Speech-Based Music Information Retrieval System

by

Jagjit Singh
Bachelor of Technology, Guru Gobind Singh Indraprastha University, 2012

A Project Submitted in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Speech-Based Music Information Retrieval System

by

Jagjit Singh
Bachelor of Technology, Guru Gobind Singh Indraprastha University, 2012

ABSTRACT

Music is considered as the art form that is completely developed and closest in interaction with the human race. Recent years have seen rapid growth in the use of mobile devices making it more accessible to humans. The close alliance of humans with music has resulted in the paramount requirement of developing and implementing intelligent conversational agents to aid music exploration. This project talks about the design strategy and implementation of an end-to-end intelligent conversational agent based on a natural language user interface for listening and learning about music. The system includes a virtual assistant based on Amazon's Alexa that combines with natural language processing, knowledge representation, and music recommendation engine. I further specify and list the limitations of the existing system and provide guidelines for future development of such music user interfaces.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

Dr. George Tzanetakis for believing in me and giving me a chance to work with him. Can't thank you enough for the project idea and continuous support all through my studies in the last two years.

University of Victoria Library for making available all the literature related to the topic.

Wendy Beggs for supporting me in all the administrative tasks related to my studies in the last two years.

DEDICATION

I dedicate this thesis to my late aunt, Miss Ajit Kaur. This journey was never possible without her love and support. She would be really happy and proud to see me what I am today.

For my parents, thank you for believing in me and working hard to fulfill all my dreams. Your love and prayers of day and night make me able to reach this stage of my life. My brother, Jujhar has always motivated and encouraged me to take bold decisions in life.

To Karamjeet, my girlfriend, whose sacrificial love and care made it possible for me to stand against all odds and achieve my goal.

Last but not least my friends Harmeet, Gurjeet, Ishmeet Kohli, Tarandeep, Amrit, Gursewak, Ravinder and Ishmeet Malhotra this was never possible without your support and love from day one. Long live our friendship.

Chapter 1

Introduction

1.1 Artificial Intelligence and Intelligent Agents

The enormous size of the data and information involved in the music world and its exploration has always been an area of interest for the computer scientist. The term *music exploration* is heavily loaded, but it basically means playing and accessing music-related information proficiently. The ideal interface for music exploration should be interactive and adaptable. One option for such interfaces is *intelligent agents*. Russell and Norvig defined an **agent** as an individual entity that has sensors and actuators to perceive and act upon its environment respectively [46][42]. When these agents have the ability to learn and use the provided knowledge to achieve a goal, they are then termed as **intelligent agents (IA)** [29].

1.2 Natural language User Interfaces

Intelligent agents can be implemented in multiple ways and formats, one such implementation is *conversational agents*. These agents are intended to communicate with humans using a logical structure. The conversations can take the form of speech, gestures and any other form of communication [7]. A conversational agent that use text conversations as the medium of user interactions are termed as *chatbots* [25]. A more advanced and adaptive way of communication is using *natural language user interfaces*. In **natural language user interfaces (NLUI)** the system controls are transformed into linguistic indications like sentences, verbs and phrases [41]. We will be implementing the NLUI using a speech-based *human-computer interaction* model.

1.3 Natural language processing and Knowledge Representation

Natural language user interfaces have been adopted as the means for human-computer interaction for all recent developments of virtual assistants like Alexa, Google Home and Siri [4]. These virtual assistants have common components that aid the process of listening, processing and replying to user requests. *Natural language processing*, *knowledge representation* and *processing unit* are building blocks for virtual assistants. **Natural language processing (NLP)** is the area of research and application where computer understands and manipulate instructions in natural languages, like text and speech [12]. Processing and using information effectively requires a robust and meaningful representation. **Knowledge representation** is the field of AI dealing with the representation of real-world information in a way that the computer system can use it to solve complex problems [5]. The information is stored using logical sets and subsets of data like a semantic structure.

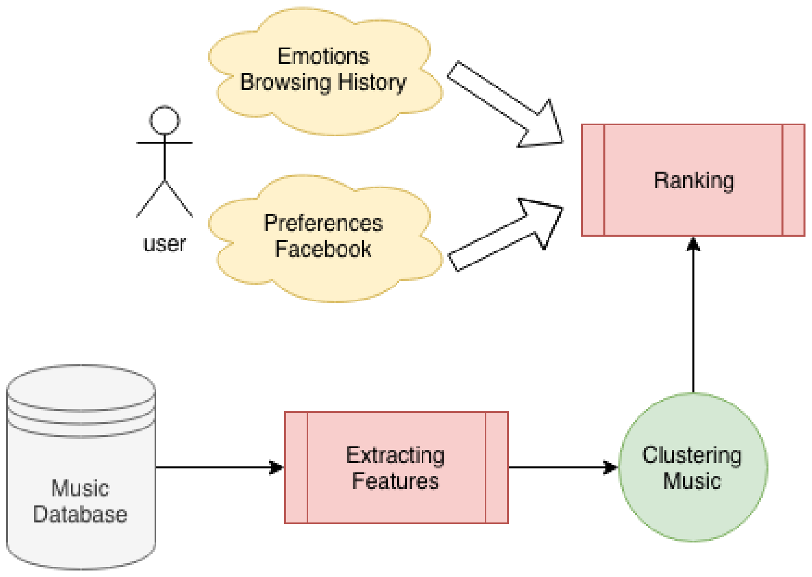


Figure 1.1: Basic recommendation system using user emotions and preferences to recommend music to the user.

