Study of Audio Information Retrieval on the World Wide Web

Zhiqin Ma

Submitted to Committee Members

Dr. Wenchen Hu (chair)

Dr. Kai H Chang

Dr. Alvin Lim

Department of Computer Science & Engineering

Auburn University

Auburn University, Alabama 36849-5347

CONTENTS

Acknowledgement

I would like to thank Dr. Wenchen Hu for his supervision, guidance, and encouragement during the many discussions we had in exploring the field of audio information retrieval on the WWW. I would also like to thank Dr. Chang and Dr. Lim for their guidance and suggestions.

1. INTRODUCTION

Traditionally, many search engines index the plethora of documents on the World Wide Web. For example, recent systems by Lycos, Alta Vista, Infoseek, and Excite index web documents by their textual content. These systems periodically scan the web, analyze the documents, and create compact and searchable indexes. Users enter query terms and select subjects to more easily find the desired web documents. However, audio is usually treated as an opaque collection of bytes with only the most primitive fields, and users accustomed to searching and retrieving text data may find a problem in looking inside the audio object [1].

A large component of the web content consists of audio information such as speech, music, and sound. The audio information on the web proves to be highly volatile. Web site developers are constantly adding, removing, and replacing audio. Searching for a particular sound or class of sound such as applause, music, or the speech of a particular speaker can be a daunting task [1]. Therefore, it is necessary to require a highly efficient automated system in cataloging and retrieving them, which regularly traverses the web, detects audio information, and processes it and indexes it in such a way as to allow efficient and effective search and retrieval.

1.1 Audio Search Engines

One of the key factors in determining needed engines depends on what type of information

you're looking for. For instance, Yahoo could be used for broad and topic-level queries, such as air pollution, where this subject can be found on the web. This site is also best when you are looking for a particular URL, such as the Environmental Protection Agency home page. Web browsers can search much information, but not for everything. That's why a specific search engine, such as audio, image, and video search engines, needs to be created in order to search timely and relevantly. Search engines crawl the web, and create their listings automatically. In order to improve search capability, it is necessary to create specific search engines according to users needs. For example, an audio search engine can be used to search audio clips on the web efficiently.

1.2 Audio Information Retrieval

Audio data types on the web increase with the rapid increase in speed and capacity of computers and networks. Normally, retrieval of text data is not suitable for audio information retrieval due to the inability to look inside the audio objects. Also, common term-matching methods are not useful to audio information retrieval because of a lack of identical words in audio documents. So audio researchers try other approaches for audio information retrieval. Those technologies include automatic speech recognition (ASR) and audio effects. Automatic speech recognition would reduce the audio retrieval problem to the text retrieval problem. In this project, instead of using ASR, audio effects are applied to facilitate content-based audio retrieval. The audio retrieval is based on the audio's acoustical and physical characteristics, such as pitch, loudness, brightness, and bandwidth. An audio software package, Cool-edit, is used to convert audio signal

to text. A statistical method is then applied on the text data to extract the audio's loudness for the audio retrieval.

2. AUDIO SEARCH ENGINES

2.1 History of Search Engines

Before a search program is created, we need to know the history of search engines. Search engines are one of the primary ways that internet users find web sites. That's why a web site with a good search engine listing may see a dramatic increase in traffic. When the web first become available to the public, the only way to navigate it was by surfing with a text-only browser (Lynx) [2]. Users began with a known page and launched forth from the links it contained, browsing until they found something. When there were only a couple hundred HTTP servers on the Internet, this method could work. Most of these machines contained centrally managed collections of documents by students and professional researchers who were aware of the major repositories of information in their fields. But the rapid growth of the web both in the numbers and kinds of resources needed better ways of organizing information on the web. The first attempts to impose order on the web involved manually creating master lists of page links in hierarchical subject categories. Yahoo is a simple list of popular links. However, the growth of the web has proven too fast: the web size prevents much human control of the collection and indexing of web pages. People can create small databases of web resources, but gathering all of the information available is impossible.

To facilitate web information retrieval, automated means are required for large-scale indexing. One automated method is the use of robots. Robots are essential ingredients of all current web

search tools. Robots perform a wide variety of tasks with some related to the web such as

- discovering web pages for inclusion in a database,
- indexing web pages,
- measuring the size of the web, and
- maintaining a database of web pages by checking old links of updates and relocation.

2.2 Search Engine Technology

Understanding of how search engines work can help in designing search programs. A search engine includes three major elements. First is the spider, also called the crawler. The spider visits a web page, reads it, and then follows links to other pages within the site. This is what it means when someone refers to a site being spidered or crawled. Everything the spider finds goes into the second part of a search engine, the index. The index, sometimes called the catalog, is like a giant book containing a copy of every web page that the spider finds. If a web page changes, then this book is updated with new information. Sometimes it can take a while for new pages or changes that the spider finds to be added to the index. Thus, a web page may have been "spidered" but not yet "indexed".

Search engine software is the third part of a search engine. This is the program that sifts through the millions of pages recorded in the index to find matches to a search and ranks them in order of what it believes is most relevant.

Ranking algorithms determine how timely and relevant the search is. Web documents are generally ranked near the top of the priority list if the keyword or phrase of the query is found in the first few words of a web page using keyword searching. When using conceptual searching, the search engine will rank the highest documents with descriptive sentences from which concept descriptions can be created. Another method is to compute a ratio of the number of times a keyword occurs on a page to the total number of words on it. Pages with highest percentages are ranked at the top. Some web pages are also ranked according to how many links point to it. Search engines also develop abstracts that are based on title, headers, links, and initial words of key paragraphs appearing on a web page. These abstracts are listed among the results, but the ranking of such findings is more erratic than those based solely on the number of words or links. The above-mentioned methods are summarized in Table 1:

Table 1. Ranking methods used by search engines

Method	**Description**
Keyword	Higher priority if keyword is found in the first few words of the web page
Conceptual searching	Descriptive sentences
Ratio	Ratio of the number of times a keyword occurs on a page to the total number of words on it
Number of links	How many links point to it
Abstract	Based on title, headers, links, and initial words of key paragraphs appearing on a web page

Selecting a better search engine is important for a timely and relevant search. In order to design good search engines, we need to know desirable characteristics of search engines [2] (Table 2).

Table 2. Desirable search engine characteristics

Characteristic	Description
Boolean Search	Useful for experienced search specialists who know how to use operators such as AND, OR, NOT
Proximity	Search picks up keywords within a specified distance controlled by the user
Similarity	Search engine has the capability to substitute synonyms in place of keywords indicated by the user
Phrases	Search engine can handle strings of words as a complete phrase, not as individual keywords
Weights	The user can designate keywords with weights indicating which are most important
Clusters	Search engine has the ability to cluster results into groups of common topics
Auto Retrieve	Search engine has the ability to notify the user that information previously indicated is available

2.3 Traversal Spider Technology

Search engines do not actually scan the web when they receive requests from users. They look up the keywords or phrases in their own database, which are compiled using spiders that continuously travel across the web looking for new and updated entries and creating links to them. Generally, there are two spiders that participate in search [3]. One is called discover, and

the other is called harvest. The main procedures are

- the discover spider travels web pages and builds a URL database,
- the harvest spider checks the URL database first, then begins automatically revisiting each web page the discover spider discovered,
- the harvest spider extracts part or all of the text from the page, breaks it down into component words, integrates it into a master index of words from other web documents, which is the index database that the uses searches,
- the search engine matches query keywords with the index in the index database, and
- the users get query results from the search engine.

The basic idea behind the index is shown in Figure 1:

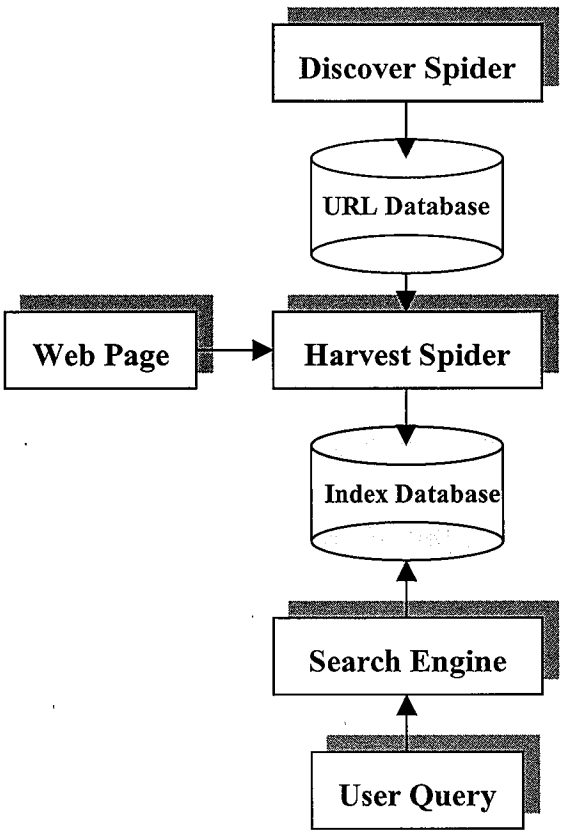

Figure 1. Process to index web pages

A spider is an essential part for all search engines, so it is also necessary to need to know good search spider characteristics, which are shown in Table 3.

