www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 5 Issue 11 Nov. 2016, Page No. 18911-18915

Alternative Method of Audio Searching (AMAS)

Biprajeet Pal[1], Vishal Ramanathan[2], Abhilaksh Sharma[3] , Riten Shah[4], Sarthak Raisurana[5]
[1 2 3 4 5]School of Computer Science and Engineering, VIT University,
VIT University , Vellore - 632014
biprajeet.pal@gmail.com

Abstract: ***With the advent of searching technologies becoming more and more efficient day by day it has become impertinent to apply the same on audio. Often times we require search results which if in audio format makes more sense. Our existing algorithms use waveforms to search on audio which requires indexing of all audio files or the audio file is matched via the names of the file itself and pertaining metadata about the file. In order to make real and real-time searching on the audio files possible we use this methodology. This paper merely outlines a possible way of searching for the content of the audio files. We do this by converting the audio to text and then search on the text itself thus giving better results. In this paper we review on the previous works and provide a rough guideline how this can be achieved.***

Keywords: speech to text, audio format, audio searching , audio conversion , text compression , text searching algorithm , review paper.

1. Introduction

The way to document audio files for future use is called oral documentation. The gathering historical information preserving them and interpreting this information through recordings of interviews of participants and communities for past events. Oral History is a research area and provides techniques of preserving history. It is a method of cultural documentation. It helps in preserving the form of verbal information which has existed since before recorded history. This can be applied for the study of any subject of history. Oral History is applied in the research of military history, social history, economic history, cultural history, business history, political history and community history.

This paper focuses on bringing the Oral History documentation into automation with advanced speech and script format. This aims to use a speech recognizer to convert audio from historical and present interviews and speeches and then convert these audio files automatically in to a textual format that records the entire audio as a script in a text file.

For the process of extracting audio from the file and recognize speech from it, we use the latest and efficient speech recognition and pattern analysis algorithms already available in the market. Real-time recognizer uses an isolated word dictation implemented with a 500 word vocabulary. It uses self-organize, statistical approaches underlying the basic speech recognition algorithms of the recognizer. Previous approaches relied heavily on expert input through the painstaking analysis of data to release speech signals to the word sequences that produced them. Such methodologies were completely displaced by casting the speech recognition problem in a probabilistic framework by modeling the joint probability distribution of speech signals and word sequences. At the beginning of the 21st century , the amount of data and computation to train and builds models has increased exponentially, and the emergence of new machine- learning algorithms and methodologies has opened new vistas in approaching complex pattern recognition problems. This is enabled by a new set of machine- learning techniques referred to as graphical models with computationally tractable training algorithms. Closely related are neuro-network modeling techniques, and there has been a resurgence of interest in the application of neural – network concepts such as deep networks to speech recognition. The explosion of data has caused the development of new ways to capture the key features in massive amounts of data using efficient methods deploying exemplar-based sparse representations. Lastly, all of these different approaches can be tied together in a principle fashion using another variation of graphical models: an exponential model framework.

This paper describes the current state of the art speech recognition systems to bring the audio speech into a scripted text file which then undergoes severe compression system to reduce its size and compounded along with the existing audio file to establish a format that allows archiving the data as a compound of audio and its script which helps in documentation of the oral history and makes for better search.

It is often required to reduce data size in order to save space and time spent in transmission, and this process is called compression. In this paper we review technologies that utilize the complementary processes of text compression and speech recognition. Data compression involves encoding information using fewer bits than the original representation. Compression is useful because it reduces resources required to store and transmit data. Computational resources are utilized in the compression process, as well as the decompression process. The first objective in order to save text with the audio is to recognize the speech and convert it to text. Speech recognition methodologies have seen improvements over the years, and advancement of technology has enabled more accurate reporting and subsequently storage of more quality information.

.

2. Review of Methodology

A. Speech recognition (speech to text)

A wide array of works in literature bears witness to the successful attempts being made at using speech recognition to extract meaningful words from dictation. For instance, [3] use a speech-to-text interface integrated to Mammo-Class which enables radiologists to dictate a mammography report rather than physically typing it in. Despite a wide range of success, speech recognition engines work well given the vocabularies are limited i.e. lexemes are less and speech is delivered in isolated and dedicated mini-worlds. Thus rendering it less suitable in noisy environments delivering a poor performance. An idea [6] is to reduce error rate by successive revisions of the dictation thus focusing on the text itself rather than the relevant word to build a data-structure for posterior automata study.

