DEVELOPING A SPEECH- CONTROLLED MUSIC INFORMATION APP

Name: Naveen Sangtani

Email Address: naveensangtani2910@gmail.com

University: Indian Institute of Information Technology & Management, Gwalior

Location (City, Country and/or Time Zone): Gwalior/Jodhpur, India. GMT: +5:30

INTRODUCTION

Speech-to-Text (STT) can be described as a system which converts speech into text. This proposal discusses about the applications of STT system in health care instruments, banking devices, aircraft devices, robotics etc. This proposal discusses the existing system like SOPC based Speech-to-Text architecture, architecture for Hindi Speech Recognition System using HTK and Phonetic Speech Analysis for Speech to Text Conversion. This proposal presents the architecture of the Speech-to-Text system. This proposal provides a tutorial to implement STT system.

The goal of Text-to-Speech (TTS) synthesis is to convert arbitrary input text to intelligible and natural sounding speech so as to transmit information from a machine to a person. Therefore, TTS goes beyond simple ‘‘cut-and-paste’’ systems used, for example, in some telecom applications to read back a phone number. Such systems string together words spoken in isolation and the artifacts of such a scheme are often perceptible. The methodology used in TTS is to exploit acoustic representations of speech for synthesis (see Speech Signal Processing chapter in this handbook), together with linguistic analyses of text to extract correct pronunciations (‘‘content’’, what is being said) and prosody in context (‘‘melody’’ of a sentence; how it is being said). Synthesis systems are commonly evaluated in terms of three characteristics: accuracy of rendering the input text (does the TTS system pronounce, e.g., acronyms, names, URLs, email addresses, a knowledgeable human would?), intelligibility of the resulting voice message (measured as a percentage of a test set that is understood), and perceived naturalness of the resulting speech (does the TTS sound like a recording of a live human?).

Today, applications of TTS are in automated telecom services (e.g., name and address rendering), as a part of a network voice server for

e-mail (E-mail by Phone), in directory assistance, as an aid in providing up-to-the-minute information to a telephone user (e.g., business locator services, banking services, helplines), in computer games, and last but not least, in aids to the handicapped (e.g., cosmologist Steven Hawking). For a much more detailed overview of TTS and its applications.

MOTIVATION

- Working and playing music without even a touch on the screen.
- Developing this type of music player can prove to be beneficial for blinds.
- Taking up technical challenges in integrating voice into an interface from music applications.

LITERATURE REVIEW

Text To Speech

A text to speech (TTS) synthesizer is a computer based system that can read text aloud automatically, regardless of whether the text is introduced by a computer input stream or a scanned input submitted to an Optical character recognition (OCR) engine. A speech synthesizer can be implemented by both hardware and software. It has been made a very fast improvement in this field over the couple of decades and lot of high quality TTS systems are now available for commercial use. Rhythm is an important factor that makes the synthesized speech of a TTS system more natural and understandable; The prosodic structure provides important information for the prosody generation model to produce effects in synthesized speech.

The Pattern Playback was built by Dr. Franklin S. Cooper and his colleagues at Haskin Laboratories. First Electronic based TTS system was designed in 1968. Many computer operating systems have included speech synthesizers since the

early 1980s. From 1990's, there was a progress in Unit Selection and Diaphone Synthesis.

Architecture of TTS

The TTS system comprises of these 5 fundamental components[3]**:**

- Text Extraction
- Text Normalization and Linearization
- Phonetic Analysis
- Prosodic Modeling and Intonation
- Acoustic Processing

The input text is passed through these phases to obtain the speech.