Table 3. Desirable search spider characteristics

Characteristic	Description
Indexing	Full-text indexing to ensure search accuracy is preserved
Word Search	Capability to search for all individual words including very common and short words
Updating	Ability to perform continuous database updates
HTML Hyperlinks	Ability to handle and preserve hyperlinks in the documents and provide their URLs
Crawling Depth	Spiders search levels should be controllable by users
Exclusion	Spider should be able to be directed to specific web servers or internet areas of interest
Robot Exclusion Protocol	Spider should be able to observe the desire of Web servers to exclude visits
Recovery	Spider should be able to pick up search where it left off after a break or transmissions failure

There are two algorithms involved in search engines: one is depth-first search; the other is breadth-first search. Depth-first search is a generalization of preorder traversal search, while breadth-first search is like order-level search in the tree. The depth-first creates a relatively comprehensive database on a few subjects, while the breadth-first builds a database that touches more lightly on a wider variety of documents. For example, if we wish to create a subject-

specific index, it is better to select a known relevant group of pages to explore, rather than one initial document, thus employing a depth-first strategy with an upper bound on the degree of depth. This would prevent the search from being lost in inevitable digressions when exploring a largely unknown space in some depth. On the other hand, if an index needs to be built for sampling the wide variety of the web, it is better to choose a breadth-first strategy. The breadth-first algorithm is used in this project, because audio clips spread over a wide variety of web documents.

2.4 Spider Implementation

Java is used to build a search utility because of its extensive internet packages. A spider is an essential element in search programs. Furthermore, Java networking is a main function in spider programming and subsequent to building of audio search systems. Therefore, Java networking and spider programming are discussed first, then methods for building an audio search system are talked about.

2.4.1 Building Search Utilities

Before creating an audio search program, we need to know what is needed in building search utilities. The spider programming takes advantage of Java networking due to Java extensive internet packages. In this project, only audio information was retrieved from the web. Audio is

one of the multimedia types. Therefore we need to analyze Java networking and multimedia types on the web, which were used to build search utilities.

2.4.1.1. Java Networking and Multimedia Data Types

The powerful Java internet classes are one of the most compelling reasons to use Java to build an audio search system. There are two main classes provided by the java.net package for exchanging web data: the java.net.URL class and the java.net.URLConnection class. The java.net.URL class provides a very high-level interface to web transactions. The java.net.URLConnection provides more comprehensive control over web transactions. This class controls when the connection to the remote Web host is made, and what the request from the host consists of. Before using these classes, an internet connection needs to be set up for executing Java code.

MIME (Multipurpose Internet Mail Extension) content type can be used to control what kind of type you download based on your need. It is useful to download multimedia data types. Some of the useful methods and protocols are described below:

- URL: This can create a new URL from a string, from a relative string and a base URL, from a protocol string, a host string, and a filename string, or from a protocol, host, filename string, and a port number. One of these methods should fit your needs.
- GetURL: This returns the core URL object.

- OpenConnection: This method opens a connection to the remote web server and returns a URLConnection object.

- GetContent: Gets an object that represents the retrieved content. This method is useful in downloading something that can be represented as an image or as a string object. It is also useful if you're writing a Web browser and you want to create "handles" for various content types.

- GetFile: Get the filename string embedded in the URL.

- GetHost: Get the host name string embedded in the URL

- HTTP protocol: The HyperText Tranfer Protocol, the key data exchange protocol of the WWW, is what web browsers use to communicate with web servers.

- getDate: Returns the last modified date of the content.

- getContentlength: Returns the content length.

- guessContentTypeFromName: This is a useful utility function that guesses what MIME content type you are dealing with based only on the filename extension.

- GuessContentTypeFromStream: This method peers into the innards of a byte stream and guesses as to what MIME type that data is. This is a very handy utility function for anyone who is receiving data in an unknown format. This function allows you to get an idea of what kind of data a stream is carrying and to act accordingly.

2.4.1.2. Spider Programming

A spider is a key part in the search engine program. The spider class is an enumeration class that traverses the web starting at a given URL. It fetches HTML files and parses them for new URLs to look at. All the files that it encounters, HTML or otherwise, are returned by the nextElement() method as an URLConnection.

Part of the code is like:

```
Enumeration spider = new Searche.Spider("http://some.site.com/whatever/" );
while ( spider.hasMoreElements() )
  {
  URLConnection conn = (URLConnection)spider.nextElement();
  // Then do whatever you like with conn:
  URL thisUrl = conn.getURL();
  String thisUrlStr = thisUrl.toExternalForm();
  String mimeType = conn.getContentType();
  long changed = conn.getLastModified();
  InputStream s = conn.getInputStream();
  ...
  }
```

2.4.2 Audio Search Engine Construction

There are two steps to create an audio search engine: 1) detect audio files under one or more specific web sites using a JAVA detection program, and 2) download audio files to a local directory, if audio files are detected.

2.4.2.1 Audio Detection Program

An audio detection program searches all audio clips under a given URL. It includes 12 classes. The spider class needs 10 classes. Specifically, the audio detection program needs application and spider classes, and ten classes were used to create a spider. The class relationship is drawn as below:

Detection

→ *Application*	// superclass for embeddable applications
→ *Spider*	*// traversing the web starting at a given URL*
→ *HtmlObserver*	// callback interface for HtmlScanner
→ *Queue*	
→ *FlexVector*	// a vector class
→ *NoRobert*	// implementing the Robot Exclusion Standard
→ *Lruhashtable*	// a hashtable class
→ *HtmlScanner*	// an HTML scanner
→ *Pair*	// object-pair class
→ *GenericClonable*	// a parent for cloneable classes
→ *Utils*	// static utility routines

→ *Fmt* // a simple sprintf-like routine

The detection program can be run in the Java Applet. The class relationship is shown below:

Detection in Applet

→ *NullInputStream* // an inputstream that is always empty

→ *AprintStream* // more efficient printstream class

→ *TextCompOutputStream* // use a textComponent as a sink of an outputstream

→ *GuiUtils* // static GUI utilities

→ *Mainframe* // run an Applet as an application

→ *Widgets*

→ *Borderpanel*

In order to implement programs of search.java and downloadaudio.java, a package, named detection, is created, which includes the above listed classes and HtmlEditObserver and HtmlEditScanner.

Part of the detection program code is like:

```
public class search extends Searche.Application
  {

  static final String progName = "search";


  public static void main( String[] args )
    {
    (new search()).compat( args );
    }
```

```
private void usage()
   {
   err.println( "usage:  " + progName + " URL ... " );
   }

// List function
void list( String urlStr )
   {
   String base = Searche.Utils.baseUrlStr( urlStr );
   Enumeration as;
   try
      {
      as = new Searche.Spider( urlStr, err ); // from spider class
      }
   catch ( MalformedURLException e )
      {...}

   while ( as.hasMoreElements() )
      {
      // Java network--get connection
      URLConnection uc = (URLConnection) as.nextElement();
      URL thisUrl = uc.getURL();

      ...

      }
   }

}
```

The detection program searches all audio files on the web by giving it one or more URLs as arguments, then it enumerates the files reachable at or below those URLs. For example, when a URL http://www.geocities.com/~tnmc/audio.html is entered in the command line, the detection program is activated to list all audio files reachable from this web site, which are shown in Figure 2:

```
cmdtool - /bin/csh
lab44.eng.auburn.edu[mazhiqi]14: java search
http://www.geocities.com/~tnmc/audio.html
audio/x-wav          80940     snds/band.wav
audio/x-wav          33086     snds/ag-stiff.wav
audio/x-wav          66876     snds/ag-lucky.wav
audio/x-wav          38642     snds/ag-crazy.wav
audio/x-wav          48000     snds/ag-shampoo.wav
audio/x-wav          41138     snds/ag-knees.wav
audio/x-wav         142262     snds/bb-5yrs.wav
audio/mpeg           71252     snds/bb-5yrs.mp3
audio/x-wav         105608     snds/bb-xcited.wav
audio/mpeg           52861     snds/bb-xcited.mp3
audio/x-wav          60722     snds/bb-milk.wav
audio/mpeg           30294     snds/bb-milk.mp3
audio/x-wav          64190     snds/bh-jerkoff.wav
audio/x-wav          26176     snds/bh-spray.wav
audio/x-wav         122342     snds/bh-lanolin.wav
audio/x-wav          31086     snds/bh-businesses.wav
audio/x-wav          18136     snds/bh-man.wav
audio/x-wav         392736     snds/bh-dolphins.wav
audio/x-wav          19930     snds/bl-dudeabides.wav
audio/x-wav          68316     snds/bl-beverage.wav
audio/x-wav          47312     snds/bl-bowling.wav
audio/x-wav          47400     snds/bl-rug.wav
audio/x-wav          20988     snds/dm-bloodtrail.wav
audio/x-wav          82724     snds/dm-counting.wav
audio/x-wav          81932     snds/dm-maritini.wav
audio/x-wav          63436     snds/dm-live.wav
audio/x-wav          81516     snds/tfm-cream.wav
audio/x-wav         110620     snds/tfm-bastard.wav
audio/x-wav         101206     snds/tfm-otherdads.wav
audio/x-wav          86796     snds/tfm-noisaid.wav
audio/x-wav          89080     snds/tg-passport.wav
```

Figure 2. Audio detection results

Also the detection program can run within the JAVA applet as shown in Figure 3:

```
Applet Viewer: Searche.ApplicationApplet
Applet
search http://www.eng.auburn.edu/users/mazhiqi/mcse/system/pmusic
audio/x-wav          13356     1_FEMALE.WAV
audio/x-wav          17452     2_FEMALE.WAV
audio/x-wav          16428     3_FEMALE.WAV
audio/x-wav          30764     CAT_MRO.WAV
audio/x-wav          14399     C_FEMALE.WAV
audio/x-wav          15404     C_MALE.WAV
audio/x-wav          20714     FEMALE_L.WAV
audio/x-wav          14380     H_FEMALE.WAV
audio/x-wav          10476     H_MALE.WAV
audio/x-wav          64556     LAPWING.WAV
audio/x-wav          29612     LAUGHTER.WAV
audio/x-wav          16060     WHY_DID_.WAV
audio/x-wav          14380     YRS_F_.WAV
audio/basic         522268     birthday.au
audio/basic        1076282     newyear.au
search done.
```

Figure 3. Audio search result from applet

2.4.2.2 Audio-Download Program

For the audio-download program, in addition to the classes used by the detection program, it needs two more classes: HtmlEditObserver and HtmlEditScanner. HtmlEditObserver is a Callback interface for HtmlEditScanner. Clients of HtmlEditScanner implement this in order to get URLs passed back to them. HtmlEditScanner is like HtmlScanner but it lets you make changes to the URLs in the HTML stream you are scanning. The regular scanner class lets you define callbacks that get called with the URLs; in this version, you can return URLs from the callbacks, and they are inserted into the stream in place of the old URLs.