Speech recognition is many times confused with voice recognition while in fact, both are quite different, has separate objectives. While voice recognition deals with identifying a person with the voice sample and previously learned voice data Irrespective of the language, whereas speech recognition only deals with identifying what the lexemes i.e. words in the speech, depending on the language and not how its' pronounced. Our aim here is to recognize and extract relevant lexemes from dictated texts. So some of the major speech to text techniques will be discussed here.

Researchers have been trying for decades to create a speech recognition system. The earlier [1] automatic speech recognition techniques involved template matching methods. Template matching involves comparing the incoming speech with a set of reference patterns and then recognizing the common words. This involved having templates for each word and the comparing which would take a lot of execution time. Hence this concept was difficult to implement and run

Other early attempts included including high level knowledge based systems that used expert knowledge for speech processing and recognition. These had limited success because creating expert knowledge would imply a lot of manual based rule methods which will involve too many constraints. So this method was discarded and data driven approach was used to extract the information.

Today's speech recognition is much broader. We aim at not only recognizing the word but also recognizing the person's signature. Also people don't speak just a single language anymore. Many languages are spoken together to form sentences. These will also require translation and hence the speech recognition is an uphill battle.

Data driven methods [1] created language models with the purpose to handle syntactical and grammar aspects of the language. Another common problem faced was different pronunciations for the same word and also the different dialects of the same language. These were solved by data driven aspect as the language model was represented by grammar especially context-free-grammar. The technique involves pre-processing the speech signal and removing noise. Then extract the phonemes and then do the acoustic matching. This method proved to generate much more accuracy in recognizing the pronunciation. The primary reason for that is the level/generation of the computers available to us today. It has made data processing much faster. Even very complex models can be solved in a few hours.

The current best and fastest method for speech recognition is using neural networks [2]. The initial goal is to build a classifier that can recognize the phonemes and convert them to text. The input speech data is either un-aligned or aligned. Aligned data is very easy to work on but unaligned data is not. So there are two options to convert the input speech into un-aligned pattern into aligned or directly build a system that works on un-aligned data.

The next objective is choosing the right algorithm for searching a phoneme. The one chosen after lot of research is forward-backward algorithm for Makarov models. The reason why this is chosen instead of the basic naive – search is that is starts by searching for the prefix of the string so instead of searching the whole word only the prefix is matched first and then when hit it moves forward with the next phoneme. Also in this method probability technique is incorporated to ensure a faster search and hit for the phoneme.

The most difficult objective here is training the software to recognize human speech and not a string of phonemes. This also includes removing the sound data and only concentrating on the speech.

Speech-to-text software is a type of software that effectively takes audio content and transcribes it into written words which is the idea of this whole context. One of the popular speech-to-text interface is provided by Google's Web Speech API (Application Programming Interface, https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html) which allows the programmer to obtain a translation of voice to text, and also Voice Note (https://voicenote.in) that is a free extension for Google Chrome.

We look at Free Voice to Text (http://download.cnet.com/Free-Voice-to-Text/ 3000-7239 4-76115951.html). This software is used to send emails and documents just dictating. It is a free tool and supports the following languages: English, Spanish, French and Japanese. Next was Talking Desktop (http://voice-recognition-software-review.toptenreviews.com/talkingdesktop-review.html). The next software is Dragon Naturally Speaking Home (and Premium version) (http://www.nuance.com/for-business/by-product/dragon/product-resources/edition-comparison/index.htm).

All of these examples require that the user installs some software. Others allow the access via web browser or via Application Programming Interfaces (API). One of them is the Web Speech API (https: //dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html).

We tested all the freely available sources and gave preference to Google web speech API. It is also interesting to note that despite many speech recognition software existing they are still only used to recognize words. The future aim is to considerably reduce the execution time broadening the scope to recognize multiple speakers at the same time.

So our proposed model will be using Google speech for speech recognition and speech synthesis.

1) System Description

This system has two modules. The recognition module and the correction module. While the first module deals with converting the speech of the dictated text to actual text the second module deals with the edition or deletion of a word. The basic working of this tool is shown in Fig. 1.

