Keyword Extraction from Multiple Words for Report Recommendations in Media Wiki

IOP Conf. Series: Materials Science and Engineering **183** (2017) 012029 doi:10.1088/1757-899X/183/1/012029

KEYWORD EXTRACTION FROM MULTIPLE WORDS FOR REPORT RECOMMENDATIONS IN MEDIA WIKI

Elakiya K, Arun Sahayadhas*

Department of CSE, Vels University, Chennai, India

*arun.se@velsuniv.ac.in

Abstract:

This paper addresses the problem of multiple words search, with the goal of using these multiple word search to retrieve, relevant wiki page which will be recommended to end user. However, the existing system provides a link to wiki page for only a single keyword only which is available in Wikipedia. Therefore it is difficult to get the correct result when search input has multiple keywords or a sentence. We have introduced a **'FastStringSearch'** technique which will provide option for efficient search with multiple key words and which will increase the flexibility for the end user to get his expected content easily.

Key Terms: mediawiki , Fast String Search , Cirrus Search

1. INTRODUCTION

In many information retrieval applications it is necessary to locate quickly some, or all the occurrence of a user specified word or phrases in one or several arbitrary text strings [1]. Specifically, the retrieval of unformatted data takes time. Therefore secondary index has been introduced which contains the keyword fragments [2]. Searching the index with user specified keywords yields the superset of pages required. In case of larger alphabets the Boyer-Moore algorithm takes larger time to search for all the occurrence of keyword in the document. Moreover, if the length of the search phrase is longer, then the performance of this technique gets even worse.

To overcome this problem, we are implementing FastStringSearch (FSS) Technique [5] [3].This technique starts comparing the text pattern from left to right with multiple keywords.The main idea of this technique is to improve the performance of the search. FSS will start to compare the pattern against some portion of the text, where a possible match occurs and will list all the pages which has any one of text specified in the input. This gives the benefit to the user by specified set of keywords instead of single keyword. Specifying multiple keywords will always give better results which is not possible with the existing system. The performance of the search is calculated by **using Squid caching** [4] **and Varnish caching** [5]. Squid stores the copies of webserver pages and when the same page is requested again, Squid will provide the page from its storage. Because of Squid, web server need not generate the page again and again which boosts the performance of webserver. Varnish caching is also same like Squid. A varnish store the page of webserver in Apache server and it retrieves back when the same page is requested again. This inturn boosts the performance of MediaWiki [6].

2. Methodology:

2.1. Acquiring the text:

The input would be acquired either as a text or audio file from user dynamically; audio file is converted into text and stored as a text in the database. This enables relevant documents to be fetched by a word or multi-words search directly by using mediawiki. It also enables to get the results simultaneously.

Speech will be recognized by microphone which will act as input for the text conversation. Invoking of online speech converter does the conversion to digital text. The converted text is considered as keyword and stored in the database. When the user searches with the corresponding keyword, match happens and results are fetched. Fig 1 illustrates as to how Speech recognition and text Conversion happens.