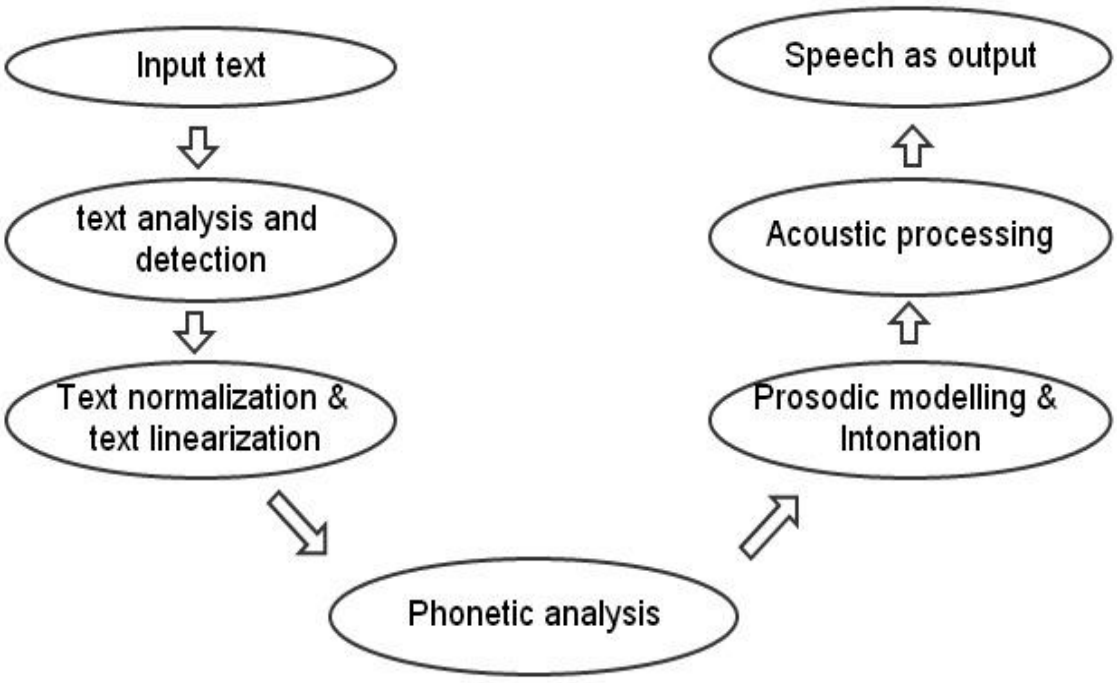


Figure 1 Flowchart for Text to speech

Text Extraction

The Text Analysis part is preprocessing part which analyses the input text and organize into manageable list of words. Text detection is localize the text areas from any kind of printed documents. We aim at developing a technique that work for all kind of documents like newspapers, books etc.

Text Normalization and Linearization

Text Normalization is the transformation of text to pronounceable form. Text normalization is often performed before text is processed in some way, such as generating synthesized speech or automated language translation. The main objective of this process is to identify punctuation marks and pauses between words. Usually the text normalization process is done for converting all letters of lowercase or upper case, to remove punctuations, accent marks , stop words or "too common words "and other diacritics from letters.

Text normalization is useful for example for comparing two sequences of characters which represented differently but mean the same. "Don't" vs. "Do not", "I'm" vs. "I am", "Can't" vs. "cannot" are some of the examples.

The main 4 phases of Text Normalization are:

- **Number converter:**

Number is pronounced differently in different situations.
1772 (date): seventeen seventy two.
1772(phone number): one seven seven two
1772 (quantifier): one thousand seven hundred and seventy two
Fractional and decimal numbers are handled.
0.302 (number): point three knot two

- **Abbreviation converter:**

Abbreviations area changed to full textual format.
Mrs. - Misses
St. Joseph St. - Saint Joseph Street

- **Acronym converter:**

Acronyms are replaced by single letter components.
S. I. - S I

- **Word segmentation:**

Sentences are a group of word segments. Special delimiter to separate segments. (i.e. „||“).Segments can be an acronym, a single word or a numeral.

Examples of acronyms:

“NATO” -“nayto”
“HIV” - “aitch eye ve”
“Henry IV” -“Henry the fourth”
“Chapter IV”- “Chapter four”

Punctuation marks are also identified.

Linearization is the process of giving a hypertext link to give the user a quick overview of the page. Then the TTS system will help to read out the linearized data. This feature helps in selecting the text and reading and also to list the links in the hypertext.

Phonetic Analysis

Phonetic Analysis converts the orthographical symbols into phonological ones using a phonetic alphabet. Basically known as "grapheme-to-phoneme" conversion. Phone is a sound that has definite shape as a sound wave. Phone is the smallest sound unit. A collection of phones that constitute minimal distinctive phonetic units are called Phoneme. Number of phonemes is relatively smaller than the graphemes, only 44.

- **Phoneme Set (English)**

Vowels (19):
/a/, /ae/, /air/, /ar/, /e/, /ee/, /i/, /ie/, /o/, /oe/, /oi/, /oo/, /ow/, /or/, /u/, /ur/, /ue/, /uh/, /w/.
Consonants (25):
/b/, /ks/gz/, /c/k/, /ch/, /d/, /f/, /g/, /h/, /j/, /l/, /m/, /n/, /ng/, /p/, /kw/, /r/, /s/, /sh/, /t/, /th/, /th/, /v/, /y/, /z/, /zh/.

Examples:
/air/: square, bear.
/ow/: down, house.
/ks/gz/: box, exist.