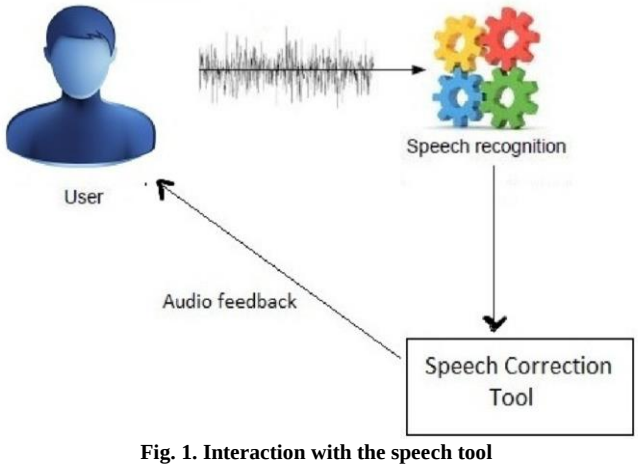

Fig. 1. Interaction with the speech tool

For speech synthesis and recognition, Google Speech API is used.

1) Google Speech Recognition

Google speech works online. Initially , the speech is recorded. This is then sent to the Google server. Google returns a result with the set of other possible results of highest probability. The language may be any known accent. The punctuation marks is also generated.

2) Recognition module
At first the sentence spoken by the user is recorded in a flac audio file. Then the audio data is sent to google server. Then Google processes this audio using its own algorithms. Then it returns the highest possible result along with other possible results. This process is shown in Fig. 2.

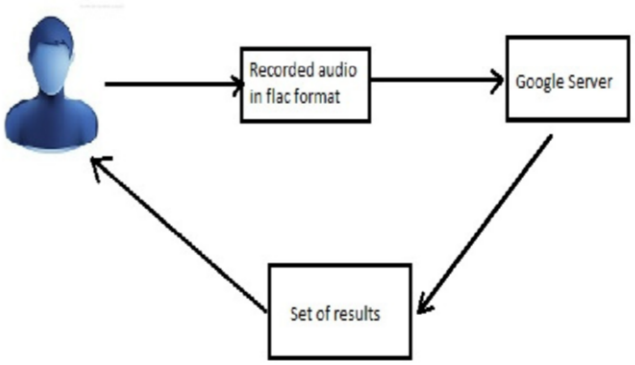

Fig. 2. Recognition module

3) Correction module
For text correction, user has to enter word replace or word deletion mode. After entering a mode, the user will be notified through the Google speech synthesizer. Then he will be asked for a specific word which should be deleted or replaced. After the user gives his choice, he will be notified through audio feedback whether it is found or not. If it is found, he will be asked which word he would like to replace it with. Then if he speaks a word, the replacing will occur successfully. The process is shown in Fig. 3.

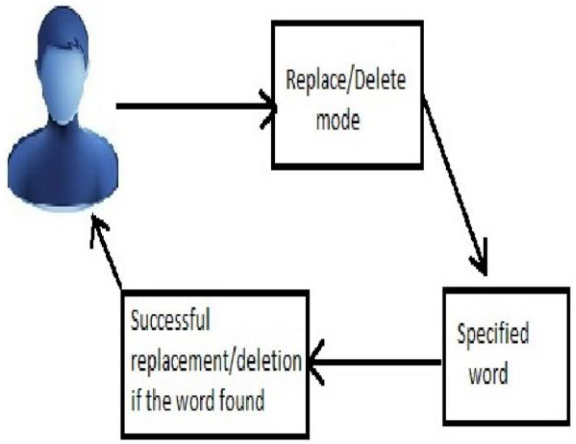

Fig. 3. Correction module

B. Text Compression

There is always a trade-off between speed and compression ratio, i.e. a space-time complexity trade-off. For instance, a compression scheme for video may require expensive hardware for the video to be decompressed fast enough to be viewed as it is being decompressed, and the option to decompress the video in full before watching it may be inconvenient or require additional storage. The design of data compression schemes involves trade-offs among various factors, including the degree of compression, the amount of distortion introduced (when using lossy data compression), and the computational resources required to compress and decompress the data.

Data compression may be viewed as a branch of information theory in which the primary objective is to minimize the amount of data to be transmitted. The purpose of this paper is to present and analyze a variety of data compression algorithms. Data compression has important applications in the areas of data transmission and data storage. Many data processing applications require storage of large volumes of data, and the number of such applications is constantly increasing as the use of computers extends to new disciplines. At the same time, the proliferation of computer communication networks is resulting in massive transfer of data over communication links. Compressing data to be stored or transmitted reduces storage and/or communication costs. When the amount of data to be transmitted is reduced, the effect is that of increasing the capacity of the communication channel. Similarly, compressing a file to half of its original size is equivalent to doubling the capacity of the storage medium. It may then become feasible to store the data at a higher, thus faster, level of the storage hierarchy and reduce the load on the input/output channels of the computer system.