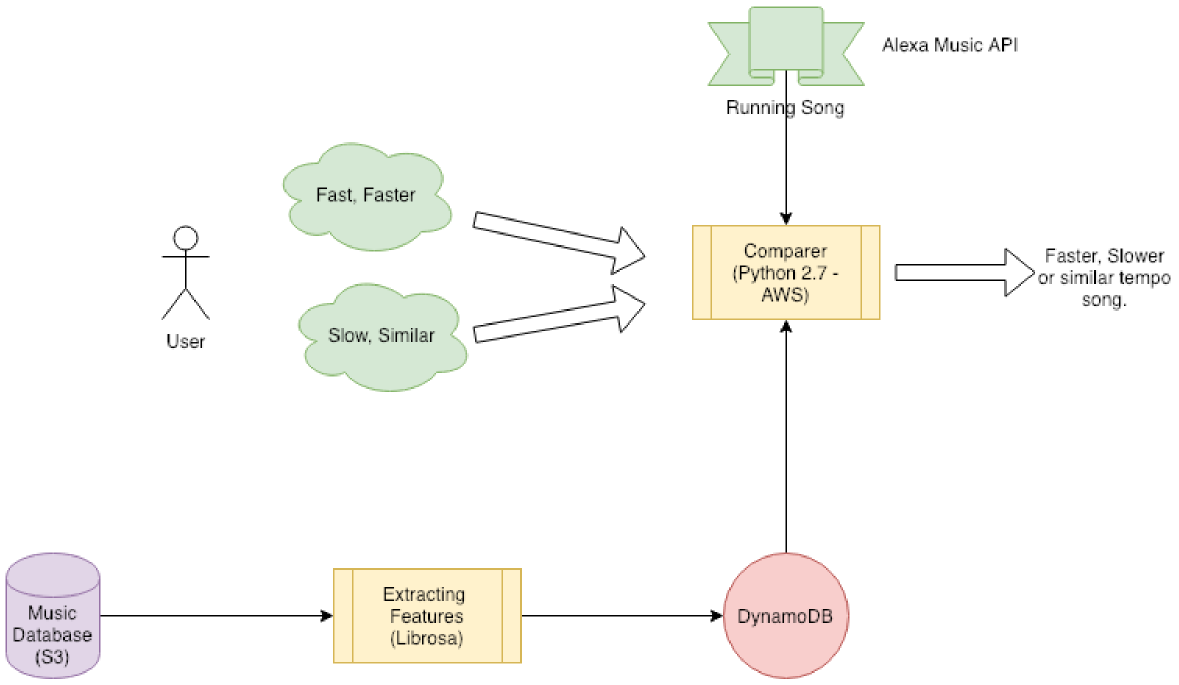

Alexa Music API
Running Song
Fast, Faster
Slow, Similar
User
Comparer (Python 2.7 - AWS)
Faster, Slower or similar tempo song.
Music Database (S3)
Extracting Features (Librosa)
DynamoDB

Figure 1.2: Music Information Retrieval

1.4 Amazon Web Services and Music Information Retrieval

Processing data on a Local machine poses a lot of challenges like distribution, compatibility, and expandability. *Cloud processing* is becoming popular day by day because of its flexibility and ease of use. **AWS Lambda** is a serverless computing platform provided by Amazon [45]. Our system will use Lambda for processing user requests from talking agent and *music recommendations*. Music recommendation is only possible if we can find features in the given set of music and relate them (Figure 1.1). **Music information retrieval (MIR)** deals with processes and techniques that can be used to retrieve information from the music signals like tempo, texture, structure, and duration [16]. Figure 1.2 shows how the MIR component processes user request by using information from music database and user parameters. **Music recommendation** deals with filtering of available options on the basis of information provided by the user. The recommendation can work on various factors like social interactions, mood, emotions and time of the day [24][17].

1.5 Outline of the Project

Chapter 2 describes the previous research and implementation work that has been done related to the project topic and various sub-parts of it.

Chapter 3 highlights in detail, the design, prototype architecture and how various components work together.

Chapter 4 includes the implementation details outlining the various technologies, algorithms, and approaches used to synthesize this product.

Chapter 5 describes limitations and future development.

Chapter 6 presents the evaluation process and results.

Chapter 2

Related Work

In this chapter, we will discuss the related work done on the topic and how it helps to inform the system design and formulate the system requirements. Conversational agents and chatbots both work on the same underlying technology of artificial intelligence with one having a text-based communication interface and other having a voice-based. Intelligent agents should follow a conversational strategy for effective communication and recommendation. In 2018 Ikemoto and Yuichiro proposed a conversational strategy to define the information shared with the user, questions asked to the user and processing the user responses [28]. They also developed a low-level prototype with a graphical user interface to do a user study to prove their strategy actually improves the effectiveness of a conversational agent. Another important agent that worked as a chatbot for recommending television movies to the user on the basis of linked open data and aimed at implementing the Internet of things was developed by Fedelucio and Marco in 2017 [38]. In 2017 Atzori, Maurizio and Boratto developed the concept of combining a conversational interface with recommender system [3]. They developed a chat-based application that was used to submit and request information for planning travels. They achieved semantic data integration, intelligent recommendation, and effective interaction. Conversational agents have also been used previously to facilitate the interaction between citizens and policy-makers. In 2017 Pavel and Achilleas did a user evaluation of one such agent to help address the challenges associated with the future [31]. Intelligent agents have also been integrated into the running systems where users interact like the Facebook messenger. These types of integrations really help users to engage with the system without a learning curve and they find it more effective and user-friendly [13]. In 2016 Holotescu exploited this feature and developed a chatbot for the Facebook messenger to recommend people their choice of courses based on their interests [27]. The chatbot would search online courses from websites such as Coursera and Lynda in order to fetch courses based on user interests. With emerging conversational agents and voice assistants, they have to be closely integrated with human life and thus can pose threat and security issues to us. Recently Cho, Kim and Choi identified potential threats in voice assistant applications and assessed the risk of those threats using the STRIDE and DREAD [11]. STRIDE is a security model developed by Microsoft, it is used to identifying computer security threats [21]. DREAD is used for risk-assessment using categories like damage, affected users and discoverability. The total number of formulated security threats came out to be 16 and the strategies used to counter these threats were also investigated and suggested in the paper. Conversational agents are getting more advanced every day with new machine learning and technologies but are they really useful form the user perspective? In 2016 Ewa and Abigail conducted interview sessions with 14 individuals that heavily interact with these conversational agents on a daily basis in order to understand the interaction factors affecting daily use [34]. The user expectations were out of step with the operation of the systems in terms of capability and goals because the software designers did not follow the practice of software prototyping. Using Norma's gulfs of execution and evaluation they further revisited the system to identify the areas of improvement and issued guidelines for future agent development [39]. An initial prototype can capture user interactions and questions asked by them. It, therefore, becomes important to have an initial prototype for such systems.

System Requirement 1: Design and develop a minimal end to end prototype that replies to user requests related to playing and fetching information about music to gather user interactions and questions asked by them.

Conversational agents have been deployed in day to day use like fitness training, cooking, shopping, and warehouse businesses [14][10][32][23]. In time with other technologies the health care system has also seen a rapid rise in the implementation of IA's to support various tasks [37][48][44]. It started in 1998 when Mettler, Liselotte and Ibrahim developed voice controlled optic holder [36]. The study suggested that voice control optic arm was better in working that a normal human controlled laparoscopic arm. A similar study for the user satisfaction and accuracy was done for the voice-controlled endoscope by Punt [40]. Classroom sessions have been replaced by e-learning these days where students feel more comfortable to interact with virtual assistants that get in the conversation and impart knowledge. This change in medium of education led to a number of projects that implemented Teaching Assistant Robot in various forms of education like music, classroom and dance [43][22][18].

