Enterprise search using speech via PocketSphinx

Roopesh Munnaluri
rm1226@wildcats.unh.edu
Master Project in Information Technology
Department of Applied Engineering and Sciences
University of New Hampshire
Manchester, New Hampshire, USA

ABSTRACT

The speech recognition technology is gradually becoming the key technology of the IT man-machine interface. In this paper, I will be developing a voice-based search application which allows me to search all products in a specific site/store. The sites can be configured from the mobile application. This helps the customers to easily access and search the sites for which the app is customized. Currently, there are several cloud-based voice search assistants available in market such as alexa and bixby which aims in providing the similar service, but they do not provide website specific advanced search features. As, it is an open-source mobile app it is free for everyone to download and use, it does not require the users to register for the application unlike Alexa. Saves time to search for products, used for customers who are not familiar with using the websites. This project is going to be developed using Android studio, PocketSphinx and CMU SLM toolkit. CMU SLM toolkit is designed to facilitate language modeling work research community. This is used to process the textual data into word frequency lists and vocabularies. Along with CMU SLM toolkit another tool which I will use is PocketSphinx. PocketSphinx is a part of the CMU Sphinx Open Source Toolkit for Speech Recognition. It is an iterator for continuous recognition or keyword search from a microphone. And Android Studio is another software tool used to integrate the toolkit to Android application. The project is aimed in making the search efficient by reducing the time to search and providing easy access to the customers promoting business.

KEYWORDS

CMU, SLM toolkit, PocketSphinx, Android Studio, Voice based search

ACM Reference Format:
Roopesh Munnaluri. 2020. Enterprise search using speech via PocketSphinx. In *Proceedings of Master's Project Paper*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Now a day's people are using voice to search over the internet or any device. They are using their devices like mobiles, tablets with any voice assistants like Alexa, Siri, Google, or Cortana, and so on. Voice search is a speech recognition technology that uses voice to search instead of typing or looking for the information. People use that because their hands are busy and faster to get the response. There is a huge difference between the way people speak and the way they write. Search engines must have the capacity to understand the context and meaning of queries, beyond simple keywords. Most of the voice assistants do not work if we do not have internet. They use more battery when compared to the other applications. Along with these, they need to have an account related to their servers because they are all relayed on their server before getting a response. Along with these, their search will be based on the internet they do not provide website specific advanced search features.

This Project is an open-source mobile app it is free for everyone to download and use, it does not require the users to register for the application, unlike other voice assistants. It saves time to search for products, used for customers who are not familiar with using the websites. This is going to be developed using Android studio, PocketSphinx, and CMU SLM toolkit.

Master's Project Paper, December 2020, Manchester, NH
2020. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 OBJECTIVES

The Primary objectives are:

- Understand about PocketSphinx and CMU toolkit
- Generating Language model
- Testing Language Model
- Develop an android application with the chosen GUI
- Integrate PocketSphinx with android applications.

2.1 Existing System

There is various famous speech recognition software some of the popular ones are listed below. The speech recognition software is platform-specific or brand-specific taking the example of Siri, which is specific to apple products, Bixby is specific to all of the latest Samsung product. [5]

2.1.1 Siri: Siri an voice recognition platform from apple mainly uses a tool called SiriKit which encompasses the Intents and Intents UI frameworks, which Siri uses to implement app extensions that integrate services with Siri and Maps[1]. Sirikit mainly works with two types of app extensions that is

- An Intents app extension receives user requests from SiriKit and turns them into app-specific actions. For example, the user might ask Siri to send a message, book a ride, or start a workout using your app.
- An Intents UI app extension displays branding or other customized content in the Siri or Maps interface after your Intents app extension fulfils a user request. Creation of this extension is optional.

2.1.2 Bixby: Bixby an voice recognition AI software by samsung which uses internet to provide searches based on the request. The

tool has a feature of learning the behaviour of the user by their response, habits etc.

2.2 Proposed System

In the existing speech recognition software, it has a problem of being platform-specific and generalizes the search in phone, and opens up the particular software which is a default selection within the phone but does not allow you to search within the enterprise application such as Walmart which has both an application and website, so I am proposing to develop an Android application with voice recognition for an option to search in an online website. A language model will be generated based on the large set of data and the dictionary with phonetic symbols. Once the language model is generated, it will be integrated into the android application to obtain the desired search result of the user.

