV. RESULTS AND DISCUSSION

4.1 Input Module

Testing of various OCR methods paired with pre-processing steps has been made to accommodate text extraction based on the format of the file ingested into the system. For audio files, the standard requirement of the Vosk model is mono-channel 16000hz .wav form which is achieved by using the FFmpeg encoder to convert audio files to the required file format. Similarly, for images, the standard form is .jpg. As mentioned by Kumar, Anubhav [1] and A, Vinnarasu et.al [2], optimizations for applying the Gabor filter and adding punctuations in a speech-to-text process respectively have been considered.

4.1.1 Extraction of text from Images

The given sample image was used for checking the accuracy of text extraction from an image, the accuracy was far better when OpenCV and Tesseract OCR was used in combination than by using PyTesseract and Tesseract OCR. The results for the same are shown below.

Fig-7: Sample Image for Text Extraction

With pytesseract and Tesseract OCR:

'7 eo [| LLL he\n LAW INSIDER\n\nTHE ABHORRENT AND AND UNACCEPTABLE TRUTH OF\n'

With OpenCV and Tesseract OCR:

THE ANDNRENT AND UNACCEPTABLE TRUTH OF
CUSTODIAL DEATH

4.1.2 Extraction of text from Audio

To extract text from the audio files, a model known as Vosk is used, which gives an average accuracy, but we are trying to find a model which could improve the accuracy even in the presence of background noise.

The actual transcription done manually for cross-verification is shown below,

"Sunday march thirteeth two thousand fourteen the time twelve fifty two pm police how can I help you. I have three hostages and I need money three million dollars may I know your name thats not relevant here as I am having the hostages. I would also need the air support and money in three different bags. I am going to hang up the phone now so that you dont get to call me again and I am repeating that I want my money within the next three minutes."

On providing the model with the audio of a 911 call it extracted the below text from the audio file.

''sunday march thirtieth two thousand fourteen the time twelve fifty two pm bucket please how can i help you or fruit offices there are people that switch from one hundred fairly clear plot i'm sorry i have three dollars and air fifteen and one hundred for a seat for a hundred gain a clear pasta that what i just saw you at yeah what's your name it's irrelevant don't have my money i need my money delivered by one on our officer otherwise i'll kill everybody so is it that''

4.2 Keyword Extraction Module

A keyword dataset was generated to train an RNN. This RNN will be capable of extracting keywords and ranking the relevance with greater accuracy for short text and unformatted text originating from audio and images. The RNN will be built by modifying PyTorch's translation code to cater to keyword extraction. The stopword removal from the keyword list and stemming and lemmatization process has been proven effective while building the web scraper.
The same can be applied to remove noise and get root words from the input extracted text of the previous input module. A comparative study as discussed by Piskorski, J et.al [3] has proven that a hybrid of YAKE and KPMiner is the best way forward. The same was validated by implementing similar algorithms. The best match for indexing keywords based on relevance is YAKE.

Below is part of the output for YAKE applied on the following sentence: "spaCy is an open-source software library for advanced natural language processing, written in the programming languages Python and Cython."

('Cython', 0.06138796274651049)
('Python', 0.09149427881730993)
('spaCy',0.10868092541297539)

The first element of the tuple is the string of keywords and the next is a float value for the keyword rank score in the context of the entire text. The larger the text, the easier it is to build context thus, arises the need of training the RNN for shorter text. The web scraper generated a keyword dataset by extracting titles from ted talk articles along with keywords in the meta tag of the web page used for search engine optimization. NLP was used to identify semantics like nouns which must be identified and used as keywords in later phases.

Table-1: Generated Keyword Dataset and Correction sample

S.no	Title (To extract keywords from)	Raw output from Scraper	After Manual Correction
1	sharon horesh bergquist how stress affects your body	sharon, bergquist, body	sharon, horen, bergquist, _stress, body
2	greta thunberg the disarming case to act right now on climate change	thunberg, act, climate, change	greta, thunberg, act, _climate, change
3	hasan elahi fbi here i am	hasan, elahi, fbi	No Changes

4.3 Indexing Parsing and Output

The Inverted index is mapped into the B-Tree with the identifier of each document taken as the index of a key. The combination of the document id and relevancy score makes up the key for the B-Tree. The B-Tree is stored as a JSON file for each keyword. To read the B-Tree while in segments, the file is stored in the below format.

```
[(48, 2)]
[[(11, 20), (35, 0)], [(73, 28), (96, 4)]]
[(11, 20), (35, 0)]
[[(1, 24), (9, 14)], [(13, 26), (16, 12), (20, 22)], [(41, 18), (43, 8), (44, 38)]]
[(73, 28), (96, 4)]
[[(49, 44), (50, 16), (55, 6), (61, 30), (66, 40)], [(77, 42), (89, 32), (91, 10)], [(99, 36), (111, 34)]]
```

Fig-8: B-Tree Storage Format

The first line contains the root node of the B-Tree. The next line stores the list of children of the root node. To read this file, the first pair of lines will be loaded into memory, with the first line as the parent node and the second line as a list of its children.