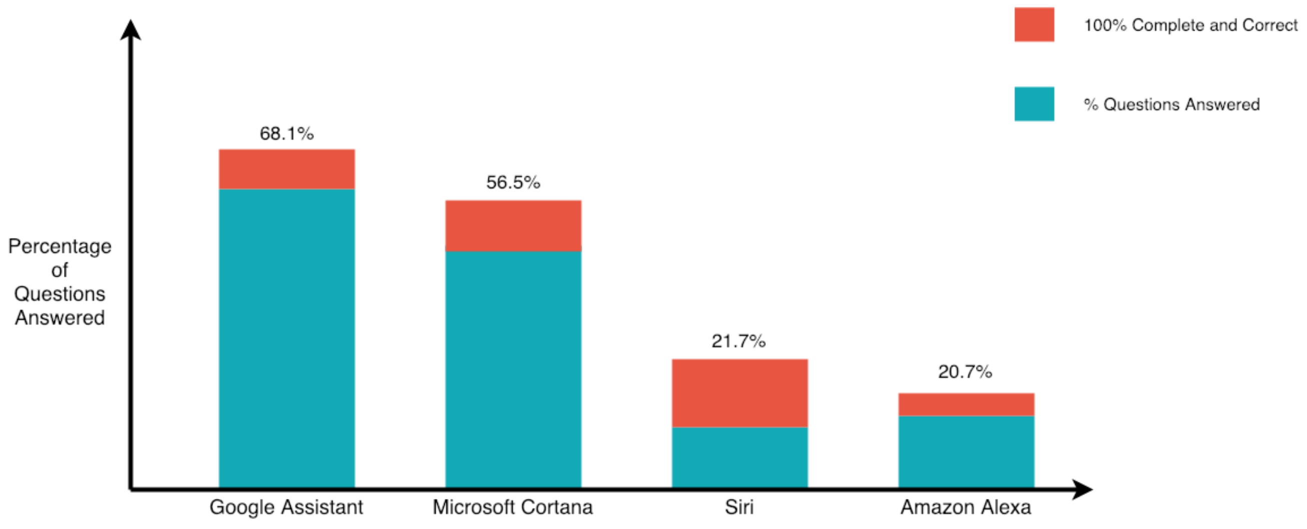


Figure 2.1: Performance of conversational agents in a quiz of 5,000 questions. [Data source - https://www.stonetemple.com/digital-personal-assistants-study]

2.1 Conversational Agents and Chatbots

The last four years have seen an era of conversational agents becoming a common part of our daily routine. All major IT giants like Apple, Google, Microsoft, and Amazon have stepped in and developed their own conversational agents to be supplied with their software packages. Figure 2.1 shows the distribution in market share for these agents with the number of correct answers to user questions. The beginning of these agents started with just voice controls where the system would understand certain voice inputs and do a specific task. In 1999, De Vet, John, and Buil Vincent developed the first personal assistant for playing, selecting and browsing music on an audio video system attached to the computer [15]. The assistant was capable of doing basic music controls but was missing complex operations related to MIR. Considering the missing functionality, I propose the following problem statement.

System Requirement 2: The system should use MIR techniques to answer questions related to the song features like - play me something faster.

The project was developed as a prototype to understand of application of voice commands to a large data in operation. The input voice controls were redundantly designed to aid users with alternative instructions in order to use the system effectively. They defined personalization and multi-user as two major limitations in their prototype and suggested that adding these to future agents would definitely impact human-computer interaction. In 2017, Gustavo and Quesada ran user studies on various speech-based natural language user interfaces and found out that even after using the best possible processing techniques, these systems still need a lot of work to get close to real natural-language user interfaces [33].

2.2 Recommender Systems

Information overload is the next big problem that is being faced by users of the Internet these days. Systems that could process a large chunk of information by prioritizing, filtering and then delivering the relevant information are becoming essential for users. These systems are called recommender system. They process the generated data and present user with personalized content and services. Recommender systems can be content-based, collaborative and hybrid in nature. Celma in 2010 defined the basic music recommendation technologies and explained these terms [8]. Recommendation system based on user interests and music data grouping were built to increase the accuracy of the recommendation process [9][26]. In 2002 Burke and Robin did a survey on various recommender systems and found that the accuracy and efficiency of the hybrid system are better than conventional ones [6]. Olivier and Thomas presented a study to prove how precision can be improved by implementing machine learning techniques in these systems [19]. With great efficiency and usages comes a lot of limitations as well. In 2005 Adomavicius and Gediminas studied various working recommenders and documented limitations and various ways to improve them [2].

2.3 Conversational Agents for Music

Implementation of conversational agents for music and its exploration has a lot of commercial value. Spotify, Google Music, Apple Music and many other music streaming services have provided their users with a voice-based search and controls for audio players. In 1996, Smith and Henderson proposed a voice-controlled music exploration system that uses MIR techniques to answer user queries [35]. Abdul and Woods did a survey on various chatbots over the last decade and discussed the similarities and differences in the implementation techniques used for chatbots [1]. They also concluded that speech-based systems were more user-friendly than text-based systems. It, therefore, becomes important to have a speech-based interface for music exploration.

System Requirement 3: The system should be voice-controlled.

Speech-based communication became the standard interaction medium for various human-computer interaction systems. Mark, Michael and Dermot designed a product for adding voice activation and voice control to a media player [20]. In 2008, Kyu-hong, Jeong-Su and Ick-sang provided a method for searching music based on speech recognition [30]. They calculated a search score for each query using speech inputs. Music can be accessed and played from various locations. A simple user may have a single source whereas a complex user may use multiple sources. The system should not restrict the user to only explore music from his personal library but also external music streaming applications like Spotify. To add this functionality, we devise the below problem statement.

System Requirement 4: The system should be capable of switching between different sources for music information and playback.

Chapter 3

Architecture

3.1 Introduction

The system under development should have an agent that would talk and send controls or access information from other components via web services and JSON objects. We will be implementing the approach outlined in the Huaigu and Natchetoi work [47]. They used a central control unit to control the asynchronous communication within the components of the system. In this section, we will specify the design specification and components of our prototype starting with high-level architecture followed by the individual component description. The high-level description of the system will specify the component structure, their interaction and flow of information within the system. Individual components detail like version, implementation, and functionality will be explained in the following sections.

3.2 Design

To start with deciding on structure components we needed a conversational agent that can talk with the user and give us back the user responses. Figure 3.1 shows various components of the system. The conversational agent can be text-based or voice-based depending on the medium of communication. With the increase in availability and increased efficiency of voice agents, we decided to use a voice communication system. Systems like Siri, Google now and Alexa can be used to communicate with the user. Alexa has a lot of available resources online and flexibility to be used for our case. We decided to use Alexa to provide us with user speech to text conversion. The attained

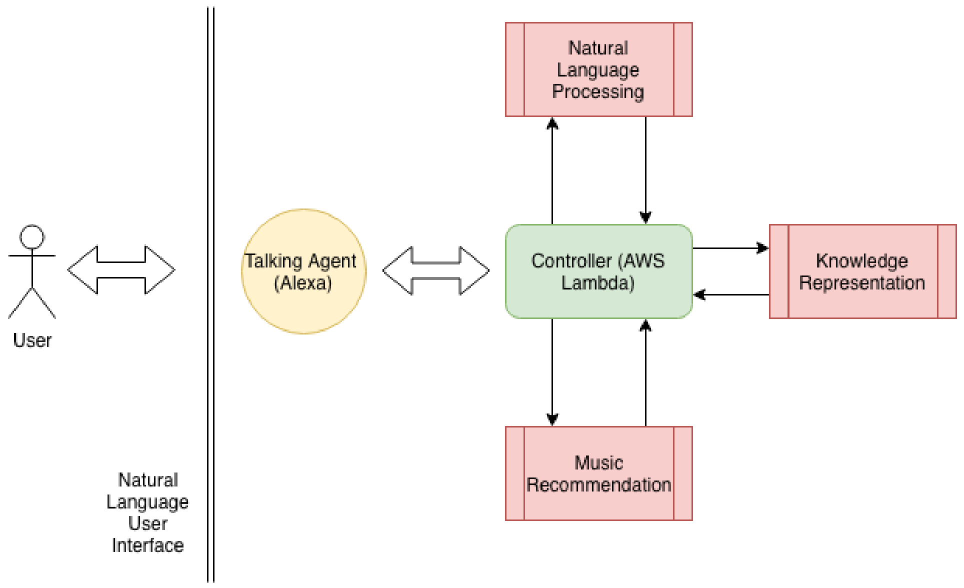


Figure 3.1: Components of a conversational agent.

text from the user was supposed to be processed by the natural language processing unit. The central coordinating unit that takes into account various communication and controls the flow of data within the system is called the controller. The controller keeps a track of the system state and depending on the user responses sends the appropriate action to the components. After receiving the converted text from Alexa is it sent to the natural language processing unit for processing, which in turn replies with the action and parameters. The action and parameters are passed to the controller and then on the basis of content specific actions are performed.