2.3 Learning Outcomes

This project helped me in learning new concepts that I didn't get to learn in my past. A language model is a distribution of probability over word sequences. Provided such a sequence, say of length m, it assigns the entire sequence a probability. To differentiate between words and phrases that sound similar, the language model provides meaning. In American English, for instance, the phrases "recognize speech" and "wreck a nice beach" sound identical, but mean various things. In many natural language processing applications, particularly those that produce text as an output, estimating the relative probability of different phrases is useful. In speech recognition, machine translation, part-of-speech tagging, decoding, Optical Character Recognition, handwriting recognition, data retrieval and other applications, language modelling is used. Sounds are paired with word sequences in speech recognition. When evidence from the language model is combined with a pronunciation model and an acoustic model, ambiguities are easier to overcome.

- Language model
- CMU SLM toolkit
- Android Studio

3 APPROACH AND METHODOLOGY

3.1 Tools

3.1.1 ***Android Studio:*** Android Studio is an IDE used to develop android based applications. It can be downloaded on Windows, mac OS, Linux based operating systems. With the help of Java and XML languages, an application can be built. XML is used to design the GUI part and the back end will be Java.[7]

3.1.2 ***CMU SLM toolkit:*** CMU SLM toolkit [6] is designed to facilitate language modeling work research community. This is used to process the textual data into word frequency lists and vocabularies. Along with the CMU SLM toolkit, another tool that I will use is PocketSphinx. PocketSphinx is also a part of the CMU Sphinx Open Source Toolkit for Speech Recognition. It is an iterator for continuous recognition or keyword search from a microphone.

CMU SLM Toolkit is used to create a language model to recognize the words from voice along with PocketSphinx.

PocketSphinx is a lightweight speech recognition engine, primarily developed for smartphones and handheld devices, but on the desktop, it works as well. It is a recognizer library written in C language. By using PocketSphinx, we can able to set a custom wake-up word and also its works offline.

In the CMU tool kit there some tools which will be helping to generate a language model.

- **text2wfreq** – It takes a text file as an input and provides a .wfreq file which contains a list of words that occurred in the text file along with its occurrences. It uses hash-table to count the word occurrences. The output will be used as input to wfreq2vocab.
- **wfreq2vocab** – It takes a word unigram file(output file from text2wfreq) and provides a .vocab file which is an ascii file which contains a list of vocabulary words. The size is limited to 65535 words.
- **text2idngram** – It takes both the frequency file(output of text2wfreq) and vocabulary file(output of wfreq2vocab) as inputs and provides a .idngram file which is a list of every id n-gram which was in the text along with its occurrences.
- **idngram2lm** - It takes both the idngram file (output of text2idngram) and vocabulary file(output of wfreq2vocab) as input and provides a language model in either binary format or in ARPA format.
- **pocket_sphinx_continous** - It runs voice recognition in continuous mode which takes input from an audio file or from a device and when it detects some voice, it starts recognizing it and produces output in the command-line window.
- **sphinx_lm_convert** - It converts and manipulates language model files. It is also used to change the character encoding of text in a model. It takes one format of language model and converts it to another format.

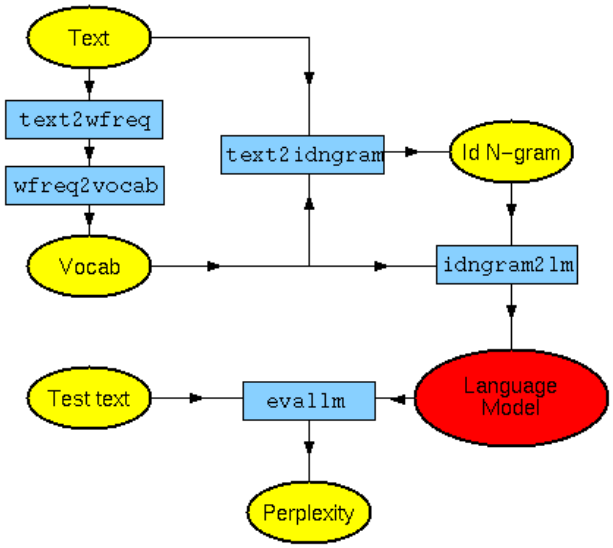

Figure 1: CMU SLM Toolkit-framework [6]

3.2 System Setup

3.2.1 System Requirement.

- Windows version: Microsoft Windows 7/8/10 (64-bit)
- IDE: Android Studio
- RAM: 4 GB RAM minimum, 8 GB RAM recommended.
- Size: 2 GB of available disk space minimum, 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- Resolution: 1280 x 800 minimum screen resolution.