Pronunciation of word based on its spelling has two approaches to do speech synthesis namely

- Dictionary based approach
- Rule based approach.

A dictionary is kept were It stores all kinds of words with their correct pronunciation; it's a matter of looking in to dictionary for each word for spelling out with correct pronunciation. This approach is very quick and accurate and the pronunciation quality will be better but the major drawback is that it needs a large database to store all words and the system will stop if a word is not found in the dictionary.
The letter sounds for a word are blended together to form a pronunciation based on some rule. Here main advantage is that it requires no database and it works on any type of input. In the same way, the complexity grows for irregular inputs.

Prosodic Modeling and Intonation

The concept of prosody is the combination of stress pattern, rhythm and intonation in a speech. The prosodic modeling describes the speaker's emotion. Recent investigations suggest the identification of the vocal features which signal emotional content may help to create a very natural[6] synthesized speech. Modeling of an intonation is an important task that affects intelligibility and naturalness of the speech. To receive high quality text to speech conversion, good model of intonation is needed.

Generally intonations are distinguished as[7]

- Rising Intonation (when the pitch of the voice increases)
- Falling Intonation (when pitch of the voice decreases)
- Dipping Intonation (when the pitch of the voice falls and then rises)
- Peaking Intonation (when the pitch of the voice raises and then falls)

Acoustic Processing

The speech will be spoken according to the voice characteristics of a person, there are three type of Acoustic synthesizing available.

- **Concatenate Synthesis**

The concatenation of prerecorded human voice is called Concatenate synthesis, in this process a database is needed having all the prerecorded words .The natural sounding speech is the main advantage and the main drawback is the using and developing of large database.

- **Formant Synthesis**

Formant-synthesized speech can be constantly intelligible .It does not have any database of speech samples. So the speech is artificial and robotic.

- **Articulatory Synthesis**

Speech organs are called Articulators. In this articulatory synthesis techniques for synthesizing speech based on models of the human vocal tract are to be developed. It produces a complete synthetic output, typically based on mathematical models.

Voice Commands

Users can use voice commands to launch your app and to execute an action. For example, a user using the Contoso Widgets app could tap the Start button and say "Contoso Widgets, show best sellers" to both launch the Contoso Widgets app and to navigate to a “best sellers” page, or some other action that the developer specifies[9]. You can add voice command functionality to your app by completing these three steps:

- Create a Voice Command Definition (VCD) file. This is an XML document that defines all the spoken commands that users can say to initiate actions when launching your app.
- Add code to initialize the VCD file with the phone's speech feature.
- Add code to handle navigation and to execute commands.

This topic contains the following sections.

Creating a VCD file:

The following sections describe how to add a VCD file to your project, and how to create the contents of the file.

- **Adding a VCD file to your project:**

The following steps describe how to add a VCD file to your project.

In Visual Studio, right-click the project name, select Add->New Item, and then select Voice Command Definition.
Type a name for the VCD file, and then select Add.
In Solution Explorer, select the VCD file.
In the Properties window, set Build action to Content, and then set Copy to output directory to Copy if newer.

- **Creating the contents of the VCD file**

Each voice command contains:
An example phrase of how a user should typically invoke the command.
The words or phrases that your app will recognize to initiate the command.
The text that your app will display and speak to the user when the command is recognized.
The screen that your app will navigate to when the command is recognized.

- **Choosing a command prefix**

Setting a value for the Command Prefix element is optional, and is useful to set in two main scenarios:
Your app name is not pronounceable. For example, if your app's name is Contoso Widg3ts, you could set the Command Prefix element to "Contoso Widgets".
You're localizing your voice commands. If you choose to have a command set for each supported language, you can set the Command Prefix element differently according to each language. For example, if your app is named "Contoso Table", you could set the Command Prefix element to "Contoso Mesa" for any Spanish language command set.

Specifying words or phrases in the VCD file

Each Command element must contain at least one Listen For element. Each Listen For element contains the word or words that will initiate the action specified by the Command element.

Initializing the VCD file

You use a single method call to initialize the VCD file during your app's first run. Initializing the VCD file registers the commands to listen for with the speech system on the user's phone.

Extracting URI parameters and executing voice commands

At this point, your app can load a VCD file that defines the voice commands your app will listen for. Now you need to tell your app how to respond to specific voice commands. Depending on the purpose of your app, this could mean that you display a map, read back a reservation confirmation, display flight status, or navigate to another screen.

Localizing voice commands

A VCD file allows you to specify multiple language versions for the commands used to launch your app and execute a command. You can create multiple CommandSets, each with a different xml:lang attribute to allow your app to be used in different markets. For example, an app for the United States might have a CommandSet for English and a CommandSet for Spanish.