1. List similar artists, list top songs by an artist.
2. Request/recommend a similar, faster or slower song.
3. Get information regarding an artist, song or album.

3.3 Components

1. Natural language processing unit - Alexa skill has a built-in speech to text converter that gives our system a natural language user interface. The converted text from the skill is sent to the Python controller which further passes the text to the natural language processing unit (NLP). The NLP unit uses a basic grammar and a small set of defined intents to extract important parameters and

actions from the supplied text. We can make the system answer more questions if we increase the number of intents, but this is practically an infinite number. The text after processing contains action and parameters that are used by the controller to either perform actions or send the information to other units for further processing.

2. Recommender system - The recommender system works on a music database and extracts music features from all the audio files using a third-party Python library called LibROSA. After extraction, these features are stored in a database that maintains music information related to songs like the year, genre and artist. LibROSA does the MIR for us and returns features like beats, texture, and timings for first 5 beats in the song. We have used a MySQL based relational database for this storage function. The analysis in our case is performed on the first 20 seconds of the song, in order to make the response of the system faster. We can also increase the accuracy of the recommendation process if we extract the features for the entire song and then recommend accordingly. If the user requests for a recommendation after listening to a song from the library, the recommender system goes through the database and returns the song that matches the most with the available features.

3. Knowledge representation - Processing and working on dialogs in natural language require information stored in a special format. The conventional data storage systems are inefficient and complex to pair with NLP. Knowledge in these systems is stored with keeping the design formalization in mind so that it makes complex systems easier to design and build. We will be using semantic nets to design the knowledge representation formalism. Semantic nets use just pure logic to represent relations between the complex entities of the system. Semantic nets can be seen as a directed or undirected graph with vertices and edges representing concepts and relations in the system.

4. Controller - Complex systems have a number of state transformations (control transfer), they move from one execution to another (data transfer) and thus it's important to maintain the parameters and state variables. The heart of the system is called a controller, this component is equipped with a state manager that keeps in place the present state of system variables. Also, the communication and information passing within the system have to be controlled by a

central unit. This task is performed by controller which is written in Python and executes actions with API calls. This controlled since has to interact with the web services like AWS Lambda, therefore, is written as a web service. We have used Flask, a Python based framework for writing web services.

5. Speech to text converter (Alexa skill)- The user is supposed to interact with the system by voice inputs only, but the NLP unit expects the responses in the text format. Alexa skill kit has a built-in speech to text conversion system that has been adapted to use with our set of requirements and other system components. We have deployed Alexa skill kit to make a basic music skill that hears user input and returns the processed text to the controller which then sends it to the NLP unit. The Alexa skill is deployed on the AWS servers. This unit is also responsible for playing and responding back to the system. Alexa skill kit, when supplied with a text phrase, can convert it into a voice output delivered to the user through its speakers. Adding to this, it can also play music when given a location of the audio file.

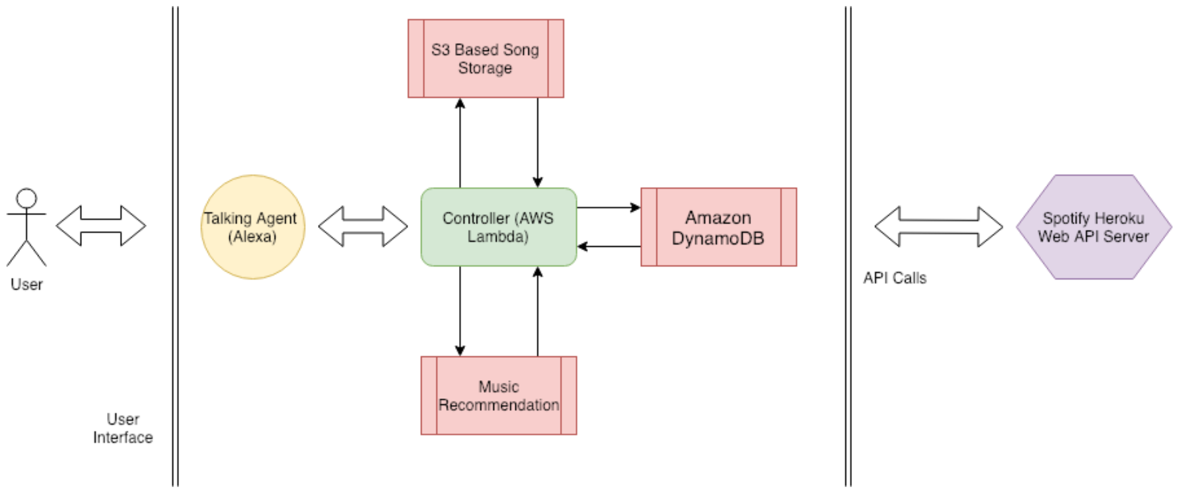


Figure 3.2: Components of implemented prototype.

3.4 Prototype Architecture

The prototype is an incomplete version of an application created at an early stage to gather user responses. We decided to cut down the natural language processing unit and knowledge representation components for prototype development. Figure 3.2 shows architecture for prototype implementation. We are using Dynamo DB in place of semantic knowledge representation system and Alexa's built-in service to convert speech to text for our natural language processing unit. The controller now uses S3 as cloud storage for songs, Dynamo DB to fetch song information and music recommendation to recommend songs.

The prototype also involves the development of a Web API server to handle requests from AWS Lambda for Spotify data. The prototype supports two different sources for music information, a user's personal music library and Spotify. AWS Lambda will communicate with the API service using GET requests.

Chapter 4

Implementation

This chapter talks about the implementation of the different components discussed in the previous chapter. We begin by describing the various libraries and technologies used in the project followed by how the MIR component is implement using S3, AWS Lambda and Dynamo DB. The Alexa Skills kit with special reference to custom slot type is then explained with respect to the prototype implementation. The Prototype also answers queries on the use music from the Spotify, therefore we will discuss the flask implementation for a web server hosted on Heroku. Lastly, we throw some light on the deployment process used to manage serverless deployments onto AWS Lambda and Heroku.