3.2.2 Deployment Requirement.

- Android Version: 8 or higher

3.3 Procedural Steps

3.3.1 Generating frequency file:

```
text2wfreq <sample_test.txt> sample.wfreq
```

3.3.2 Generating vocabulary file:

```
wfreq2vocab <sample.wfreq> sample.vocab
```

3.3.3 Generating n-gram file:

```
text2idngram -vocab sample.vocab -temp .
-n 3 <sample_test.txt> sample.idngram
```

3.3.4 Generating language model:

```
idngram2lm -idngram sample.idngram -vocab
sample.vocab -arpa sample.arpa
```

3.3.5 Generating binary NGram language model file(DMP file):

```
sphinx_lm_convert -i sample.arpa -o sample.dmp
```

3.3.6 Executing Pocketsphinx in UNIX environment.

```
pocketsphinx_continuous -lm sample.arpa -infile
test.wav -dict sample.dic -hmm ~/en-us/en-us/
```

3.3.7 Integrating with Android Studio: After installing Android Studio, a JAR/AAR package needs to be downloaded from the official GitHub page of CMUSphinx. After downloading, it has to import to android studio. After importing manually, we need to include those in Gradle file as well. Along with including, permission has to set in AndroidManifest.xml file to recognize the audio provided by the user. To run the PocketSphinx, an acoustic model need to set and a dictionary file which contains the phonetic symbols of the words. These are some of the inputs we need to provide before attaching our language model to Android Studio. Once the application can recognize the words from the voice, we need to generate the apk file. With that apk file, we can able to install that application in any of the android phones.

4 DATA SOURCE

Data source for the project is taken from file containing a dictionary which has various words containing phonetics and 300+ hours of English transcripts as the learning training model through which we can train the language model to recognize various accents from people from various countries. for example

5 DISCUSSION AND EVALUATION

5.1 Choosing the right IDE:

There are few notable IDE's such as Eclipse, Android Studio, Visual Studio etc.

5.1.1 Android Studio: Android Studio is Google's Android operating system's official integrated development environment (IDE), based on JetBrains' IntelliJ Concept software and specifically designed for Android development. It is available for download on operating systems based on Windows, macOS, and Linux. Android Studio supports all of the same IntelliJ (and CLion) programming languages, e.g. Extensions such as Go, Java, C++, and more and Kotlin supports Android Studio 3.0 or later.[2]

5.1.2 Eclipse: Eclipse is an IDE (Integrated Development Environment) that is used in computer programming. It provides a base workspace and an extensible environment customization plug-in system. Eclipse is mainly written in Java and its main use is for the development of Java applications, but it can also be used in other programming languages to create applications. It can also be used to produce LaTeX papers (via a TeXlipse plug-in). IBM VisualAge emerged as the initial codebase. For Java developers, the Eclipse software development kit (SDK), which includes the tools for Java development, is intended. Through downloading plug-ins written for the Eclipse Framework, such as development toolkits for other programming languages, users can expand their abilities and can write and contribute their own plug-in modules. After Google discontinued the development of its plug-in for the Eclipse IDE, which is designed to provide an integrated framework in which to create Android apps, Android Development Tools (ADT) was replaced in 2015 by the Eclipse foundation's own plugin, called Andmore: Development Tools for Android.[3]

Andorid Studio completely supports Android where as Eclipse supports Android through ADT extension. When it comes to user interface Android Studio is much easier and quicker but Eclipse is not a native IDE for Android and is thus more complex. Eclipse is much larger than IDE as compared to Android and requires high RAM space and substantially greater CPU speed.

5.2 Language Model:

A subfield of linguistics, computer science and artificial intelligence, Natural Language Processing (NLP) is concerned with the interactions between computers and human language, in particular with how computers are programmed to process and interpret large quantities of natural language data. The outcome is a system that can 'understand the content of documents, including the qualitative complexities of the language within them. The program will then reliably extract data and observations found in the documents and categorize and organize the documents themselves. Natural language processing problems often include speech recognition, comprehension of natural language, and generation of natural language.

For this particular project, we discussed about various language models. In that the most optimal language model was to generating our own training set. and this training set has few common words which are used by people to search online as the proposed model is for enterprise search.[4]

5.3 Difficulties Faced:

- Understanding how the language model is generated is one of the difficulties which I faced.

- In User Interface, generating the frequency bars based on the voice input by user is tough thing. I could not able to complete it. Instead of that I added GIF image for that.
- Finding the correct data set to train the language model.
- Including GIF image in UI. For displaying Images or videos there are predefined widgets and objects. But there are no predefined objects for GIF image.