Part of the HtmlEditObserver code is like:

```
Public interface HtmlEditObserver

    {
    public String editAHREF(String, URL, ...)
    public String editBODYBACKGROUND(String, URL ...)
    public String editFRAMESRC(String, URL ...)
    public String editBASEHREF(String, URL ...)
    public String editLINKHREF(String, URL ...)
    ...
    }
```

Part of the HtmlEditScanner code is like:

```
Public class HtmlEditScanner extends FilterInputStream implement
```

```
Searche.HtmlEditObserver

{
// Constructor
public HtmlEditScanner(InputStream, URL, ...) {...}
// Add an extra observer to this editor
public void addObserver(Searche.HtmlEditObserver, ...) {...}
// Callback from HtmlScanner
public void gotAHREF(string, URL,...) {...}
public void gotLINKHREF(string, URL,...) {...}
...
}
```

When given one or more URLs as augments, the audio-download program downloads only audio clips, which are the files reachable at or below those URLs

The audio-download program downloads audio files to a local directory, which is used to construct an audio database. Because the most popular audio clips on the Web are .au and .wav files, the audio-download program downloads just those kinds of files. Figure 4 shows the results from the audio-download program.

```
cmdtool - /bin/csh
hellcat.eng.auburn.edu{mazhiqi}15: java downloadaudio -v
http://www.eng.auburn.edu/~mazhiqi/mcse/system/pmusic
Downloading--Audio:  http://www.eng.auburn.edu/~mazhiqi/mcse/system/pmusic/1_FEMALE.WAV
Downloading--Audio:  http://www.eng.auburn.edu/~mazhiqi/mcse/system/pmusic/2_FEMALE.WAV
Downloading--Audio:  http://www.eng.auburn.edu/~mazhiqi/mcse/system/pmusic/3_FEMALE.WAV
Downloading--Audio:  http://www.eng.auburn.edu/~mazhiqi/mcse/system/pmusic/CAT__MEO.WAV
Downloading--Audio:  http://www.eng.auburn.edu/~mazhiqi/mcse/system/pmusic/C_FEMALE.WAV
Downloading--Audio:  http://www.eng.auburn.edu/~mazhiqi/mcse/system/pmusic/C_MALE.WAV
Downloading--Audio:  http://www.eng.auburn.edu/~mazhiqi/mcse/system/pmusic/FEMALE_L.WAV
Downloading--Audio:  http://www.eng.auburn.edu/~mazhiqi/mcse/system/pmusic/H_FEMALE.WAV
Downloading--Audio:  http://www.eng.auburn.edu/~mazhiqi/mcse/system/pmusic/H_MALE.WAV
Downloading--Audio:  http://www.eng.auburn.edu/~mazhiqi/mcse/system/pmusic/LAPWING.WAV
Downloading--Audio:  http://www.eng.auburn.edu/~mazhiqi/mcse/system/pmusic/LAUGHTER.WAV
Downloading--Audio:  http://www.eng.auburn.edu/~mazhiqi/mcse/system/pmusic/WHY_DID_.WAV
Downloading--Audio:  http://www.eng.auburn.edu/~mazhiqi/mcse/system/pmusic/YES__F_.WAV
Downloading--Audio:  http://www.eng.auburn.edu/~mazhiqi/mcse/system/pmusic/birthday.au
Downloading--Audio:  http://www.eng.auburn.edu/~mazhiqi/mcse/system/pmusic/newyear.au
hellcat.eng.auburn.edu{mazhiqi}16:
```

Figure 4. Audio download results

The downloaded audio files are located in the assigned directory, which, in my case, is the .../audiofile subdirectory, which is shown in Figure 5. Another attempt to download a web content under a URL is to use an applet, but the content contains a full text format, where a filtering operation has to be applied to find audio files.

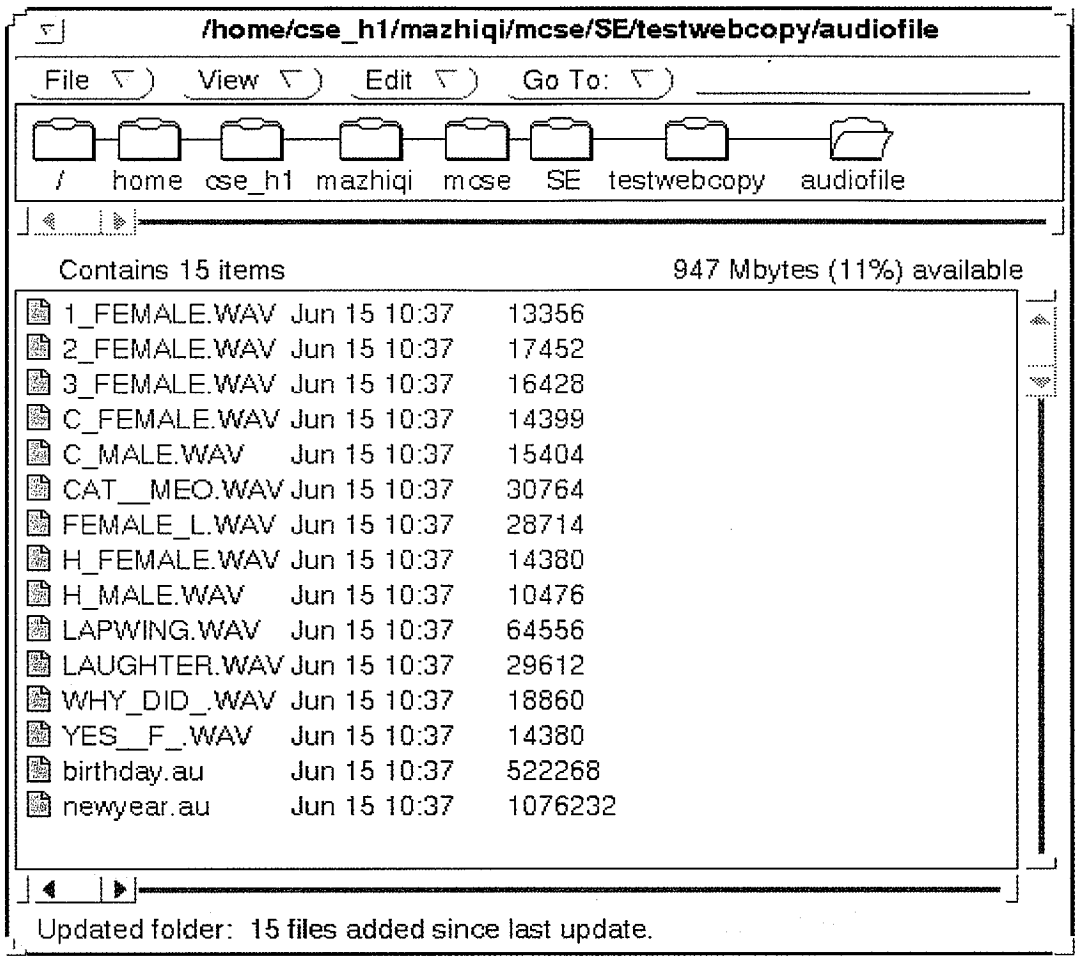

Figure 5. Downloading audio files to a specific directory

3. AUDIO INFORMATION RETRIEVAL

3.1 Problems in Audio Document Retrieval

Unlike text information retrieval, an audio information retrieval needs to decode its content before processing. The term-matching in text retrieval is not suitable to the audio information retrieval due to lack of identified keywords in the audio. Content-based audio retrieval could solve problems in the audio retrieval. Audio content-based retrieval is different from text retrieval, and audio index is more complicated than text index. So we need to discuss the audio content and index, and compare text retrieval with audio retrieval.

3.1.1 Audio Content and Index Issues

Attempts to retrieve spoken documents encounter similar problems to those associated with text document retrieval, but there are further important issues that need to be considered. Most obvious among these is that the contents of spoken documents are unknown. In this project, Cool-edit software package, is used to convert audio digital signals to ASCII data, where a statistical analysis is applied on the data to extract audio content features. The speech recognition system may also attempt to perform a full transcription of the contents of the documents using a large vocabulary recognizer. In speech recognition, the indexing vocabulary is limited to that of the recognizer. Large vocabulary systems are now available with vocabularies approaching 100,000 words, but this should be compared to the 500,000 word vocabularies encountered in

text retrieval systems. Many words, particularly proper nouns, can not be recognized collectively by either recognition system since they are outside the domain of its vocabulary. This creates a significant search problem, which does not exist in text-based systems where new document terms are added into the file structure.