4.1 Technology Stack

4.1.1 Libraries

A software library consists of pre-written code that can be used to develop software programs and applications. These libraries are language specific, table 4.1 shows Python specific libraries used in this project. The process of selecting the right library to use depends on various factors like available documentation, platform compatibility, required functionality and learning curve. LibROSA was selected over other Python based MIR libraries like PyMIR and MIReval because its mature, has a lot of functionality and good documentation is available.

Library	Description
LibROSA	LibROSA ia a Python based library for any type of audio, specially music analysis. If you are working on a Music Information Retrieval systems, this library provides a bunch of functions that can be used to analyse and extract features from the music files.
Spotipy	Spotipy is a Python based client library for Spotify Web API. It is capable of doing authorized API requests to the Spotify API for use related confidential data. It has tons of documentation and an easy learning curve.
NumPy	NumPy is the fundamental package for scientific computing with Python. NumPy can also work with generic data and arbitrary data-types can be defined which makes it more effective to use with databases.
Pynamo DB	A Python based interface for Amazon's Dynamo DB. It supports Unicode, Binary, JSON, Number, Set, and UTC Datetime attributes.
IPython	IPython provides a rich architecture for Python used for interactive computing. It is compatible with Python 2.7 and Python3.3 and up. We used it for its GUI capabilities like displaying graph and playing audios.

Table 4.1: List of Python specific libraries used to develop the prototype.

4.1.2 Tools and Frameworks

A framework is a reusable software component that provides a standard way to build and deploy applications. They speed up the development process by including support programs or tool sets. Selecting a specific framework or tool for software development is a task in itself. Python has a number of frameworks for web development like Django, Pyramid and Flask. We selected Flask for our prototype because of previous experience with it and it's a minimalistic solution that can be scaled later on to a finished product. Virtual environment for Python development and package management can be together taken care by Pipenv. Pipenv does the job of PIP and Virtualenv but is an overkill for a small prototype project.

Tool	Description
Virtualenv	Virtualenv is used to create Python development environment on the Local system. It helps to address the problem of dependencies and library versions by creating an environment that has its own installation directories.
Serverless	Serverless is a toolkit that provides a command-line interface for deploying and managing serverless functions. It works with AWS Lambda, and also supports other cloud providers serverless implementations.
Anaconda	With over 6 million users, the open source Anaconda Distribution is the fastest and easiest way to do Python and R data science and machine learning on Linux, Windows, and Mac OS X. It's the industry standard for developing, testing, and training on a single machine.
PIP	Pip is a package management system used to install and manage software packages written in Python.
Flask	Flask is a Python based web framework. It can be used to make web applications and RESTful API's. Flask applications are lightweight and fast because it implements a bare-minimum and leaves the rest on developer.
Zappa	Zappa helps to build and deploy server-less Python applications on AWS Lambda + API Gateway. It gives each request is given its own virtual HTTP "server" by Amazon API Gateway so that no request times out even at the time of heavy traffic.
Heroku CLI	Heroku CLI is a set of commands that work directly from on your terminal to create and manage Heroku apps. The CLI provides functionality to bundle and deploy Heroku apps directly to the Heroku server.

Table 4.2: List of Tools and Frameworks

4.1.3 Technologies

Technologies in the field of Software Development are never constant. Spotify Web API was used to demonstrate a third-party music service compatibility in our prototype. Spotify Web API was used over Google Music because it was easy to implement with the availability of Python-based client library called Spotipy.

Technology	Description
Python 2.7	This is previous stable release version of Python, the Lambda function from amazon Web Services supports Python 2.7. Python is high-level general purpose programming language used for various programming problems. Its widely used in modern computer science branches like artificial intelligence and machine learning.
Spotify Web API	Based on simple REST principles, the Spotify Web API endpoints return JSON metadata about music artists, albums, and tracks, directly from the Spotify Data Catalogue.
Dynamo DB	Dynamo DB is a NoSQL database with key-value and document data structures. Dynamo DB is capable of delivering really fast query responses even is the scale of data is really high. The database features built-in support for backup and restores, security and in-memory cashing for handing internet applications.
JSON	JSON stands for JavaScript Object Notation, it is majorly used as a data format for asynchronous web server and client communication. JSON is language independent and therefore can be used with various programming languages like C, C++, Python, and many others. JSON can transmit/represents data in attribute-value pairs and ordered list of values (array).

Table 4.3: List of Technologies

4.1.4 Introduction to Amazon Web Services

With the rise of on-demand computing technology, a lot of IT giants started offering cloud computing services to their customers. In 2006 Amazon.com launched Amazon Web Services, an on-demand cloud computing platform to customers on a paid basis. The service allows subscribers to access all the available virtual resources through the internet. They can use any modern-day web browser to log in, configure and use their virtual resources just like a real computer. The initial launch in 2006 involved availability of three services S3 as the cloud storage, SQS as fully managed message queuing service and EC2 as web service that provides secure, resizable compute capacity in the cloud. In 2017, AWS now includes more than 90 services related to computing, storage, database, networking, deployment, mobile, analytic and developer tools.