6 RESULTS

6.1 Generation of Language Model and its intermediate files

By following the steps mentioned in 3.3 there are some intermediate files that will be generated before the language model. In the first step, frequency file is generated which contains the information about the number of times each word is repeated. Next, we convert this frequency file to the ascii file with .vocab extension. The frequency file and vocabulary file are sent as inputs for generating n-gram file. Then, the language model is generated by passing the n-gram file generated in the previous step and the vocabulary file.

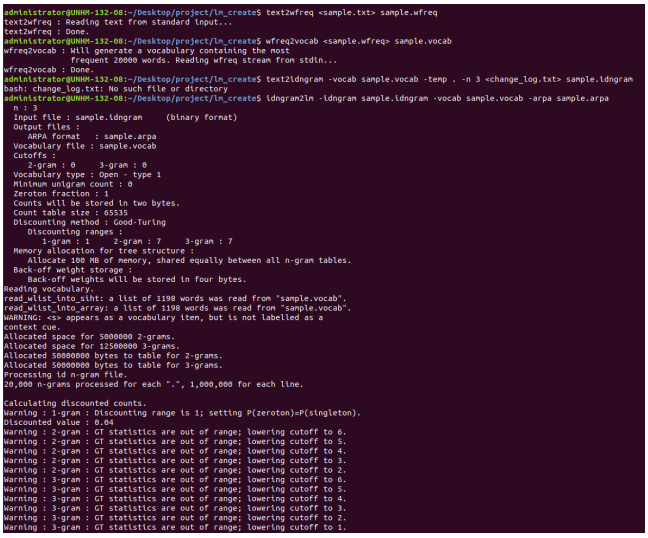

Figure 2: Language Model

The language model, dictionary and acoustic models should be sent to pocketsphinx, a library which converts speech to text. The output will be a text which is displayed on the terminal. Below figure is the screenshot of the output.

Figure 3: Testing PocketSphinx with Language Model

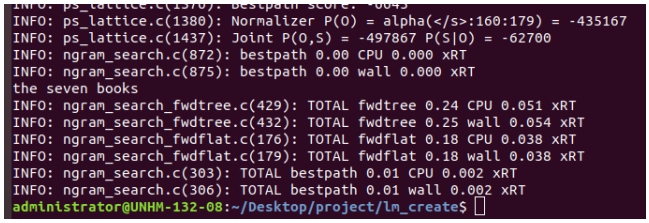

Figure 4: Testing PocketSphinx with Language Model

The generated .arpa file of the language model is to be then converted to .dmp file. This .dmp file generated is a binary n-gram file which is used as a language model in android studio.

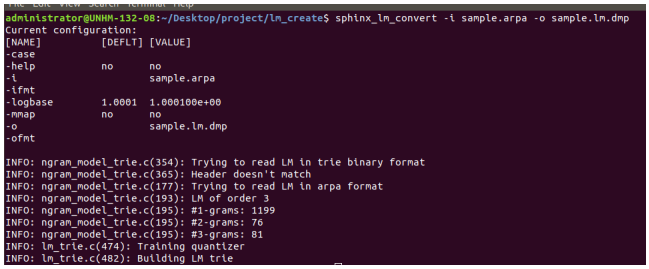

Figure 5: Converting .arpa to .dmp

6.2 Working of android application

The home page of the application consists of gif view, text view, enterprise search option and buttons for search and listen. First the user needs to choose the enterprise option and then click on listen which enables the gif view and the recognizing phase of the speech starts. After speech completion, user needs to click stop, which will now convert the speech to text and is displayed in the recognized text field. With the text field and enterprise option selected, the user can now click on search which will redirect to the enterprise application or the browser.