Programmatically modifying phrase lists

The API for Voice Commands includes two methods that allow you to dynamically update phrase lists in your VCD file. Updating the PhraseList programmatically is useful for cases in which the voice command is specific to a task involving updated data such as a user's favorites list or updated app data. To update a phrase list in the VCD file, you first get the CommandSet that contains the phrase list you want to update. You then store the retrieved CommandSet in a VoiceCommandSet object. The call specifies the CommandSet by the value of its Name attribute, which must be unique in the VCD file.

Handling phone backups

If a phone backup occurs and your app reinstalls automatically, any voice command data is not preserved. To avoid this scenario and to make sure your app's voice command data stays intact, consider initializing your VCD file each time your app launches, or store a setting that indicates if the VCD is currently installed and check the setting each time your app launches.

GUI screens for voice commands

The voice commands feature includes GUI screens that help users discover which apps on the phone support voice commands, and which voice commands can be used to launch an app.
A user selects an app from the phone's App list.
A user speaks a voice command, from which the app, but not the requested action, is recognized.

Voice command errors

You might encounter voice command errors when working with the feature

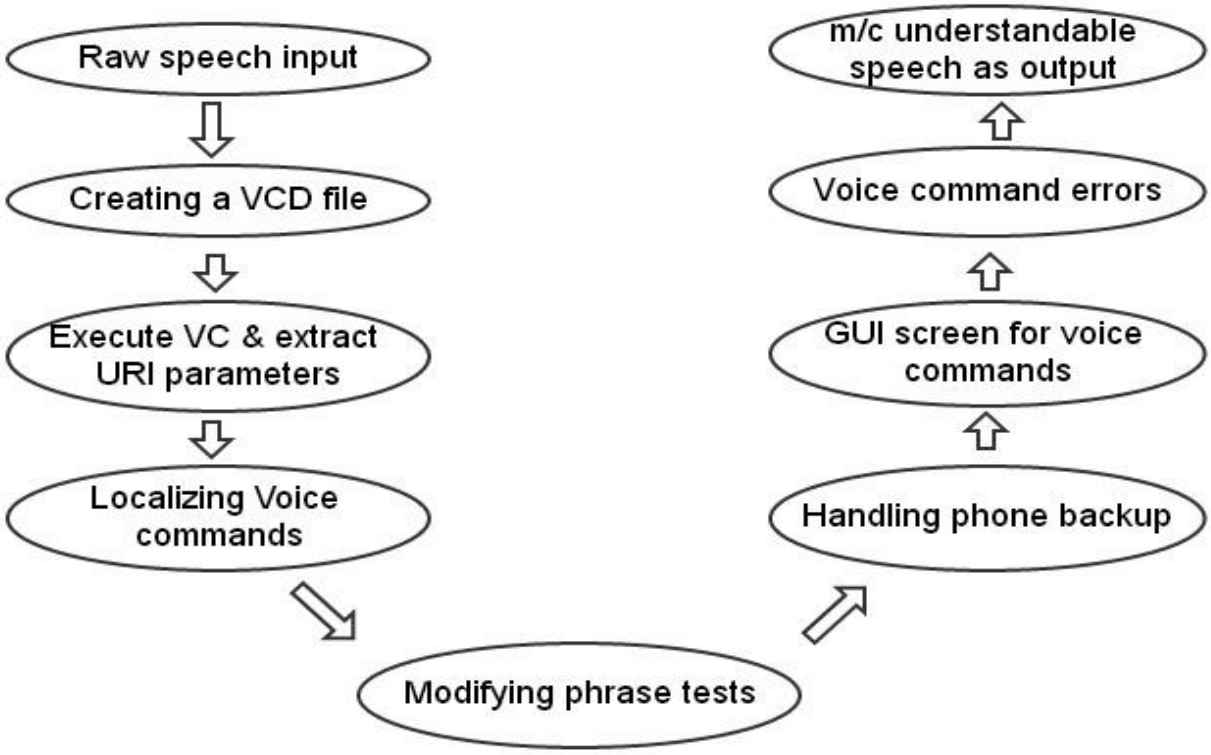


Figure: Flowchart for Voice Command

GAP ANALYSIS

- Only text search is available for Music search in smartphones till date.
- Further, no integration of speech recognition.
- There are only limited (only four) tasks that can be performed using voice recognition: Open an Application, Call a contact, Read an incoming sms, Ability to search in a Search engine.