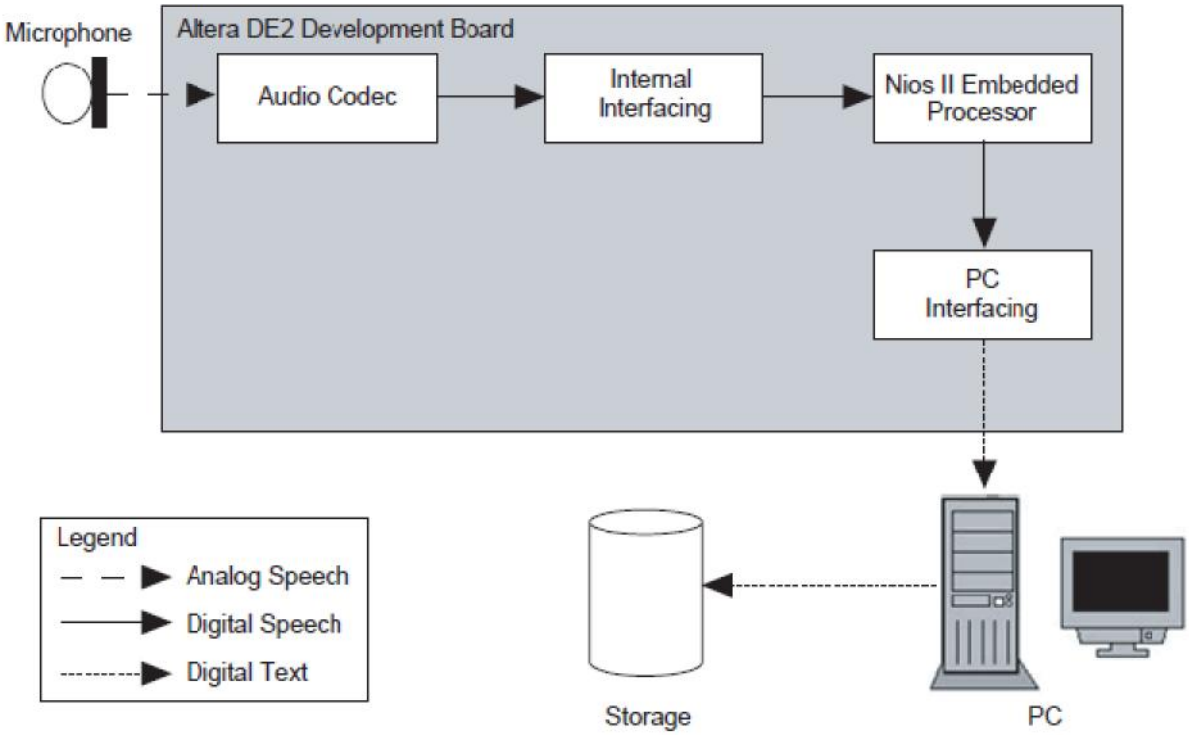

Fig 1: Speech recognition and text Conversion [7]

2.2. Keyword extraction:

The keyword is extracted from the search engine using Cirrussearch [8]. The same Cirrus features can be used in API queries that users can enter in the search box. For example, special prefix can be used to find related pages.

Fig 2 shows how text is forwarded to message formulation. The code converts the text into code form which is a discrete input with a speed of 200bps. Neuro-Mascular controls the articulatory motions with a constant speed of 200bps and transferring those motions to Vocal Tract System is done with a constant speed of 64-700 Kbps. Transmission channel helps to transfer the acoustic waveform from one form to another. Basilar Membrane Motion analyze the spectrum and send

to Neural Transduction for feature extraction and forms a continuous output. This output is sent to language translation with proper sentences, words and phonemes. Thus the given audio file would be converted into in a text message.

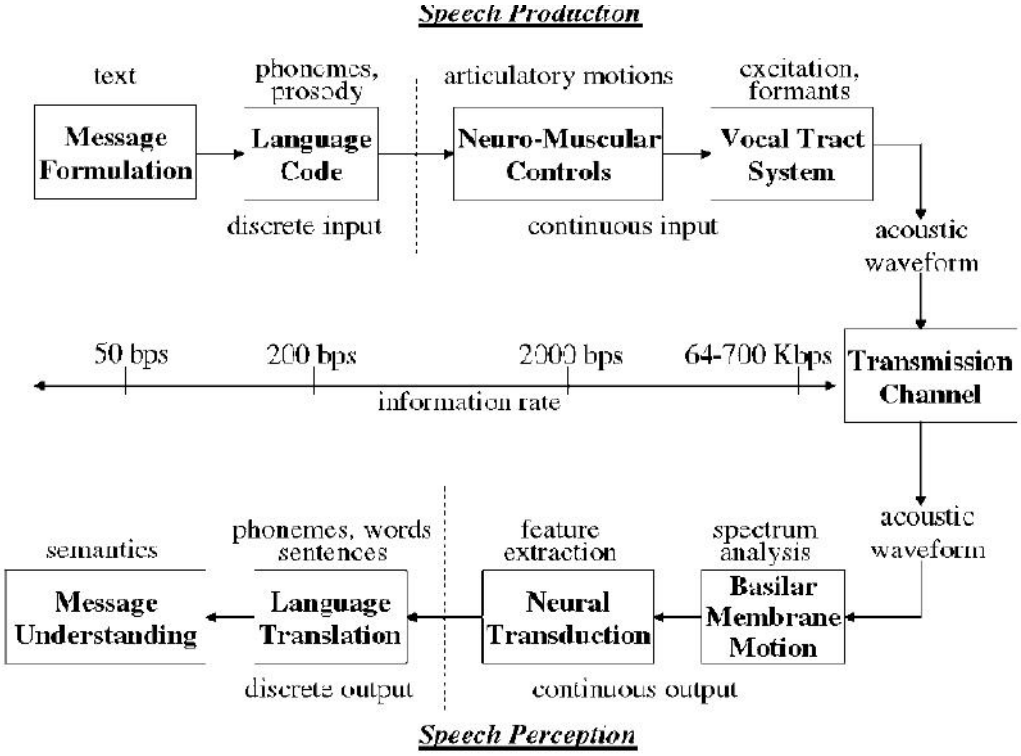

Fig 2 : Flow chart for converting audio file into text message

2.3. Extracting relevant Documents:

Through fast string search and cirrus search technique we can fetch any type of relevant documents as (text, pdf, video, audio etc) based upon the performance tuning by Squid caching [11]and varnish caching [5] methods.