4.2 Music Information Retrieval

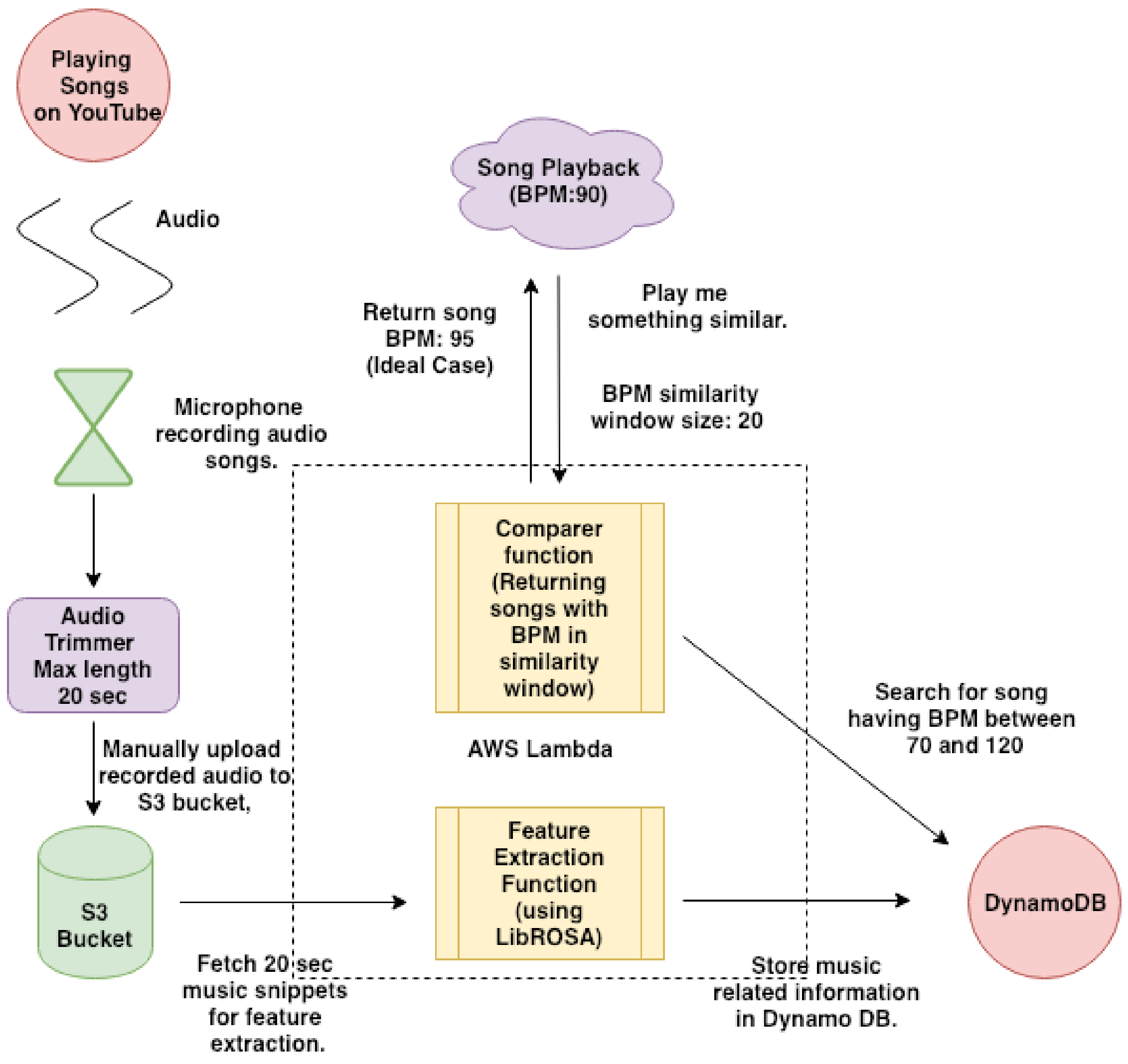

Playing Songs on YouTube
Audio
Microphone recording audio songs.
Audio Trimmer Max length 20 sec
Manually upload recorded audio to S3 bucket,
S3 Bucket
Fetch 20 sec music snippets for feature extraction.
Song Playback (BPM:90)
Return song BPM: 95 (Ideal Case)
Play me something similar.
BPM similarity window size: 20
Comparer function (Returning songs with BPM in similarity window)
AWS Lambda
Feature Extraction Function (using LibROSA)
Search for song having BPM between 70 and 120
Store music related information in Dynamo DB.
DynamoDB

Figure 4.1: MIR component implementation.

MIR component is the most complex component in the prototype because it uses multiple services to function. It consists of AWS Lambda for serverless processing of music files, S3 for music playback and Dynamo DB to fetch similar songs. For better understanding, we have divided the complex functionality into individual smaller components.

4.2.1 Audio Processing

The first step in the process is to upload music files on the S3 bucket. Songs carry copyright information, uploading a song to S3 can result in the violation of copyrights and immediate deletion. To overcome this problem, each song was played on YouTube as a video and recorded by a microphone to scrape of all the copyright information. Figure 4.1 shows on the top left corner the process of recording audio. The recorded music files were then trimmed down to 20 sec for faster processing.

4.2.2 Feature Extraction (AWS Lambda)

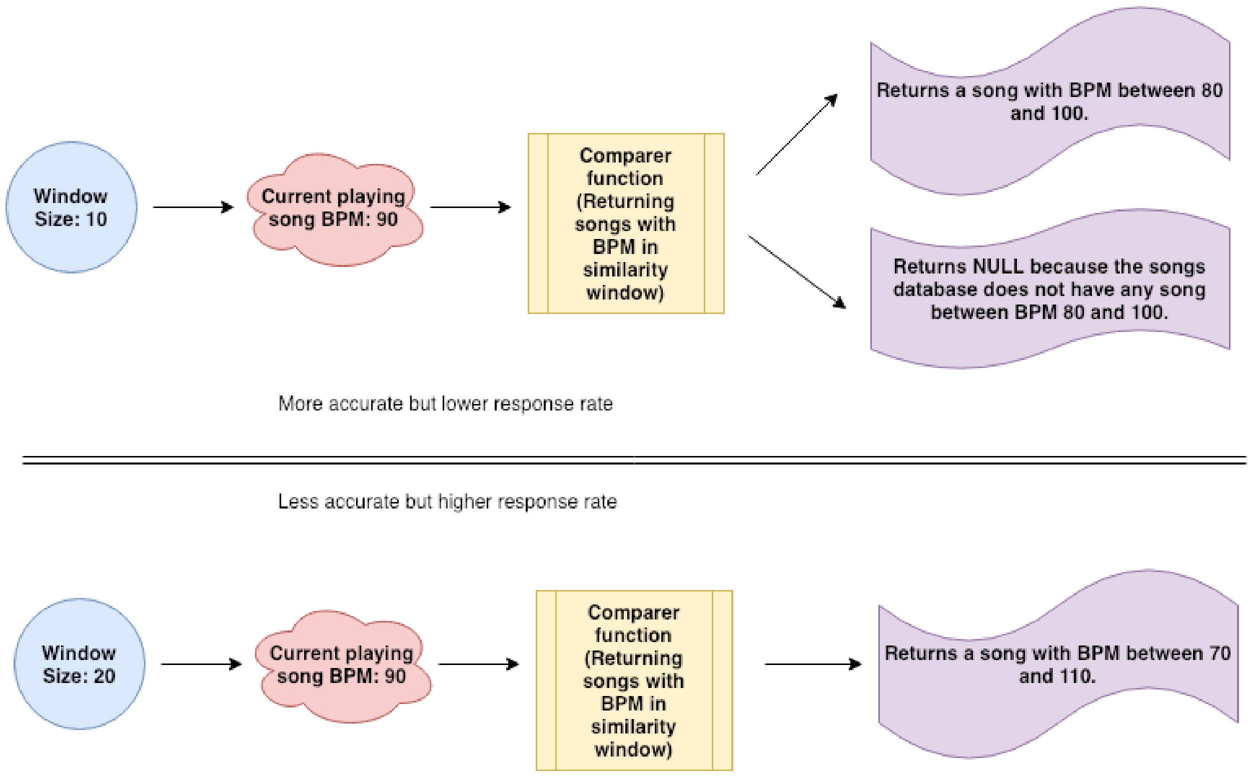
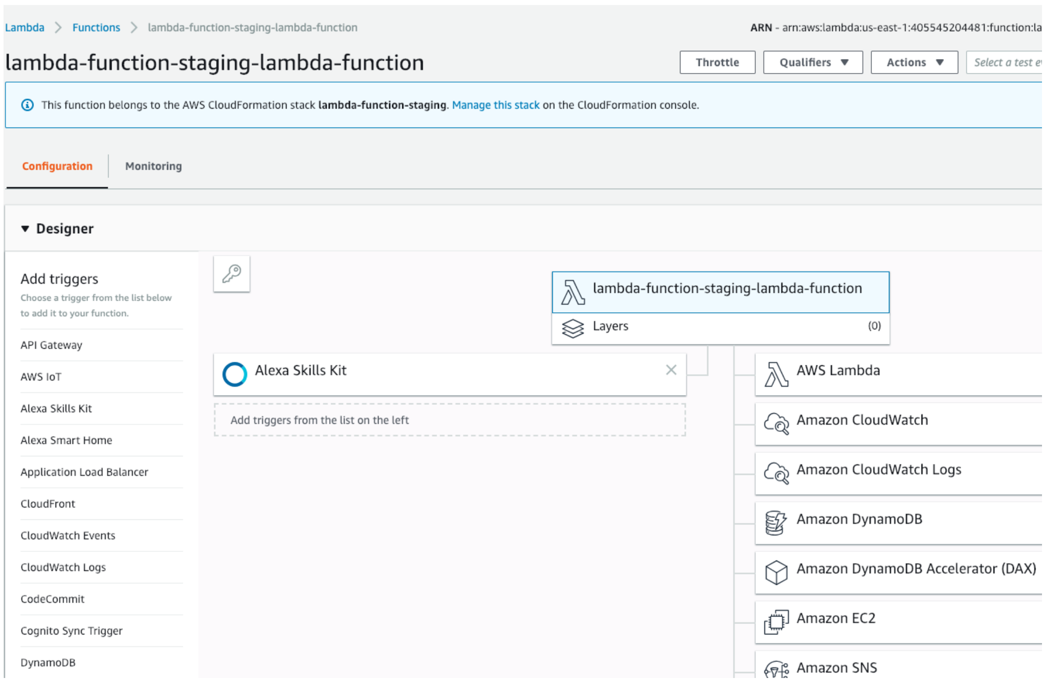