OBJECTIVE

- To develop a system which is capable of recognizing a limited set of sentences to be spoken using the speech to text converter.
- Download corresponding songs from the API and mix-mash up those songs.

METHODLOGY

- Phase 1 (Learning):
  - A set of sentences which will be of our interest are to be first imparted to the system.
  - Text to Speech converter will be used during this phase.
- Phase 2 (Preprocessing):
  - From raw machine language to machine understandable language.
  - Receives input in the form of raw speech and transforms it into machine understandable language.
- Phase 3 (Interlinking API with voice commands)
  - Preprocessed text will act as an input for the API.
  - If songs are available for the particular string format in database and we have an active web connection, the songs will be downloaded and later can be mixed upon.

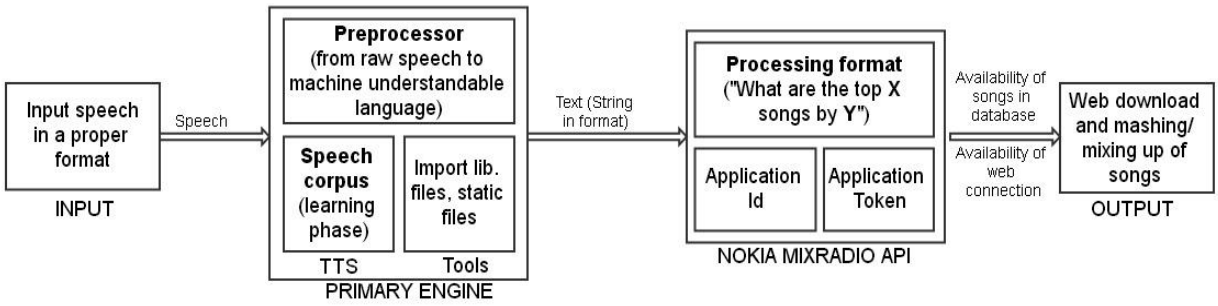
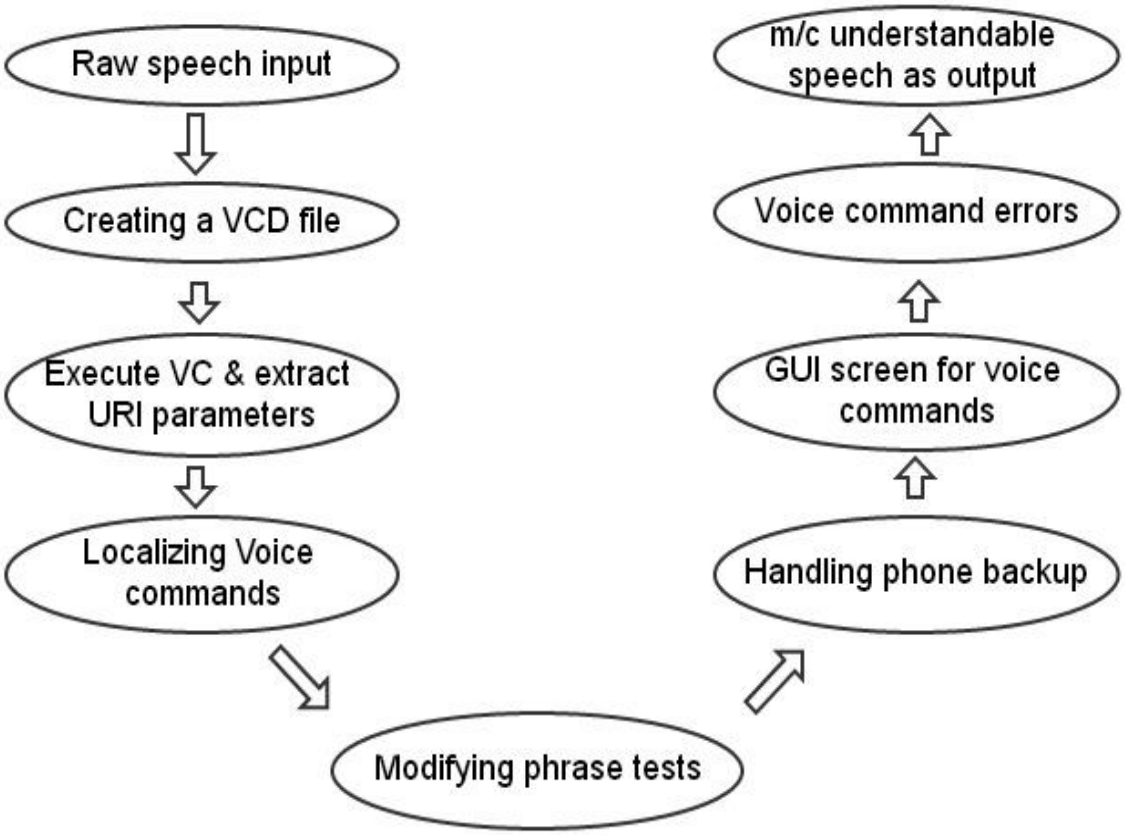


Figure Working Model

PROGRESS

Voice Commands

This algo is converted speech to machine understandable language. Input is given by user is speech . it is convert in machinery language.