2.4. Displaying the Document.

The multiple word semantic search is used to do retrieval, fragmentation of each word, pertinent documents which can be suggested to end user and it will display the result as **Automatic Content Linking Device(ACLD)** [12] of relevant multiple documents.

3. Results:

3.1. Acquiring the audio & conversion to text

The GUI for acquiring the speech from end user is as shown in Fig 3 and converting the speech to text is shown in Fig 4.

Fig 3:Audio listener from user end

User has been provided with the option to choose the preferred language. In Fig 4, 'English' has been chosen as the preferred language. Once the user chooses his preferred language, on clicking the button 'Voice', listener gets invoked and user provides his speech input via microphone.

Fig 4: conversion message from audio to text

Converter converts the recognized speech to text and fills the text box as shown in Fig 4. User also has been provided with the option to save the input text into the data base. When the user clicks on 'Talk It!' button, system converts the text into audio and reads the same to the user.

3.2. Keyword & relevant document extraction

As shown in Fig 5 and Fig 6, the media wiki takes either a single word or Multiple words search.

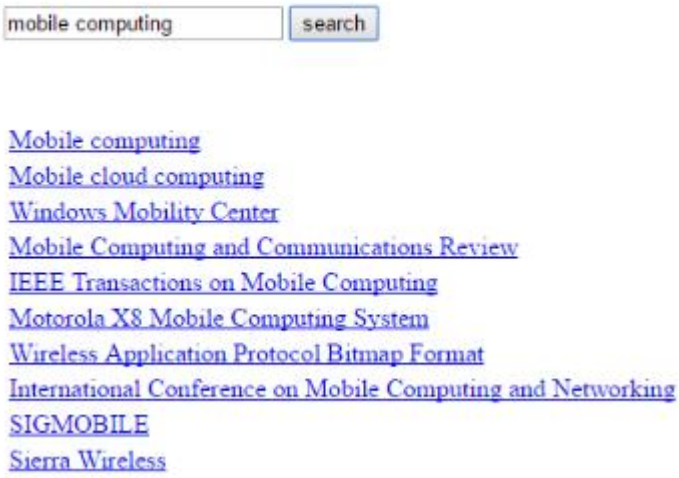

Fig 5: The media wiki search takes as a keyword and fetches the result.

Once speech is converted in to either single word or multiple keywords, those are given as input into MediaWiki. The system extracts the relevant documents based on either single keyword or mulitple keywords.

Fig 6: MediaWiki takes multiple word as search input and retrieves the document

3.3. Displaying the documents to the user

The system displays the extracted relevant documents to user based on the search input for both multiple keywords and single keyword.

3.4. Performance comparison of proposed system with existing system

The Proposed system gives an option to do efficient search with a sentence instead of a word which gives better results. The search is much faster when compared to the existing system. Users have an option to look the information directly in MediaWiki, and in Wikidata by reading Help:**NavigatingWikidata**. Search can be done with whole words, with spaces or punctuation marks. So if the search has the word 'book', the resulting pages with the word 'books' or 'booklet' will not be displayed. Fig 7 to Fig 9 shows the audio conversion loading time, performance and audio tuning timeline respectively for the proposed system.