Additionally, speech recognition is not completely reliable. Even the very best systems will make recognition errors, often arising from variable pronunciation or events outside its domain. The recognizer typically maps out-of-vocabulary words to something in its existing vocabulary, which will result in a recognition error. Short words are more susceptible to recognition errors than longer ones, both because of their inherent greater confusability and the greater tendency to poor articulation. Actual recognition is dependent on these and many other factors. Of particular significance is the degree of spontaneity in the speech, such as the amount of disfluency, the formality of the linguistic structure, and the clarity of articulation. The effect of these factors is evidenced by the difference in recognition performance between formal dictation, where over 90% of the words can often be recognized correctly, and informal conversational speech, where performance may fall to less than 40%. In any case it is important to realize that good text search terms may not be the best approach in the speech domain because of their acoustic properties. Therefore, speech recognition is not the best approach to realize content-based retrieval. Other approaches are needed to improve audio content-based retrieval. In this project, Cool-edit was used to extract its content with audio signal processing.

3.1.2 Comparison with Text Information Retrieval

There are some similarities between term identification for text and spoken document retrieval. For example, in the text case there may be false alarms on searches with multiple senses. If one of these terms is present in a query it will match any occurrence in a document regardless of the sense used in each case. This has been shown to have a minimal effect on retrieval effectiveness except for very short queries. There may also be misses on search terms which occur in the documents as synonyms of a query term; although, of course, there are techniques designed to overcome this problem. Finally, there may be query-document term matching errors, arising from spelling errors in the documents or query, or inappropriate term stemming.

These problems are also potentially present in spoken document retrieval. However, there are two additional sources of potential errors similar in effect to those just described. Acoustic false alarms, which occur when the speech recognizer hypotheses the presence of a term which actually doesn't exist, and acoustic misses, where the occurrence of a term is not detected by the recognizer. Adding more search terms to the query may offset all these sources of search error.

3.2 Types of Audio Files on the Web

The understanding of audio types on the web can help in building an audio database. Unlike images, audio clips not only let you see the information, but also hear the information. Also, putting an audio file on the web that embeds an actual file must be downloaded and played [4]. Different audio types need different audio players. Therefore, analysis of audio types on the web is necessary.

Different types of audio files on the web are shown in Table 4. Note that choosing an audio player is like picking the right kind of film for your camera.

Table 4. Audio types

Format	Description
AU	Developed by Sun, AU files tend to be small, compact, and easy to download. Also AU files work on all types of computers. The software required to listen to .AU files is built into Netscape and internet Explorer.
WAV	Microsoft Windows audio format. WAV files tend to be of higher quality than the AU format. WAV files are ideal if you expect most visitors to your Web site to be using Windows.
RA	RealAudio is the defining standard for streaming sound on the Internet. RealAudio is usually only used for professional Web sites
MIDI	The MIDI sound format is popular because it works across platforms and stores sounds without losing any quality.
AIFF	The Macintosh audio file format. Not very common on the Web, AIFF is also fully supported by most common browsers, so using them is a viable alternative

The population of audio types on the web is shown in Figure 6 [5], where .au and .wav audio files are most popular on the web. Therefore, this project focuses on the two audio data types.

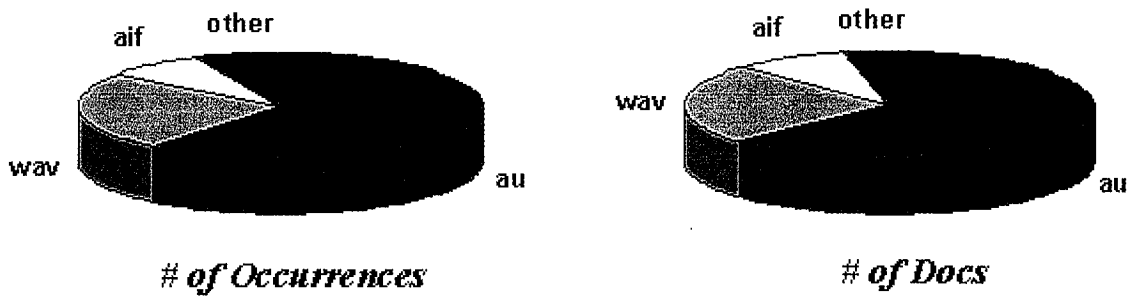

Figure 6. Population of various audio types on the web

3.3 Audio Information Retrieval Skills

Traditionally, most of the information retrieval on the web is text retrieval, while audio retrieval needs to concern its content. Audio content indicates its physical and acoustical characteristics, which can be extracted by digital signal processing. The best approach for audio information retrieval may combine text retrieval with audio's acoustical attributes.

3.3.1 General Information Retrieval

There are a couple of search methods used for information retrieval, but various methods affect searching timely and relevantly. A variety of search methods are shown in Table 5:

Table 5. Search methods

Method	Description
Full Text	Complete data and information Difficult to assess large documents
Keywords	Narrow user query Requires insight into document contents
Synonyms	Refines search query May return imprecise documents
Abstracts	Very precise definitions Requires creating meaningful abstracts
Description	Precise and unambiguous Does not involve document content
Context-based	Highly accurate Requires new multimedia technologies

In the simplest method, the full document text is retrieved. Another method uses one or more keywords, which classifies the document as an appropriate match. This method narrows the search to only documents containing the keywords, but also requires some insight about information retrieval. However, most text retrieval programs cannot provide that capability. One of the highly accurate methods is context-sensitive searching, which is used for multimedia information retrieval.

3.3.2 Audio Information Retrieval

Audio files are classified by pitch, loudness, duration, and so on. Research has focused on algorithms for extracting audio structures from a sound or on indexing audio bases using neural nets. However, these methods are difficult in specifying which elements of a sound are important and which should be ignored. Some features are used to search sounds, which include acoustical perception, onomatopoeia, simile, and subjective description. Acoustical perception refers to description of sounds. Subjective description involves the use of words to provide a special meaning to the sound. In an audio search engine, this feature requires that the system be trained to understand the meaning of such descriptions, which implies AI content.

All of these features are useful in searching for specific sounds, but researchers also agree that they should be used in combination with traditional keyword or text phrase queries. To implement such an audio search engine, it is necessary to reduce a sound to a set of parameters for audio information retrieval.

3.4 Implementation

The following three steps are conducted for implementing content-based audio retrieval: 1) a Cool-edit package is used to convert audio digital signals to ASCII text data; 2) a statistical method is applied to analyze the ASCII data to extract audio content features such as sloudness, one of the typical acoustical features; and 3) a web graphic user interface is constructed.

3.4.1 Conversion of Speech to Text for Content-Based Retrieval

When the audio has been classified to contain speech, it is very useful to provide a text version of the speech. Conversion of speech to text is a necessary part of a comprehensive audio content-based system [6]. Conventional information retrieval (IR) research has been mainly based on computer-readable text [7, 8], and is familiar to many through the popular web search engines such as Lycos, Yahoo, and AltaVista. The Classic IR problem is to locate desired text documents using a search query consisting of a number of keywords. Even though powerful IR algorithms are available to text, it is clear that for audio, or multimedia in general, common term-matching approaches are useless due to the simple lack of identical words in audio documents [9].

Content-based retrieval of audio means a variety of things [1]. At the lowest level, a user could retrieve a sound by specifying the exact numbers in the excerpt of the sound's sampled data. This is analogous to an exact text search. The next higher level is that the retrieval would match any sound containing the given excerpt without concern of the data's sample rate, quantization, compression, and so on. This is analogous to a fuzzy text search and can be implemented using correlation techniques. A further higher level is that the query might involve perceptual properties of the sound and acoustic features that can be directly measurably. Some of the perceptual properties of a sound, such as pitch, loudness, and brightness, correspond closely to measurably features of the audio signal, which provide fields for these properties in the audio database. In order to implement audio content-based retrieval, the first step is the conversion of speech to text. Cool-edit was used to convert the audio signal to ASCII text data.

3.4.1.1 Cool-Edit Software

Cool-edit is a digital sound processor, which converts the speech signal to ASCII text. Cool-edit operates directly on the samples of the audio waveform, and the user can specify locations and values numerically or graphically [6]. It can paint with sound such as tones, piece of sounds and voices and miscellaneous noises, sine waves and sawtooth waves, pink noise, white noise, and brown noise. Cool-edit gives a wide variety of special effects to touch up your sounds, which are reverberation, noise reduction, echo and delay, flanging, and filtering. Cool-edit is both flexible and powerful. Scientists, musicians, multimedia authors, broadcast professionals, telephony system engineers, and sound enthusiasts have chosen the cool-edit as their preferred sound processor. It provides the following features:

- Perform frequency analyses on your signals,
- Convert file and sample types,
- Edit mono or stereo files up to 1 gigabyte in size,
- Touch up files with functions like filter, amplify, compress, stretch, and noise reduction,
- Record files from a CD, keyboard, or any signal played through your sound card,
- Add fantastic effects with reverb, delay, echo, flanger, distortion, and envelope functions,
- Generate silence, noise (white, pink, or brown), DTMF signals, and tones,
- Edit multiple files simultaneously,
- Merge files together with paste overlap and paste modulate functions,

- Use the cue list and play list to cue up and play the segments,
- Create named presets to store your favorite processing function settings, and
- Create scripts for automated processing and batch operations.

3.4.1.2 Digital Signal Processing

Before sampling audio digital data for extracting audio content features, we need to introduce speech digital signal processing. Speech signal processing is defined as information processing based on speech signal analysis, and it has application in speech synthesis and speech recognition [10]. Audio information can be observed and processed in the form of sound waveforms. A physical sound wave can be picked up by a microphone as an electrical waveform for processing. Speech waveforms are usually converted to digital form for processing. To better understand what the content of audio means and how audio features can be obtained, it is first useful to know audio digital signal processing. Here we just discuss the signal and sampling.