Create a Voice VCD file

Create a Voice Command Definition (VCD) file. This is an XML document that defines all the spoken commands that users can say to initiate actions when launching your app. Add code to initialize the VCD file with the phone's speech feature. Add code to handle navigation and to execute commands. In Visual Studio, right-click the project name, select Add->New Item, and then select Voice Command Definition. Type a name for the VCD file, and then select Add. In Solution Explorer, select the VCD file.

Extracting URI parameters and executing voice commands : At this point, your app can load a VCD file that defines the voice commands your app will listen for.

Localizing voice commands : A VCD file allows you to specify multiple language versions for the commands used to launch your app and execute a command.

Programmatically modifying phrase lists: The API for Voice Commands includes two methods that allow you to dynamically update phrase lists in your VCD file. Updating the PhraseList programmatically is useful for cases in which the voice command is specific to a task involving updated data such as a user's favorites list or updated app data.

Handling phone backups : VCD file each time your app launches, or store a setting that indicates if the VCD is currently installed and check the setting each time your app launches.

GUI screens for voice commands : The voice commands feature includes GUI screens that help users discover which apps on the phone support voice commands, and which voice commands can be used to launch an app.

Voice command errors : You might encounter voice command errors when working with the feature. For more info about these errors, see handling errors in speech apps for Windows Phone.

Coding part is given below for speech to understandable language:

```
public async void SpeakText(string TextToSpeak)
{
  SpeechSynthesizer synth = new
SpeechSynthesizer();
  synth.SetVoice(InstalledVoices.Default);
 }
protected async override void
OnNavigatedTo(System.Windows.Navigation.Navigation
EventArgs e)
{
  base.OnNavigatedTo(e);

  if (NavigationContext.QueryString.Any())
  {
    if
(NavigationContext.QueryString["reco"].ToString().
Contains("..."))
    {
      SpeechRecognizerUI recogUI = new
SpeechRecognizerUI();
      recogUI.Settings.ExampleText = "Johnny
Cash";
      recogUI.Settings.ListenText = "Please speak
the name of the artist.";
      SpeechRecognitionUIResult result = await
recogUI.RecognizeWithUIAsync();

RespondWithTopSongs(result.RecognitionResult.Text.
TrimEnd('.'));
    }
    else
    {

RespondWithTopSongs(NavigationContext.QueryString[
"Artist"]);
    }
  }
```

```
  else
  {
    await
VoiceCommandService.InstallCommandSetsFromFileAsyn
c(new Uri("ms-appx:///VCD.xml",
UriKind.RelativeOrAbsolute));
    var installedCommands =
VoiceCommandService.InstalledCommandSets["MusicInf
oCommands"];
    MusicClient nmClient = new MusicClient("App
ID", "App Code");
    nmClient.GetTopArtists(async (artists) =>
    {
      foreach (Artist artist in artists)
      {
        await
installedCommands.UpdatePhraseListAsync("Artist",
new String[1] { artist.Name });
      }
    }, 0, 200);
  }
}
```

speech input is converted in understandable language.

Availability of Text: Preprocessor part is done . this algo is converted speech input . but how is possible for api search for singer name & their songs .So I use again a different second algo for matching with api.means communicated with mix radio api.

Matching method:. singer have a unique id in api. When voice input is given by user & understandable machine language converted with above process. Finally I found string data . api have singer name & their songs. So , I using this process . in this process have some steps

- i/p XML file
- parse XML file by using XML pull parser method.
- Create array list of singer & their unique ids.
- Search singer name (if user gives input)

- Return .