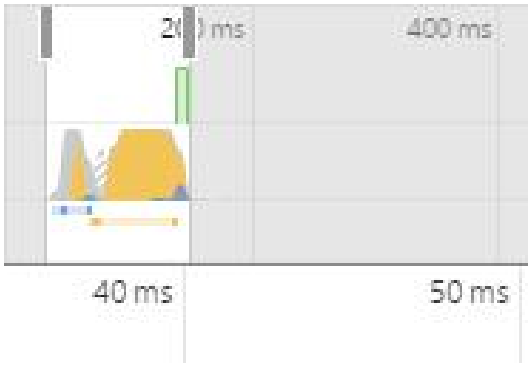

Fig 7: Audio conversion loading time for proposed system

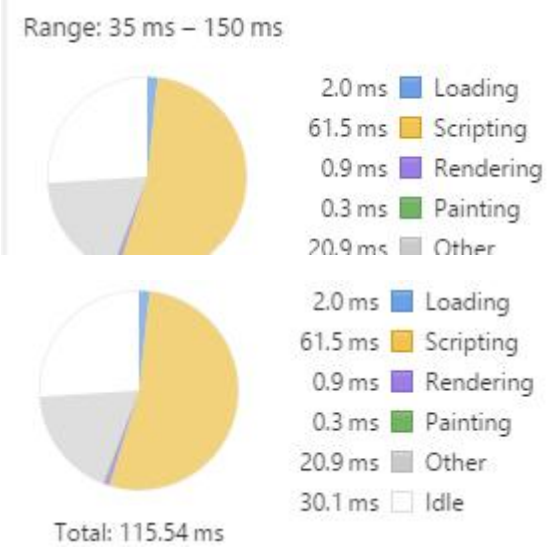

Fig 8: Audio conversion performance for proposed system

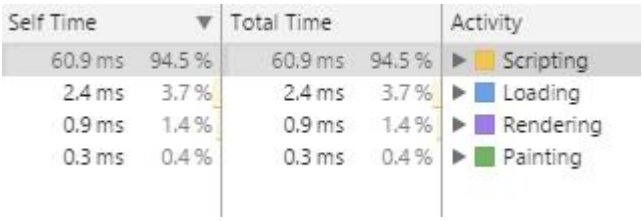

Fig 9: Audio tuning timeline for proposed system

4. Discussions:

The existing system takes an input as a word and retrieve the result by a **Automatic Content Linking Device (ACLD)** [13]. Fig 10 shows the time taken to convert a audio file into text message in the existing system and Fig 11 depicts the frequency timeline for existing system.

Some of the limitations of the Existing system are:

1. Does not searching in an efficient way (i.e) Searching using a sentence will result in incorrect results.

2. The input can only be a word and multiple words cannot be processed. For example, if the word "mobile computing" is given as the input in the search box then it will show relevant document in 60.0 milliseconds. However, if a sentence "What is mobile computing?" is given as input; it will throw an error "Incorrect sentence".

Fig 10: Converting a audio file into a text messages with a speed of 59ms-61ms for existing system

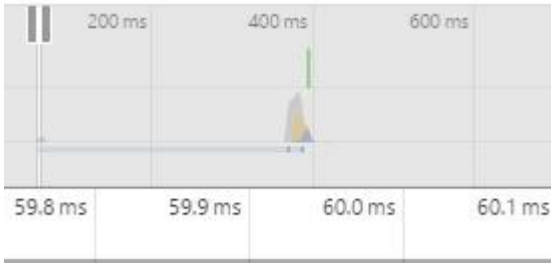

Fig 11: Audio to text conversion frequency timeline for existing system

So, we have overcome this problem by using **FastStringSearch (FSS)** [3]**.** Using FSS, we can search either a word or multiple words and retrieve multiple relevant documents in minimum time (40.0 milliseconds) compared to the existing systems (60 ms). Our system, takes an input as speech (audio), and then retrieves relevant document in text format from the database.

5. CONCLUSION

This fastsearchstring technique improves the search time for the user and the performance is much better than the existing systems. In the proposed system, the single word search or multiple word search retrieves the relevant document in much lesser time than the current system which retrieves only through single word search. In future, real time index updates can be added and the overall stability and performance of search can be improved by upgrading to Elastic based search 2.x.

REFERENCES

[1] C. Charras, T. Lecroq, and J. D. Pehoushek, “A very fast string matching algorithm for small alphabets and long patterns,” *Comb. Pattern Matching*, pp. 55–64, 1998.

[2] D. Brodie, A. Gupta, and W. Shi, “Keyword-based fragment detection for dynamic Web content delivery,” *Thirteen. Int. World Wide Web Conf. Proceedings, WWW2004*, pp. 1030–1031, 2004.

[3] A. Hume and D. Sunday, "Fast string searching," *Softw. Pract. Exp.*, vol. 21, no. 11, pp. 1221–1248, 1991.
[4] B. J. Cooper, "High Performance Web Caching With Squid Table of Contents," 2002.
[5] K. Lyngstøl, "Varnish Cache : What it is and why it ' s cool . Quick Glance," 2013.
[6] D. Filipiak and A. Ławrynowicz, "Generating Semantic Media Wiki Content from Domain Ontologies," *Third Int. Work. Semant. Web Collab. Spaces*, pp. 49–58, 2014.
[7] B. Venkataramani, "SOPC-Based Speech-to-Text Conversion," pp. 83–108, 2006.
[8] N. Everett, S. Engineer, C. Horohoe, S. Engineer, D. Garry, and P. Manager, "CirrusSearch."
[9] K. Kluge, "Elasticsearch in 10 seconds," 2014.
[10] D. Bijl and H. Hyde-Thomson, "Speech to text conversion," 2001.
[11] K. Saini, "Squid Proxy Server 3 . 1 Beginner ' s Guide About the Author," no. 3.
[12] M. Habibi and A. Popescu-Belis, "Diverse keyword extraction from conversations," *Proc. ACL 2013 (51th …*, no. 2010, pp. 651–657, 2013.
[13] A. Popescu-Belis, M. Yazdani, A. Nanchen, and P. N. Garner, "A Speech-based Just-in-Time Retrieval System using Semantic Search," *Acl2011*, no. June, pp. 80–85, 2011.