The key is printed if the identifier of the document to be found is equal to the id of the parent node's key. Else, if the document id of to be found is just lesser than one of the keys of the parent node, then for the corresponding index, the child node will replace the parent node and the new children of the new parent node will be fetched from the file with offset = (currentIndex*2) + 2. Repeat the previous step. If the document id is greater than the largest key in the current node, the above steps will be applied to the last child of the parent node.

4.4 Search using Solr

Apache Solr being a fast and powerful search engine helps in improving the user experience on the web application. Being a Rest API-based search server and full-text search system along with many other features helps in the fast retrieval of required documents. It also allows us to upload files of various formats like XML, JSON, and CSV. To search for all the files, a query is passed with the value " *:* ".

To search for documents where the name field must contain the word "black", a query is passed with the value "name: *black*". We can also limit the number of fields in the result by giving only those field names in the field list input. It also provides a sorting option that arranges the resultant documents in ascending or descending order. It also allows us to add a specific schema to the Solr instance depending on which form of data is required.

.

V. CONCLUSION

We proposed a system that can efficiently search for files using NLP techniques based on keywords provided by the user as search parameters. We identified methods for keyword extraction based on the different file formats and lengths of text extracted. Keyword extraction from audio gives less accuracy due to inaccurate speech-to-text conversion inherent to the Vosk model. Only those images can be taken for keyword extraction which are above a certain quality level. To make our system work on the short text, using web scraping and some manual correction, we generated a purpose-built dataset. The trained model can be further used in our own system and by others in related research work. The use of inverted indexing to effectively store information and test it using B-tree which gets loaded into memory in segments, so as to reduce the load on memory was shown. We shall also utilize the Solr search engine which uses its REST-like API to give users results in a time efficient manner. Our system will be able to list out all the documents based on the user's input and also sort them as per their relevance. In order to deliver a more personalized experience, we would also attempt to add a certain amount of bias to the users' input depending on their prior search history. The bias will not limit the output based on the user's past patterns only but also include other relevant documents.

REFERENCES

[1] Kumar, Anubhav. (2014). An Efficient Approach for Text Extraction in Images and Video Frames Using Gabor Filter. International Journal of Computer and Electrical Engineering. 6. 316-320. 10.7763/IJCEE.2014.V6.845.

[2] A, Vinnarasu & Jose, Deepa. (2019). Speech to text conversion and summarization for effective understanding and documentation. International Journal of Electrical and Computer Engineering (IJECE). 9. 3642. 10.11591/ijece.v9i5.pp3642-3648

[3] Jakub Piskorski, Nicolas Stefanovitch, Guillaume Jacquet, and Aldo Podavini. 2021. Exploring Linguistically-Lightweight Keyword Extraction Techniques for Indexing News Articles in a Multilingual Set-up. In Proceedings of the EACL Hackashop on News Media Content Analysis and Automated Report Generation, pages 35–44, Online. Association for Computational Linguistics.

[4] Luburić, N., Ivanović, D. Comparing Apache Solr and Elasticsearch search servers. In: Konjović, Z., Zdravković, M., Trajanović, M. (Eds.) ICIST 2016 Proceedings, pp.287-291, 2016

[5] Das, Abhishek & Jain, Ankit. (2012). Indexing the World Wide Web: The Journey So Far. Next Generation Search Engines: Advanced Models for Information Retrieval. 1-28. 10.4018/978-1-4666-0330-1.ch001.

[6] Wang, Yang & Gong, Yeyun & Huang, Xuanjing. (2016). Keyphrase Extraction Using Deep Recurrent Neural Networks on Twitter. 836-845. 10.18653/v1/D16-1080.

[7] Devlin, Jacob & Chang, Ming-Wei & Lee, Kenton & Toutanova, Kristina. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

[8] Amodei, Dario & Ananthanarayanan, Sundaram & Anubhai, Rishita & Bai, Jingliang & Battenberg, Eric & Case, Carl & Casper, Jared & Catanzaro, Bryan & Cheng, Qiang & Chen, Guoliang & Chen, Jie & Chen, Jingdong & Chen, Zhijie & Chrzanowski, Mike & Coates, Adam & Diamos, Greg & Ding, Ke & Du, Niandong & Elsen, Erich & Zhu, Zhenyao. (2015). Deep Speech 2: End-to-End Speech Recognition in English and Mandarin.

[9] Gerritse, Emma & Hasibi, Faegheh & de Vries, Arjen. (2020). Bias in Conversational Search: The Double-Edged Sword of the Personalized Knowledge Graph. 133-136. 10.1145/3409256.3409834. ICTIR '20