Signals (or Waves):

Waves in the context of Cool-edit are sound waves (Figure 7). A sound wave can be considered as the how the air pressure on the ear changes over time. When you hear a loud sound, the pressure on your eardrum is greater, and it vibrates harder. A wave is a convenient representation of how the sound level varies over a time interval. The illustration is a sine wave of a constant pitch (Figure 7). It shows the sound pressure oscillating from low pressure to high

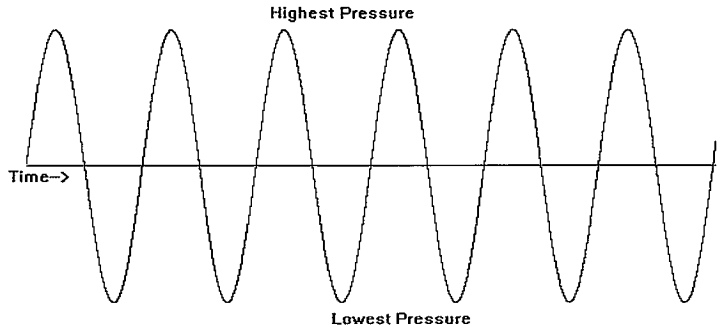

Figure 7. Regular audio waveform

pressure. In natural systems, this motion follows the path of a sine wave. The wave here is of a constant frequency and constant amplitude. If you choose any time along the time axis and draw a vertical line up and down, there will be exactly one spot where the wave crosses this vertical line. This is because a wave can have only one value at any instant in time. Therefore, a waveform is depicted as a line that can vary up and down freely, going from left to right, and has exactly one value at any time along the time axis.

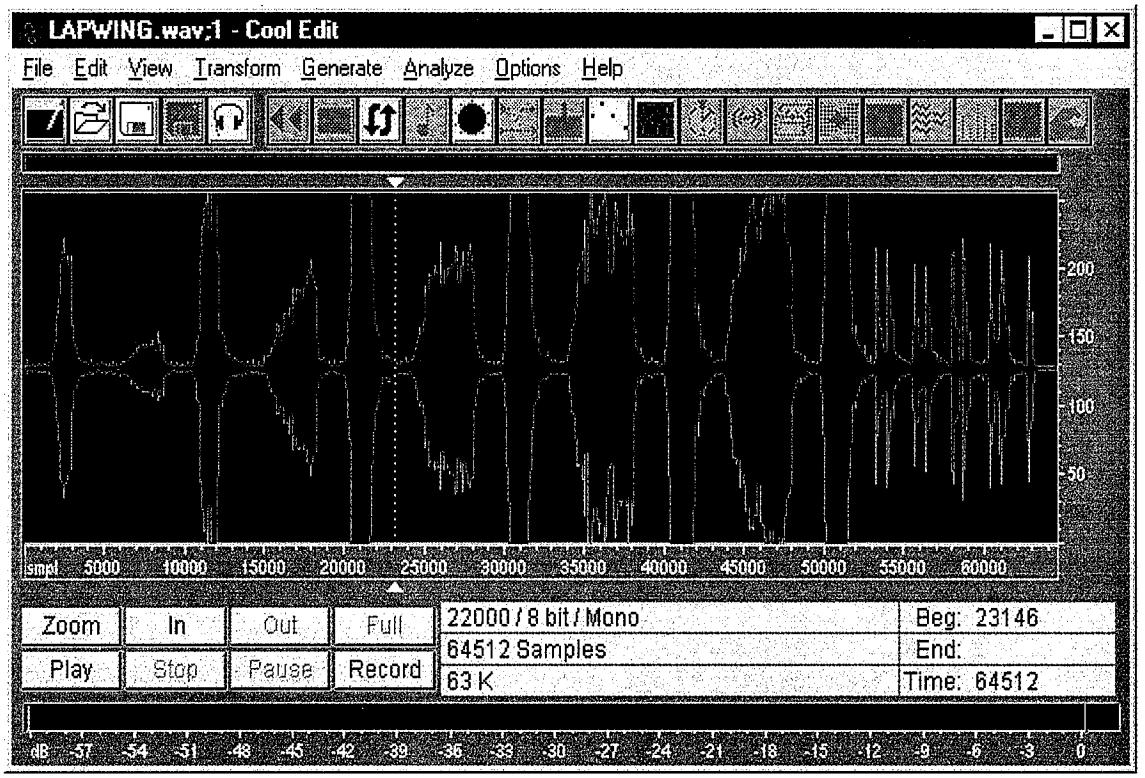

Figure 8. Waveform of the LAPWING.wav

Waves in the natural world are continuous (Figure 8), which means that no matter how much you "zoom in" to the waveform, or no matter how small of a time interval you look at, there are an infinite number of values needed to represent the progression of the waveform during that interval. Cool-edit's normal waveform view displays waveforms such as bird voice [Figure 8].

Sampling:

Sampling is the first phase in converting an analog signal to a digital form. Sampling means approximating a waveform to a discrete signal, that is, a sequence of samples that are discrete in time [10]. In the processing of digital signal data, it is impossible to work with an infinite amount of data, which is what would be required if a continuous wave is to be represented, so at every possible instant in time we would have the value of the waveform at that instant.

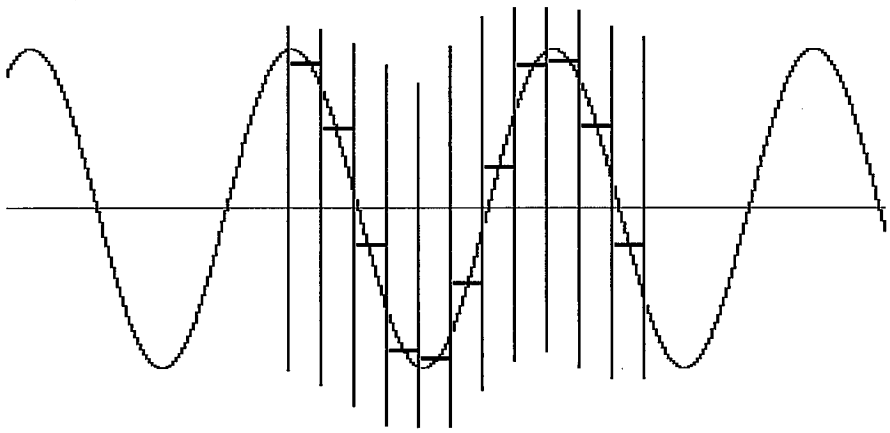

Figure 9. Sampling of audio

For this reason, it is necessary to sample the data from the audio digital signal. The sampling consists of breaking up the waveform at constant intervals, and representing all values in that interval by a single value (Figure 9). By dividing the waveform, one second of audio can now be represented by a finite number of values. The sampling rate needs to be determined when sampling data from the audio. The sample rate is the number of divisions taken per one second of audio, as you can see by the graph (Figure 9). The wave above (Figure 9) can be thought of as going from values between 100 (at the top) and -100 (at the bottom). Thus 11 samples can be

given as roughly 93.7, 51.5, -22.1, -89.4, -97.6, -48.0, 25.7, 92.1, 93.9, 54.5, and -21.1.

3.4.1.3 Audio Content Analysis

Though many systems exist for content-based retrieval of images, little work has been done on the audio portion of the multimedia system. The content of audio can be viewed from two angles: 1) the measurable properties such as amplitude, and 2) the properties of human cognition such as subjective loudness or harmony [11]. Segmentation of audio is needed to analyze audio content. A segmentation of audio must be performed based on the recognition of acoustical content both in the temporal and the frequency domains. For example, an analysis of amplitude (Loudness) statistics belongs to the temporal domain whereas an analysis of pitch or frequency patterns belongs to the frequency domain. In this project, one of the typical audio acoustical features known as sloudness, similar to loudness, was used to represent audio content.

3.4.1.4 Sloudness

Many audio and multimedia applications would benefit if they could interpret the content of audio rather than relying on description of keywords [6]. One might access sound via physical attributes, which are commonly understood acoustical characteristics, such as loudness, brightness, and pitch.

Sloudness is defined as being similar to loudness, one of the typical audio acoustical

characteristics. How is sloudness determined? First of all, the Cool-edit software was used to convert the speech signals to ASCII text, then a statistical method was used to analyze these data. For example, if the data from an ASCII text file is 10, 30, 40, 20, the mean of these data is 25, which is used to represent Sloudness. For a bird voice (Figure 10), the sample rate is 22000, and 64512 samples are taken to statistically calculate sloudness. The sloudness is 118, which is the mean value of 64512 samples.