Window Size: 10
Current playing song BPM: 90
Comparer function (Returning songs with BPM in similarity window)
Returns a song with BPM between 80 and 100.
Returns NULL because the songs database does not have any song between BPM 80 and 100.
More accurate but lower response rate
Less accurate but higher response rate
Window Size: 20
Current playing song BPM: 90
Comparer function (Returning songs with BPM in similarity window)
Returns a song with BPM between 70 and 110.

Figure 4.2: Comparer function to return similar songs.

The second step is to extract features from the processed audio files stored in the S3 bucket using an AWS Lambda function. AWS Lambda is a serverless computing platform provides by AWS that runs code in the response to the triggers attached to various events. Lambda automatically decides on the required resources and computational power. This function uses the LibROSA library to extract features like BPM (beats per minute). BPM is used to measure the tempo of a song. If the value of BPM is high the song will have a higher tempo and hence fast music. After feature extraction, the Lambda function uses API calls from Pynamo DB library to access Dynamo DB and insert values related to each song.

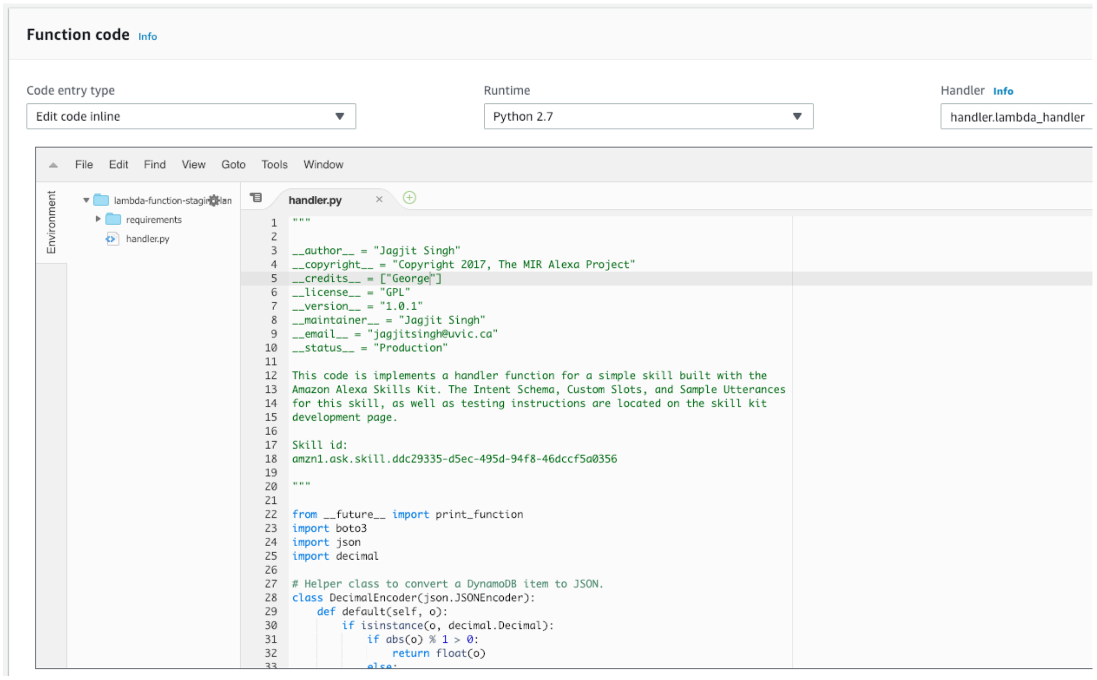
4.2.3 Comparer (AWS Lambda)


Lambda > Functions > lambda-function-staging-lambda-function
ARN - arn:aws:lambda:us-east-1:405545204481:function:la
lambda-function-staging-lambda-function
Throttle
Qualifiers ▼
Actions ▼
Select a test e
Configuration
Monitoring
▼ Designer
Add triggers
Choose a trigger from the list below to add it to your function.
API Gateway
AWS IoT
Alexa Skills Kit
Alexa Smart Home
Application Load Balancer
CloudFront
CloudWatch Events
CloudWatch Logs
CodeCommit
Cognito Sync Trigger
DynamoDB
lambda-function-staging-lambda-function
Layers (0)
Alexa Skills Kit
Add triggers from the list on the left
AWS Lambda
Amazon CloudWatch
Amazon CloudWatch Logs
Amazon DynamoDB
Amazon DynamoDB Accelerator (DAX)
Amazon EC2
Amazon SNS

This function belongs to the AWS CloudFormation stack **lambda-function-staging**. Manage this stack on the CloudFormation console.

Figure 4.3: AWS Lambda - Triggers and Services.

Third Step in the process is to use the similarity window size and BPM from the currently running song to return a song faster, slower or similar depending on what user requested. Similarity window size can be adjusted according to the size of the song database. If the song database is small, larger similarity window size is required to achieve a better response rate and vice-versa. Figure 4.2 illustrates how the change in similarity window size affects the response rate of the prototype. The code can be easily attached via available triggers to the other AWS services or any other web or mobile app. In our case, we have used triggers from Alexa Skills Kit to run the serverless code. Figure 4.3 shows the AWS Lambda configuration dialogue.