1) Types of Data Compression

Compression can be of two types:

1. Lossy - Lossy compression is compression done by reducing bits by eliminating unnecessary or less important information. After decompression, the data that is retrieved will be close to the original data, but will not be the exact same. Some data from the original message will be lost.

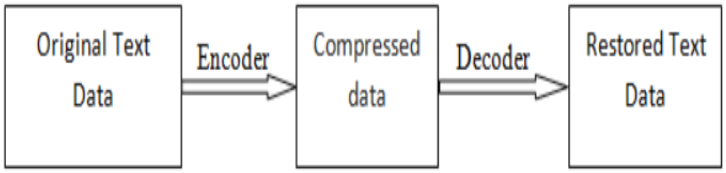

Fig. 4 Lossy Compression

Since we are focusing on compression of textual data, lossy compression is not suitable, since all the information in the text is critical. Lossy compression is mainly used for Digitally Sampled Analog Data (DSAD). DSAD are files that include audio, graphics, images or video. We shall focus on Lossless data compression for textual data.

2. Lossless - Lossless compression reduces bits by identifying and eliminating statistical redundancy. Here, data is not lost during compression and decompression. The exact original data can be retrieved from the compressed file. Algorithms to compress text data in a lossless manner exploit statistical redundancy in a manner that the data is represented in a more concise manner without loss of important information within the text data. Lossless compression is possible due to the presence of statistical redundancy in most real-world data.

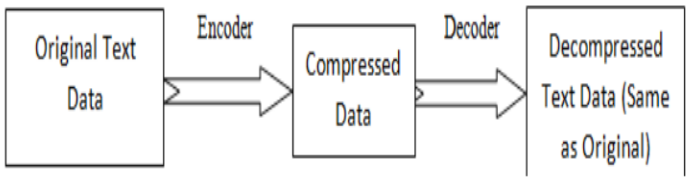

Fig. 5 Lossless Compression

Most lossless compression algorithms have two parts – one which generates a statistical model for the input data and another which maps the input data to bit strings using this model in such a way that frequently encountered data will

produce shorter output than improbable(less frequent) data. Lossless data compression works best for text data.

2) Existing lossless data compression techniques

1. Bit Reduction Algorithm

This algorithm was originally implemented for use in a text file like message communication application **[7]**. The main idea behind this program is to reduce the standard 8-bit encoding to some application specific 5-bit encoding system and then pack into a byte array. This method reduces the size of a string considerably when the string is lengthy and the compression ratio is not affected by the content of the string.
Bit Reduction Algorithm in steps:

- Select the frequently occurring characters from the text file which are to be encoded and obtain their corresponding ASCII code.
- Obtain the corresponding binary code of these ASCII key codes for each character.
- Then put these binary numbers into an array of byte (8bit array).
- Remove extra bits from binary no like extra 3 bits from the front.
- Then rearrange these into array of byte and maintain the array.
- Final text will be encoded and as well as compression will be achieved.
- Now decompression will be achieved in reverse order at the client-side Improved Dynamic Bit reduction algorithm (Hybrid Approach).

2. Syllable Based Morphology

This approach considers the syllables in words, and their significance in many languages. Many syllables are shared between words, and are usually longer than a single character and shorter than a word. **[8]** Therefore, both character compression and word compression can be utilised. This approach is most suitable for short text files. The proposed approach is different than previous syllable based compression approaches in that, syllables are used as the basic unit and these fragments are compressed utilizing an automaton to produce a volatile dictionary.
Languages are classified into Mono-syllable and Multi-syllable languages.
The algorithm works as follows:

- A file, considered as the source, consisting of text is taken
- The text is searched for characters that may not be included in the alphabet using a filtering unit.
- The words are divided into syllables using the syllables unit.
- A dictionary for compressing the input text is created by the compression unit, which produces the target file.
- The dictionary consists of different syllables contained in the text and binary codes corresponding to them. This is a volatile file.
- Finally, the target file which contains the compressed data.

This algorithm is useful in many ways. It reduces chances of corruption in the resulting compressed files. This is because any corruption occurring when compressing a syllable results in a typo only for that syllable as a result of decompression leaving the rest of the text uncorrupted. Bit errors are extremely dangerous for most of the other algorithms which result in unreadable files after the point of corruption. This algorithm, is modular with clearly defined modules. This makes the entire system easy to understand and make modifications to. Less memory space is needed due to the use of a volatile dictionary. Finally, it extends theoretical insights from other algorithms reported in the literature and our results may provide a basis for discussions and extensions regarding the use of languages' syllabic characteristics in text compression.