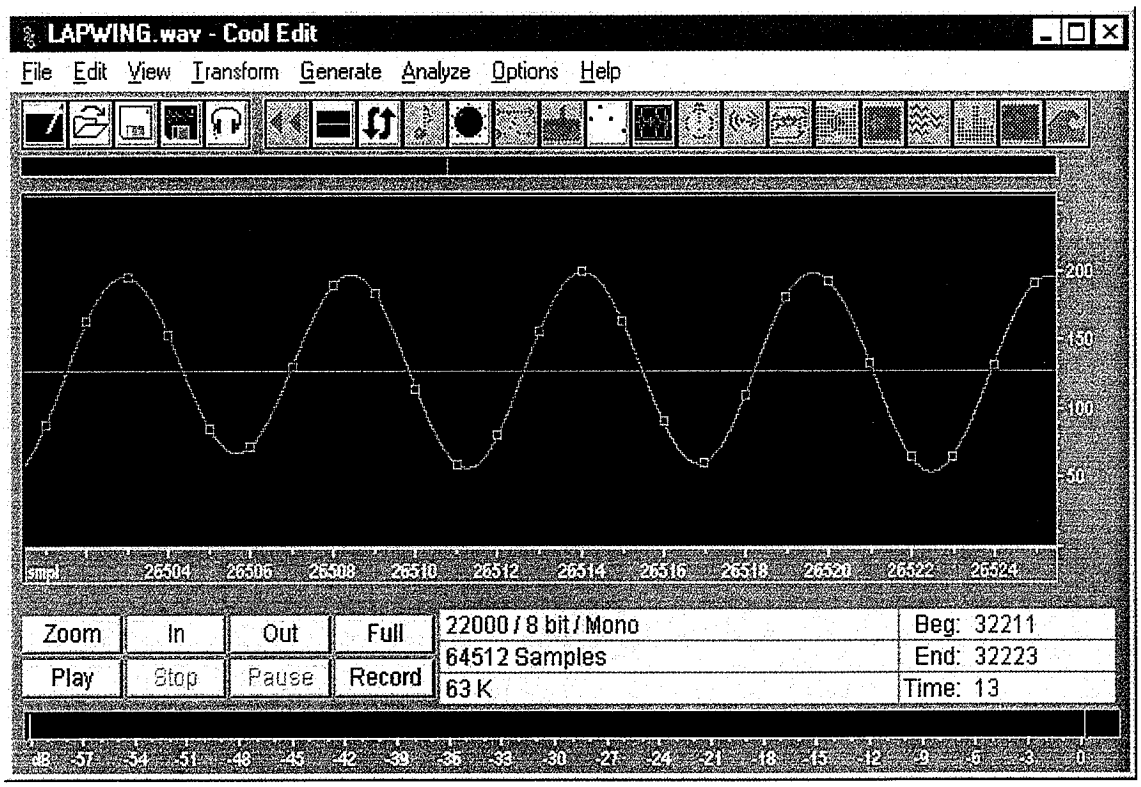

Figure 10. Sampling of file LAPWING.wav

4. A Prototype of Web Audio Information Retrieval System

The prototype of web information retrieval system consists of two parts. One is web audio download; the other is audio database on the web. The detection program and audio-downloaded program are used to detect and download audio information on the web. The audio database is used in the content-based audio retrieval. The system structure, system interface, experimental results, and performance evaluation are discussed below.

4.1 System Structure

A spider visits web pages. A detection program searches and downloads audio clips reachable from the web pages. The audio clips and URLs are storded in a database for queries by sloudness and names. Figure 11 shows the structure of the audio search engine system, which possesses the following functionalities:

- automates collection of audio information from the Web,
- performs search using innovative audio content-based techniques, and
- displays query results.

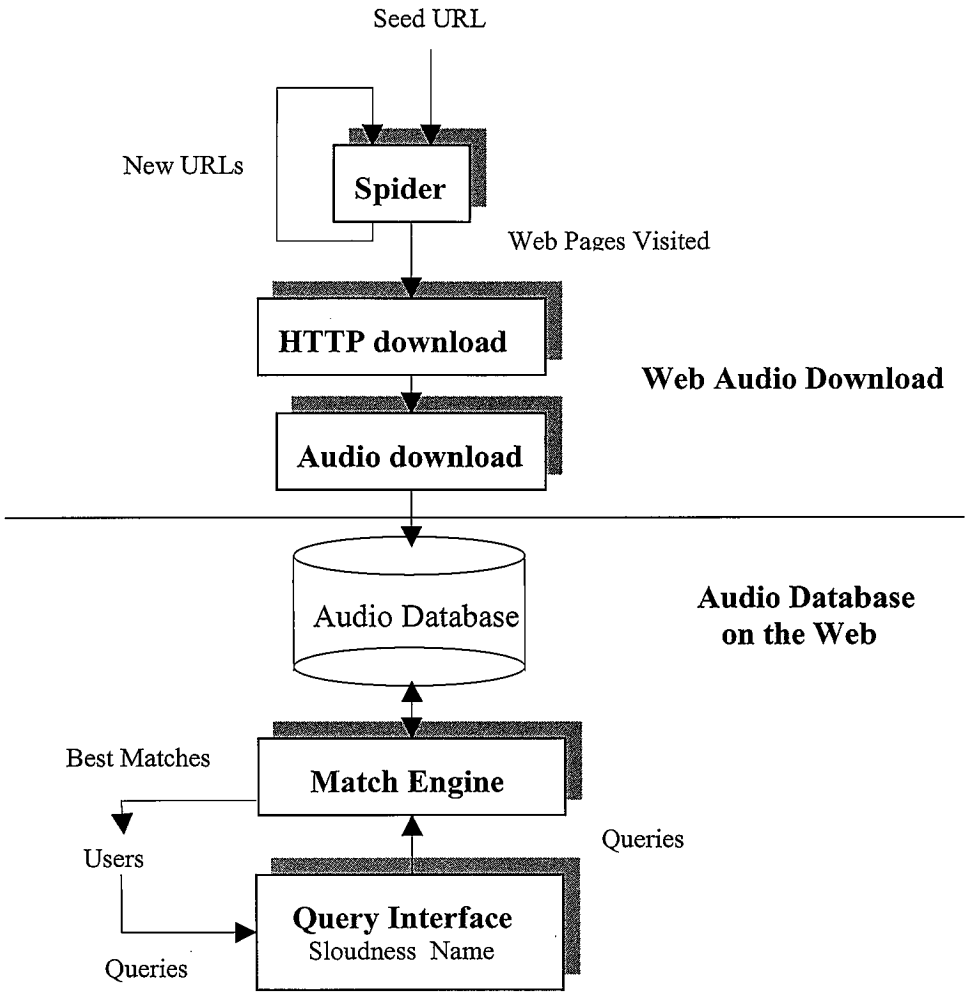

Figure 11. System structure

This project used Oracle 7, Pro*C, CGI, SQL, and Perl to implement an audio retrieval system. The process of building a prototype of audio retrieval is shown in Figure 12. It converts the audio signal to text first, extracts audio features, such as sloudness, creates an oracle audio feature database, and finally queries audio clips through the search engine.

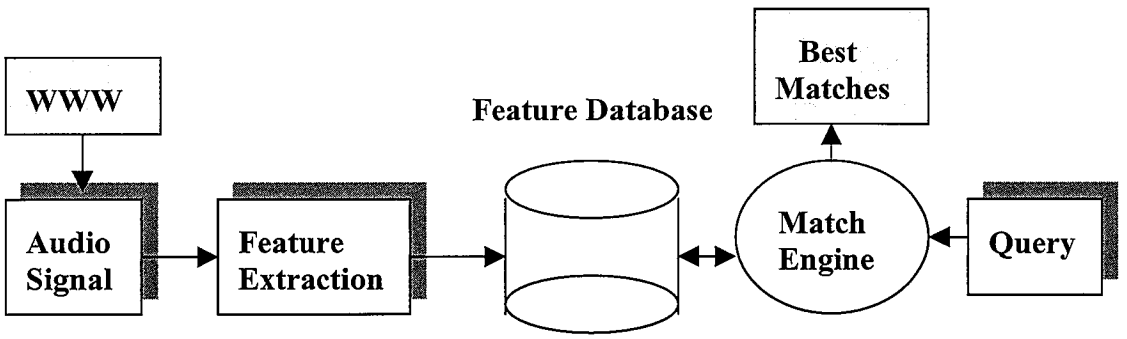

Figure 12. Process of audio retrieval

The audio database is usually designed to work with large numbers of audio files, and benefits from content-based capabilities. Typically, the database contains the following audio file attributes, user attributes, and acoustical features.

- Audio file attributes: URL, name, sample rate, sound file format, number of channels, creation date, and analysis date,
- User attributes: keywords and comments, and
- Acoustical features: loudness[mean, stdev], pitch [mean, stdev], and tone [mean, stdev].

In this project, audio features consisted of sloudness, audio name, audio type, audio size, audio

category, and its URL, which were used to build the audio database. The users retrieve audio information through their sloudnesses and file names.