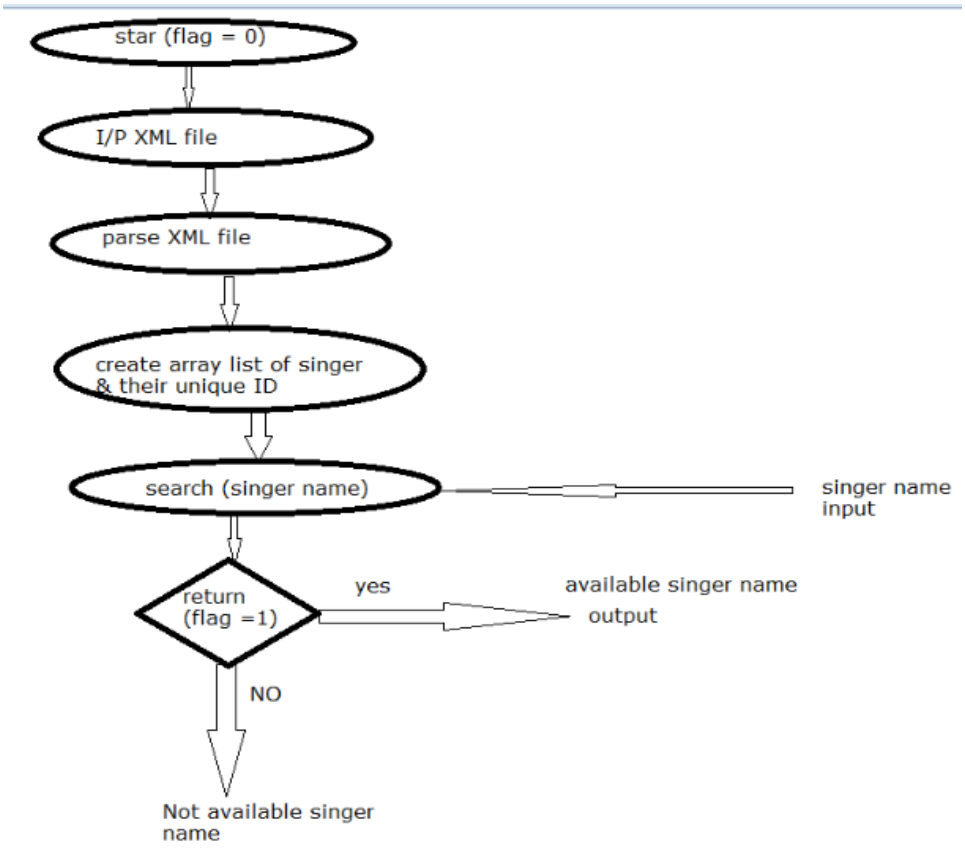


Figure: String matching Algorithm

Parse XML file api have database in XML . so, some data parse in XML database have 2 method.

a) SAX parser method
b) XPP parser method.

So, I used Xml Pull parser method . it is fetch data in api database.

XML pull parsing method:

Pull parsing treats the document as a series of items which are read in sequence using the Iterator design pattern. This allows for writing of recursive-descent

parsers in which the structure of the code performing the parsing mirrors the structure of the XML being parsed, and intermediate parsed results can be used and accessed as local variables within the methods performing the parsing, or passed down (as method parameters) into lower-level methods, or returned (as method return values) to higher-level methods. Examples of pull parsers include StAX in the Java programming language, XML Reader in PHP, Element Tree.iterparse in Python, System .Xml .Xml Reader in the .NET Framework, and the DOM traversal API (Node Iterator and Tree Walker).A pull parser creates an Iterator that sequentially visits the various elements, attributes, and data in an XML document.

Create array list of singers name &ID

By Xml pull parser method ,singer name & their ID's are fetched. Create array list for singers name & id's by this method.

This is xml format data in api database.

```
<?xml version="1.0" encoding="utf-8"?>

<VoiceCommands
xmlns="http://schemas.microsoft.com/voicecommands/1.0"
>
  <CommandSet xml:lang="en-US"
Name="MusicInfoCommands">
    <CommandPrefix>Music Info</CommandPrefix>
    <Example> What are the top songs by Adele?
</Example>

    <Command Name="MusicInfo">
      <Example> What are the top songs by Adele?
</Example>
      <ListenFor> what are the top songs by {Artist}
</ListenFor>
      <ListenFor> what are the top songs by {*}
</ListenFor>
      <Feedback>Thinking...</Feedback>
```

```
      <Navigate />
    </Command>
  <PhraseList Label="Artist">
  </PhraseList>
  </CommandSet>
</VoiceCommands>
```

Search Singer Name

User give input for searching singer name. in this process text input matched with api data . it is taken in array list with id matched.
If decision box means return ,flag = 0,its false condition .singer name is not available in api database.If flag = 1,its true condition .singer name is available in api database ,show it on display.