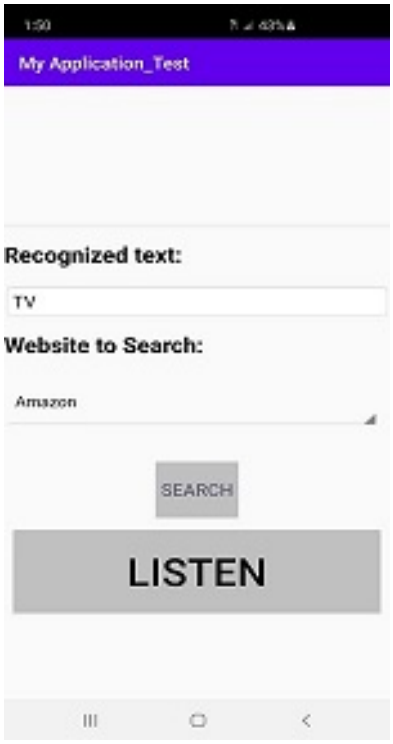

Figure 6: Home Page

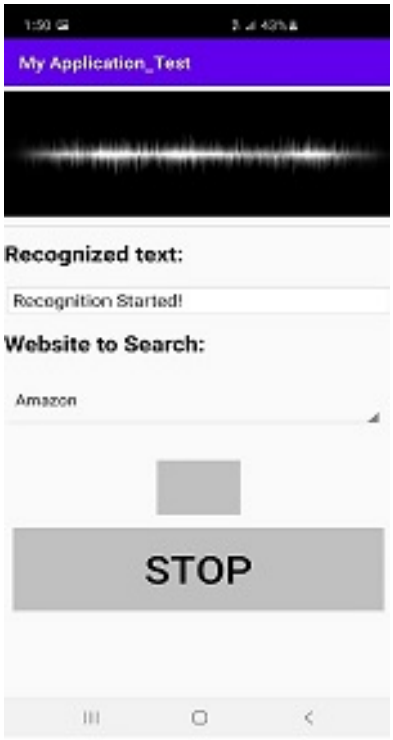

Figure 7: Recognizing

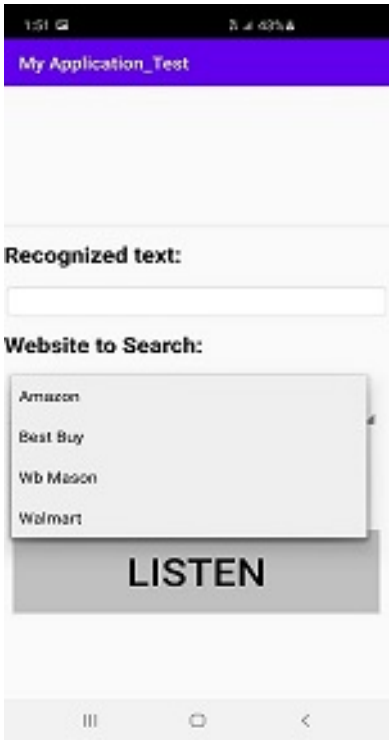

Figure 8: Choosing

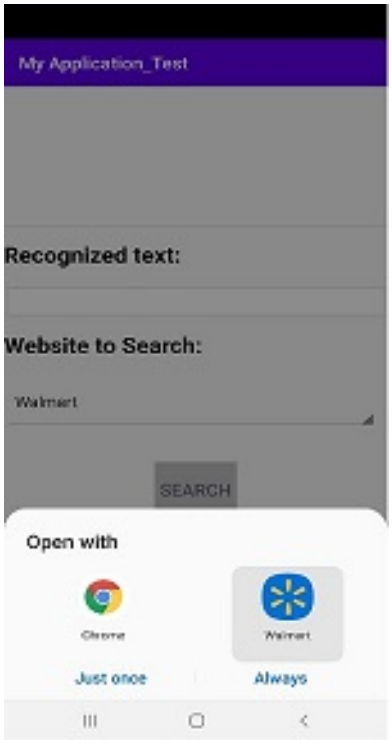

Figure 9: Redirecting

7 CONCLUSION AND FUTURE WORK

There are many notable voice recognition tools such as alexa, siri, bixby from well established companies such as apple, samsung, amazon. The main objective for the project was to understand about PocketSphinx and CMU Sml toolkit and develop an android application where i was successfully able to use a language model. This language model was created with n-gram file generated from the frequency of words and its ascii file from the sample dataset. This android application is for mainly android phones so its platform specific so in future we can make this application even for apple phones as well. we could also implement or train the model with big training set to increase the efficiency of the application.

REFERENCES

[1] [n.d.]. Apple Developer Documentation. https://developer.apple.com/documentation/sirikit
[2] 2020. Android Studio. https://en.wikipedia.org/w/index.php?title=Android_Studio&oldid=991417078 Page Version ID: 991417078.
[3] 2020. Eclipse (software). https://en.wikipedia.org/w/index.php?title=Eclipse_(software)&oldid=991422879 Page Version ID: 991422879.
[4] 2020. Natural language processing. https://en.wikipedia.org/w/index.php?title=Natural_language_processing&oldid=992892251 Page Version ID: 992892251.
[5] Pavel Averin. 2016. Voice Recognition Tools Review. Alexa, PocketSphinx, Google API or Project Oxford? https://www.netguru.com/blog/voice-recognition-tools-review
[6] Philip Clarkson. 1999. The CMU-Cambridge Statistical Language Modeling Toolkit v2. http://www.speech.cs.cmu.edu/SLM/toolkit_documentation.html
[7] Google developers. 1999. Android Developers. https://developer.android.com/