4.2 System Interface

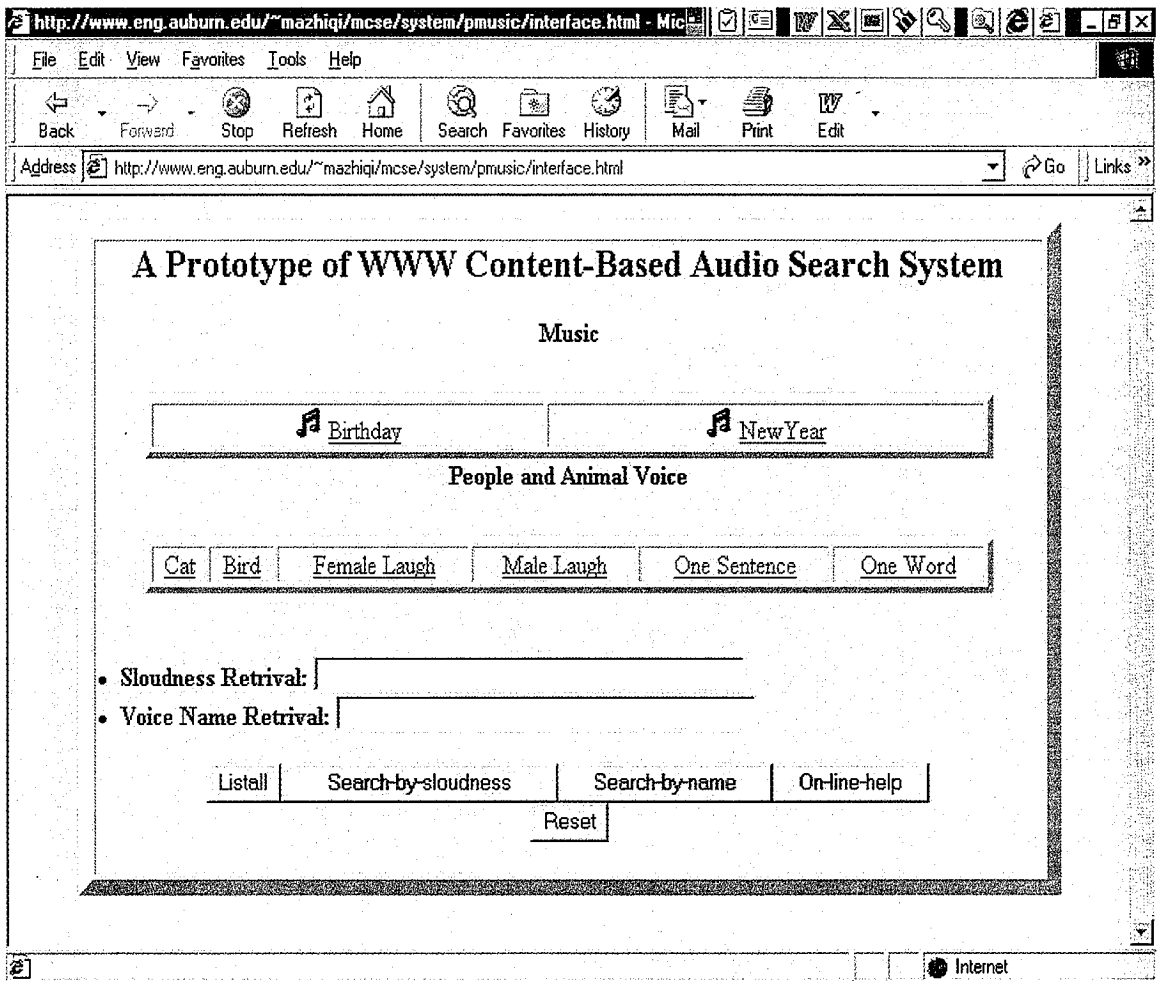

Figure 13. The System Interface

The interface provides following functions (Figure 13):

- Listall: lists all audio clips,
- Search by sloudness: needs input of audio sloudness,
- Search by audio name: needs input of audio name, and
- On-line help.

In this project, the users can scan all of audio clips in the database through Listall, search audio clips on the web through Search-by-sloudness and Search-by-name, and play them according to users needs. However, the audio clips that can be reached are limited to the database created by programmer.

4.3 Experimental Results

A prototype of WWW content-based audio search system is built to search audio clips on the web through audio sloudness. The search results are shown in Figures 14, 15, 16, and 17. Figure 14 showed one URL list of searching band audio clip with sloudness 23. Figure 15 showed three URLs list of searching a couple of speech files with sloudness 222. Figure 16 showed more than 13 URLs for alphabet and number sound clips with sloudness 85. Figure 17 showed 13 URLs list of searching bird sound clips with sloudness 118. Therefore, it appears that same kind of audio effects such as bird sound come to one search group, which implies sloudness might be used to classify web audio information.

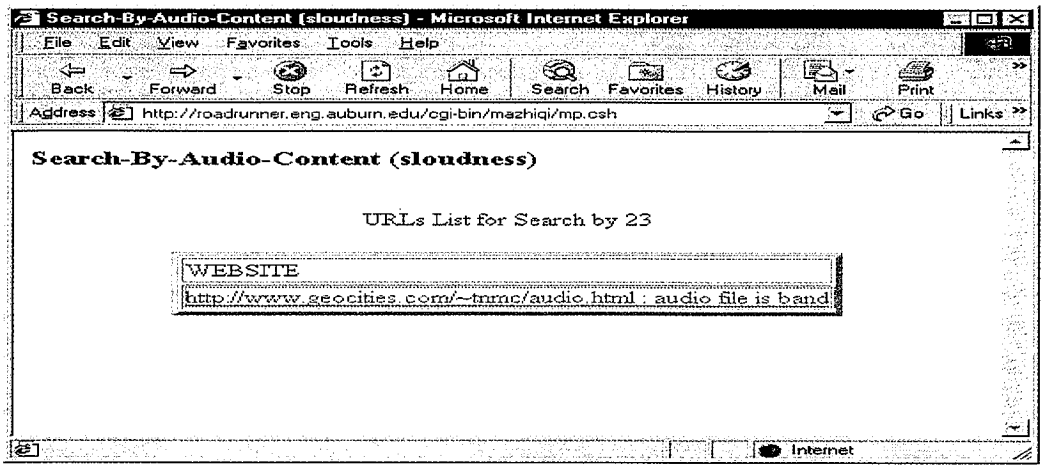

Figure 14. Search results by sloudness (23)

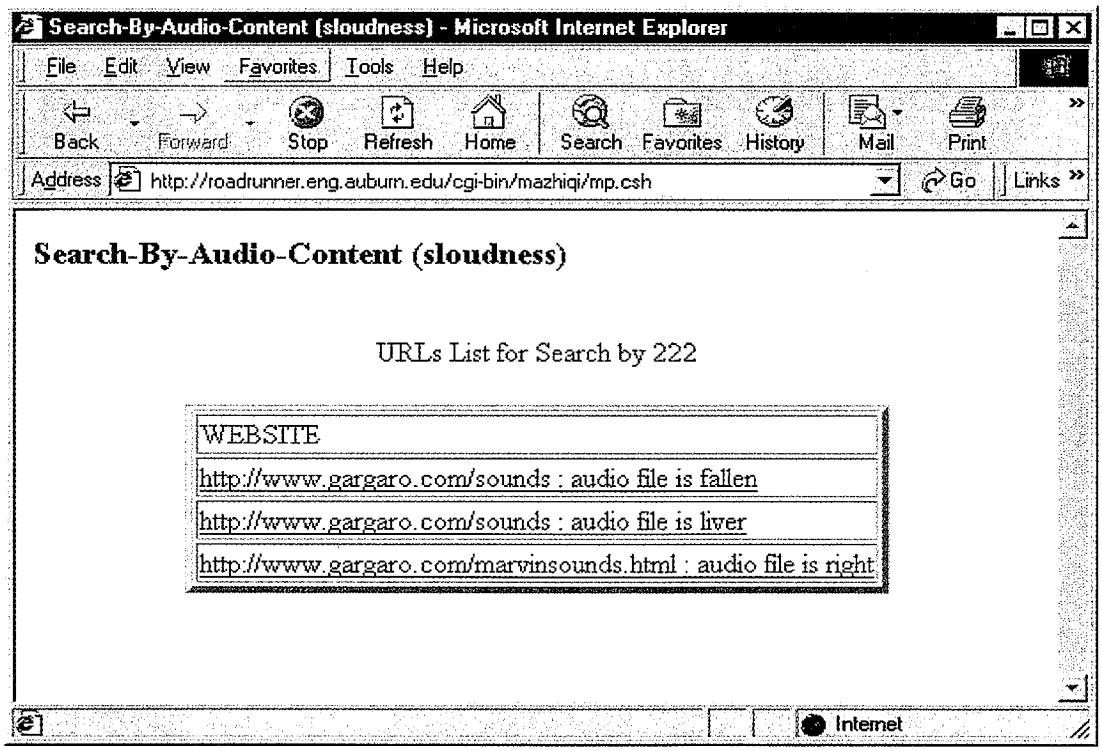

Figure 15. Search results by sloudness (222)