1. Two stage text compression and decompression using Adaptive Huffman (AH) and Parallel Dictionary Lempel-Ziv-Welch (PDLZW) algorithms.

In this approach, two very famous algorithms, the Huffman and the Lempel-Ziv-Welch algorithms, are combined to result in a higher compression ratio than when both are used individually.

This approach works as follows:

A. Compression:

The PDLZW algorithm **[9]** replaces the input text strings into fixed length codes in stage one. In stage two, the Adaptive Huffman algorithm replaces the fixed length codes into variable length codes.

B. Decompression

In stage one, the input string is replaced into variable length codes by the Adaptive Huffman algorithm. These variable length codes are replaced into fixed length ones by the PDLZW algorithm.
The efficiency of data compression by using the two algorithm together can be up to 41.50%.

In the new model,

A. Compression:

The Huffman algorithm replaces the input text strings into variable length codes in stage one. In stage two, the LZW algorithm replaces the variable length codes into fixed length codes.

B. Decompression

In stage one, the input string is replaced into fixed length codes by the LZW algorithm. These fixed length codes are replaced into variable length ones by the Huffman algorithm.
By interchanging the compression stages in the new model, the efficiency can be up to 46.19%.

2. Hybrid Approach combining Dynamic Bit Reduction and Huffman coding.

This algorithm works in two stages for the compression of text data. **[10]** The data is first compressed using dynamic bit reduction followed by further compression using Huffman coding. When input data is entered by the user, the number of

occurrences in the string will be discovered by the system with numeric codes assigned to these unique symbols. For every numeric code, corresponding binary codes will be generated to obtain the (compressed) binary output. The binary output generates ASCII code to which Huffman coding is applied, further compressing the data. Huffman coding follows top down approach means the binary tree is built from the top to down to generate an optimal result. In Huffman Coding the characters in a data file are converted to binary code. The most common characters in the file have the shortest binary codes, and the least common ones will have the longest binary code. Decompression works in reverse order. Compressed data is first decompressed by Huffman Decoder and then by dynamic bit reduction decoder to get back the original data.

References

[1] Mikko Kurimo, Bowen Zhou, Rongqing Huani's and John HL. Hansen, "Language Modelling Structures in Audio Transcription for Retrieval of Historical Speeches" in SPC 2004.

[2] Eniko Beatrice Bilcu, Petri Salmela, Janne Suontausta, Jukka Saarinen," Application of Neural Networks for Text-to-Phoneme Mapping" in SPC 2004.

[3] Ricardo Sousa Rocha, Pedro Ferreira, Inˆes Dutra, Ricardo Correia, Rogerio Salvini, Elizabeth Burnside, "A Speech-to-Text Interface for MammoClass" in ISCBMS 2016

[4] Md. Nafiz Hasan Khan, Md. Amit Hasan Arovi, Hasan Mahmud, Md. Kamrul Hasan, and Husne Ara Rubaiyeat, "Speech based text correction tool for the visually impaired" in ICCIT 2015

[5] Annisa Istiqomah Arrahmah, Aulia Rahmatika, Samantha Harisa, Hasballah Zakaria , Richard Mengko, "Text-to-Speech Device for Patients with Low Vision" in ICICI-BME 2015

[6] R. Patel, B. Greenberg, S. Montner, A. Funaki, C. Straus, S. Zangan, and H. MacMahon, "Reduction of voice recognition errors in radiological dictation: Effects of systematic individual feedback,"
http://clinicaleffectiveness.uchicago.edu/files/2013/07/Reduction- of-Voice-Recognition-Errors-in-Radiological-Dictation-Effects-of-Systematic-Individual-feedback.pdf, accessed in Feb 2016.

[7] Rajinder Kaur, Er. Monica Goyal, "An Algorithm for Lossless Text Data Compression", International Journal of Engineering Research & Technology (IJERT), Vol. 2, Issue 7, July – 2013

[8] Ibrahim Akman, Hakan Bayindir, Serkan Ozleme, Zehra Akin, Sanjay Misra, "Lossless Text Compression Technique Using Syllable Based Morphology", The International Arab Journal of Information Technology, Vol. 8, No. 1, January 2011.

[9] Raja P., Saraswathi D., "An Effective Two Stage Text Compression and Decompression Technique for Data Communication", International Journal of Electronics and Communication Engineering, Vol. 4, Number 2, 2011.

[10] Amandeep Singh Sidhu, Er. Meenakshi Garg, "Research Paper on Text Data Compression Algorithm using Hybrid Approach", International Journal of Computer Science and Mobile Computing, Vol. 3, Issue 12, December 2014.