Code is …

```
import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;
import org.xmlpull.v1.XmlPullParserFactory;
    private String getEventsFromAnXML(Activity activity)
                    throws XmlPullParserException, IOException {
            ArrayList<String> a = new ArrayList<String>();
            ArrayList<String> singers = new ArrayList<String>();
            ArrayList<String> endtime = new ArrayList<String>();
            ArrayList<String> videoname = new ArrayList<String>();
            ArrayList<String> present = new ArrayList<String>();
            String test = null;
            XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
            factory.setValidating(false);
            XmlPullParser xpp = factory.newPullParser();
            xpp.setInput(new StringReader(stringXmltoxml));
            xpp.nextToken();
            int eventType = xpp.getEventType();
```

```
        int attributecount = 0;
        while (eventType != XmlPullParser.END_DOCUMENT) {
                if (eventType == XmlPullParser.START_DOCUMENT) {
                        a.add("--- Start XML ---");
                } else if (eventType == XmlPullParser.START_TAG) {
                        a.add(xpp.getName());

                        test = a.get(a.size() - 1);

                        attributecount = xpp.getAttributeCount();
                        if (attributecount != 0) {
                                for (int i = 0; i < attributecount; i++) {
                                        a.add(xpp.getAttributeName(i));
                                        a.add(xpp.getAttributeValue(i));

                                        if ((test.equalsIgnoreCase ("MixAPI"))) {

linkattribute.add(xpp.getAttributeName(i));

linkattributeval.add(xpp.getAttributeValue(i));
                                        }

                                        if ((test.equalsIgnoreCase("Singer"))) {

themeattribute.add(xpp.getAttributeName(i));

themeattributeval.add(xpp.getAttributeValue(i));
                                        }

                                }
                        }
                } else if (eventType == XmlPullParser.END_TAG) {
                        a.add(xpp.getName());
                } else if (eventType == XmlPullParser.TEXT) {
                        a.add(xpp.getText());
```

```
                if ((test.equalsIgnoreCase("linkurl"))) {
                        linkurl.add(xpp.getText());
                }

                if ((test.equalsIgnoreCase("slidename")))
                {
                        String dg = xpp.getText();
                        int j =dg.lastIndexOf('.');
                        String dg1 = dg.substring(0,j);
                        slidename.add(dg1);
                }

                if ((test.equalsIgnoreCase("starttime"))) {
                        starttime.add(xpp.getText());
                }

                if ((test.equalsIgnoreCase("endtime"))) {
                        endtime.add(xpp.getText());
                }

}

        eventType = xpp.next();
}
a.add("\n--- End XML ---");}
```

About Me:

I am a 20 year old ICT(Information and Communication Technology) student pursing Integrated Post Graduate at Indian Institute of Information Technology & Management Gwalior, India. I am very passionate about programming and coding from a very young age. I am a person who always wants to learn new things. I am thus a quick learner and a person who does not give up early. I like to learn from my mistakes. Even when learning, I research a lot and do not hesitate to take help , ofcourse after web surfing a lot.

I have a keen interest in data structures, algorithms and programming. Besides this, I am a tech person who is always updated with different technologies and the impact it has on society. As a software developer, my intention is to make softwares which not only enrich user experience of the computers but also help the society in general. I am a Linux user from past 3 years.

On the soft skills part, I believe I have leadership qualities with excellent communication, writing and analytical skills

Technical Skills:

Operating Systems:
Linux(Ubuntu, Kubuntu, Fedora), Windows, Android ICS

Programming Languages:
C, C++, Java, Python and familiar to using REST API's

Web designing skills:
HTML, CSS and familiar to JavaScript

Mobile Technology:
Android Development and started with Windows 7 App Development

Tentative Timeline :

Coding PHASE I

May 19 - June 1 (2 weeks)

- Implementing the project as will be discussed with the mentor.

June 1 – June 14 (2 weeks)

- Work on polishing online backend support
- Improvement of existing UI in the existing plugins as will be discussed with the mentor.

June 15 – June 29 (2 weeks)

- Cleaning the new code, starting with documentation, bug-fixing for Phase-I
- Mid-term evaluation

Coding PHASE II

June 30 – July 13 (2 weeks)

- Implementing the TTS.
- Bug-fixing for the above mentioned task.
- Researching for libraries to further improve.

July 14 – August 8 (More than 2 weeks)

- Implementing improved search and categorizing
- Cleaning the new code, documentation, bug-fixing for Phase-II

Pencil down PHASE III

August 9 – August 16 (app. 1 week)

- Wrapping up the project
- Documentation
- Cleaning the code
- Submitting the final code for evaluation and integration in the main repository.