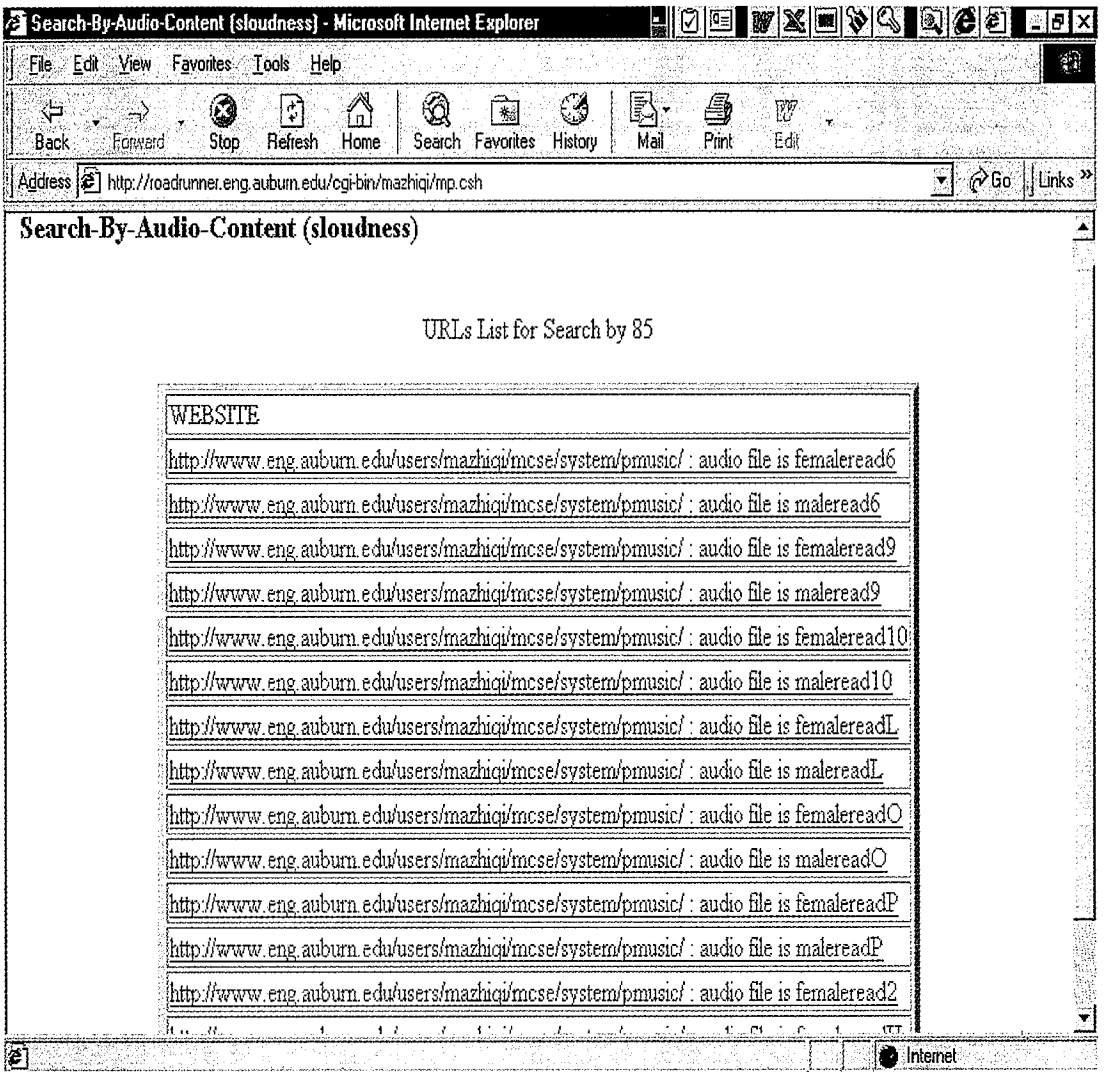

Figure 16. Search results by sloudness (85)

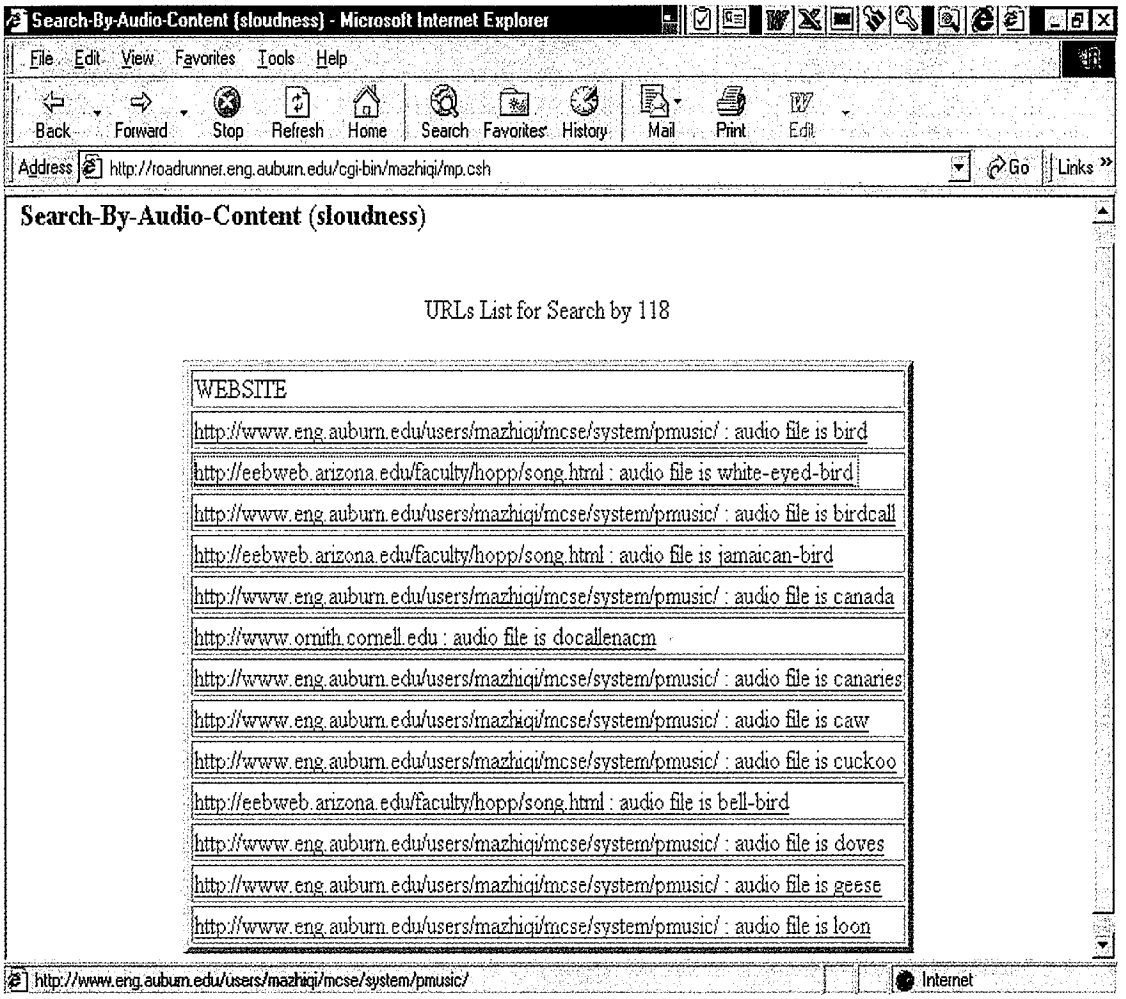

Figure 17. Search results by sloudness (118)

4.4. Performance Evaluation

In this project, audio clips can be detected and downloaded to a specific directory. That is, detection program detects audio clips, and audio-downloaded program automatically downloads audio clips if audio clips are exist. The prototype of WWW content-based audio search system tries to model a content-based audio retrieval. The system provides the functions of search-by-sloudness and search-by-name. The users can search audio clips through audio content (sloundess) and names. It is found that files with similar audio effects, they are grouped together. For example, alphabet and number reading come to one audio group with sloudness 85, and bird sounds go to another audio group with sloudness 118. It appears that sloudness could be used to classify web audio information. The performance might be better if the implementation of audio content-based retrieval can be added by audio speech recognition.

4. CONCLUSIONS

A large component of web content consists of audio information such as speech, music, and sound. The audio information on the web proves to be highly volatile. It requires a highly efficient automated search system in cataloging and retrieving information. The system regularly traverses the web, detects audio information, and indexes the web in such a way as to allow efficient and effective search and retrieval.

This project is a trial of content-based audio information retrieval on the web. A prototype web system is built to facilitate audio information retrieval. The system can detect audio clips, and download them if audio clips exist. The Cool-edit is used to convert the downloaded audio clips to ASCII text. A statistical method is applied on the text data to extract audio sloudness. The audio database is built to include audio type, name, size, sloudness, and URLs. The graphic interface is constructed for query by sloudness and names. It provides fundamentals to further develop audio searches in the future. The search would be improved if the implementation of the content-based retrieval can be added by an audio speech recognition, and audio clips are classified according to subjects such as music, speech, and so on.

5. REFERENCE

1. Erling Wold, Thom Blum, Douglas Keislar and James Wheaton. Content-Based Classification, Search, and Retrieval of Audio. IEEE Multimedia, pp. 27-36, Fall 1996.

2. Bohdan O.Szuprowicz. Search Engine Technologies for the World Wide Web and Intranets. Prentice Hall, 1997.

3. Susan Maze, David Moxley, and Donna J. Smith. Authoritative Guide to Web Search Engines. New York, Neal-Schulman, 1997.

4. Andy Shafran. Creating Your Own Web Pages. Second edition, Que Corporation. pp.195-221, 1997.

5. Lawrence A. Rowe. An Investigation of Documents from the World Wide Web. Fifth International WWW Conference, Paris. May 7, 1996. Http://bmrc.berkeley.edu/people/larry/talks/www5/sound.gif.

6. Erling Wold, Thom Blum, Douglas Keislar, and James Wheaton. Classification, Search, and Retrieval of Audio. Http://www.musclefish.com/cbr.html.

7. Van Rijsbergen CJ. Information Retrieval. 2nd edition, Butter-worths, London, 1979.

8. Frakes W and Baeza-Yates R. Information Retrieval: Data Structures and Algorithms. Prentice Hall, Englewood Cliffs, N.J., 1992.

9. J. Foote. An Overview of Audio Information Retrieval, 7: 2-10, Multimedia Systems, 1999.

10. Shuzo Saito and Kazuo Nakata. Fundamentals of Speech Signal Processing. pp.226, Academic Press Japan, Inc., 1985.

11. S. Pfeiffer, S. Fischer, and W. Effelsberg. Automatic Audio Content Analysis. Tech . Rep. TR-96-008, University of Mannheim, D-68131 Mannheim, Germany, April 1996. ftp://pi4.informatik.uni-mannheim.de/-pub/techreports/1996/TR-96-008.ps.gz.

12. M. Brown, J. Foote, G. Jones, K. Sparck-Jones, S. Yong. Open-Vocabulary Speech Indexing for Voice and Video Mail Retrieval. Proceeding of ACM Multimedia 96, Boston, November 1996.

13. L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. pp. 257-286. Proceeding of the IEEE, February 1989.