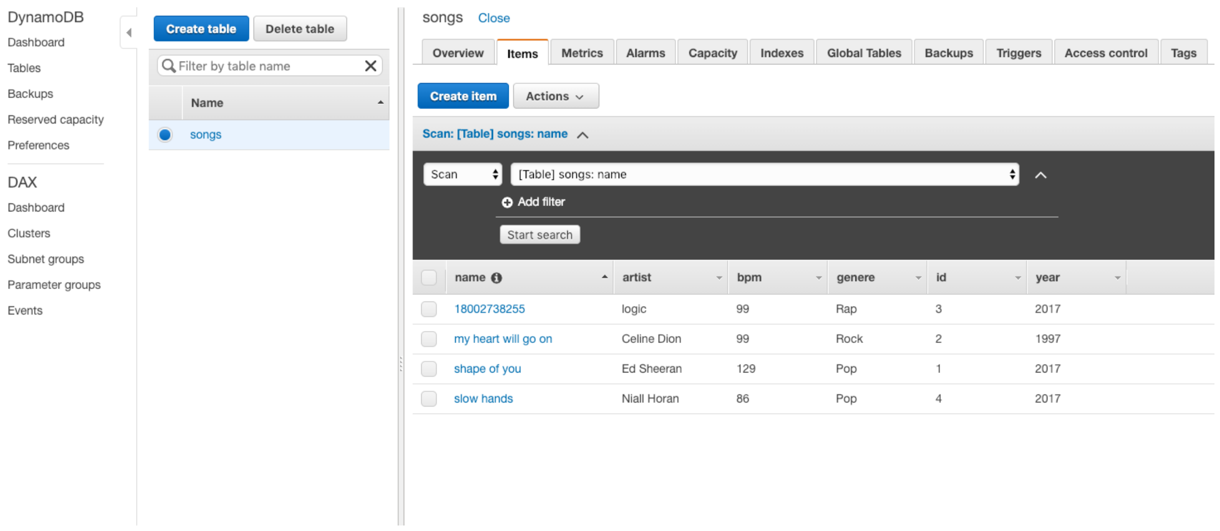
Code entry type: Edit code inline
Runtime: Python 2.7
Handler: handler.lambda_handler

handler.py

```python
"""

__author__ = "Jagjit Singh"
__copyright__ = "Copyright 2017, The MIR Alexa Project"
__credits__ = ["George"]
__license__ = "GPL"
__version__ = "1.0.1"
__maintainer__ = "Jagjit Singh"
__email__ = "jagjitsingh@uvic.ca"
__status__ = "Production"

This code is implements a handler function for a simple skill built with the
Amazon Alexa Skills Kit. The Intent Schema, Custom Slots, and Sample Utterances
for this skill, as well as testing instructions are located on the skill kit
development page.

Skill id:
amzn1.ask.skill.ddc29335-d5ec-495d-94f8-46dccf5a0356

"""

from __future__ import print_function
import boto3
import json
import decimal

# Helper class to convert a DynamoDB item to JSON.
class DecimalEncoder(json.JSONEncoder):
    def default(self, o):
        if isinstance(o, decimal.Decimal):
            if abs(o) % 1 > 0:
                return float(o)
```

Figure 4.4: AWS Lambda function code.

4.2.4 Dynamo DB Feature Storage



Scan: [Table] songs: name

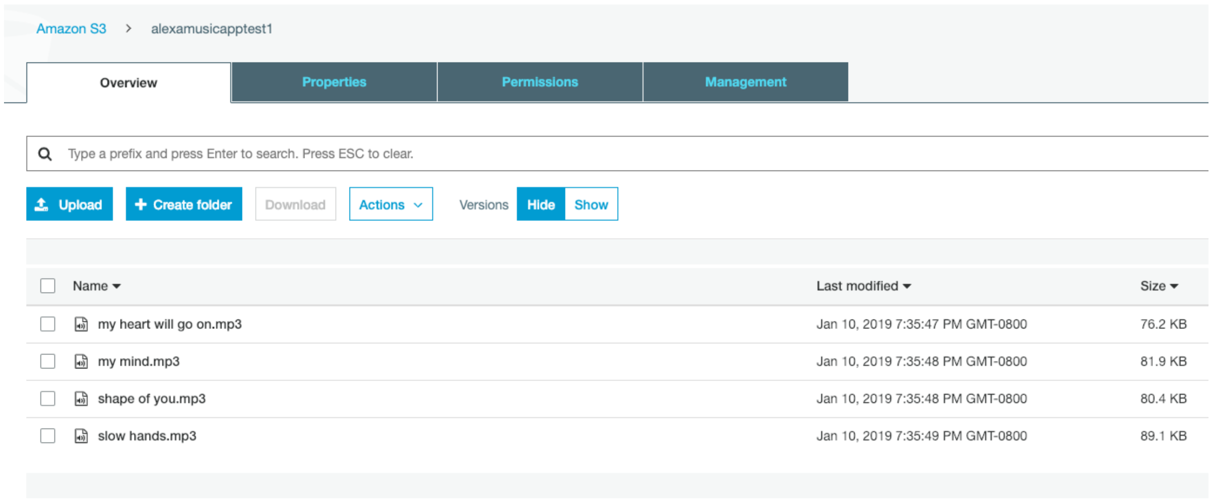
Scan | [Table] songs: name

name	artist	bpm	genere	id	year
18002738255	logic	99	Rap	3	2017
my heart will go on	Celine Dion	99	Rock	2	1997
shape of you	Ed Sheeran	129	Pop	1	2017
slow hands	Niall Horan	86	Pop	4	2017

Figure 4.5: Dynamo DB key-values storage for song information.

MIR functionality is build using serverless AWS Lambda functions. Storing and querying data has been done using AWS Dynamo DB. Lambda functions are written in Python 2.7 and use Pynamo DB (Python interface to query Dynamo DB) to create tables, add music information and query when requested by the user. Dynamo DB is a NoSQL database with key-value and document data structures. Dynamo DB is capable of delivering really fast query responses even is the scale of data is really high. The database features built-in support for backup and restores, security and in-memory cashing for handing internet applications. Dynamo DB can handle more than 10 trillion requests per day and support peaks of more than 20 million requests per second. Dynamo DB stores information related to the songs, where each song is represented as item and has values for name, artist, BPM, genre and year. Figure 4.5 shows songs table from the AWS Dynamo DB console.

4.2.5 S3


Amazon S3 > alexamusicapptest1
Overview
Properties
Permissions
Management
Type a prefix and press Enter to search. Press ESC to clear.
Upload
Create folder
Download
Actions
Versions
Hide
Show

Name	Last modified	Size
my heart will go on.mp3	Jan 10, 2019 7:35:47 PM GMT-0800	76.2 KB
my mind.mp3	Jan 10, 2019 7:35:48 PM GMT-0800	81.9 KB
shape of you.mp3	Jan 10, 2019 7:35:48 PM GMT-0800	80.4 KB
slow hands.mp3	Jan 10, 2019 7:35:49 PM GMT-0800	89.1 KB

Figure 4.6: S3 song storage bucket.

AWS makes it easy to connect and access data between its different services, as a result, we decided to use S3 for storing audio songs in buckets. Amazon Simple Storage Service (S3) is an object storage service offered by AWS with scalability, security, availability, and performance in mind. A bucket in S3 is just like a container that you can use to store data and give access permissions. AWS Lambda functions can access data from these buckets if the AWS user executing the functions has access to it.

4.3 Alexa Skills Kit

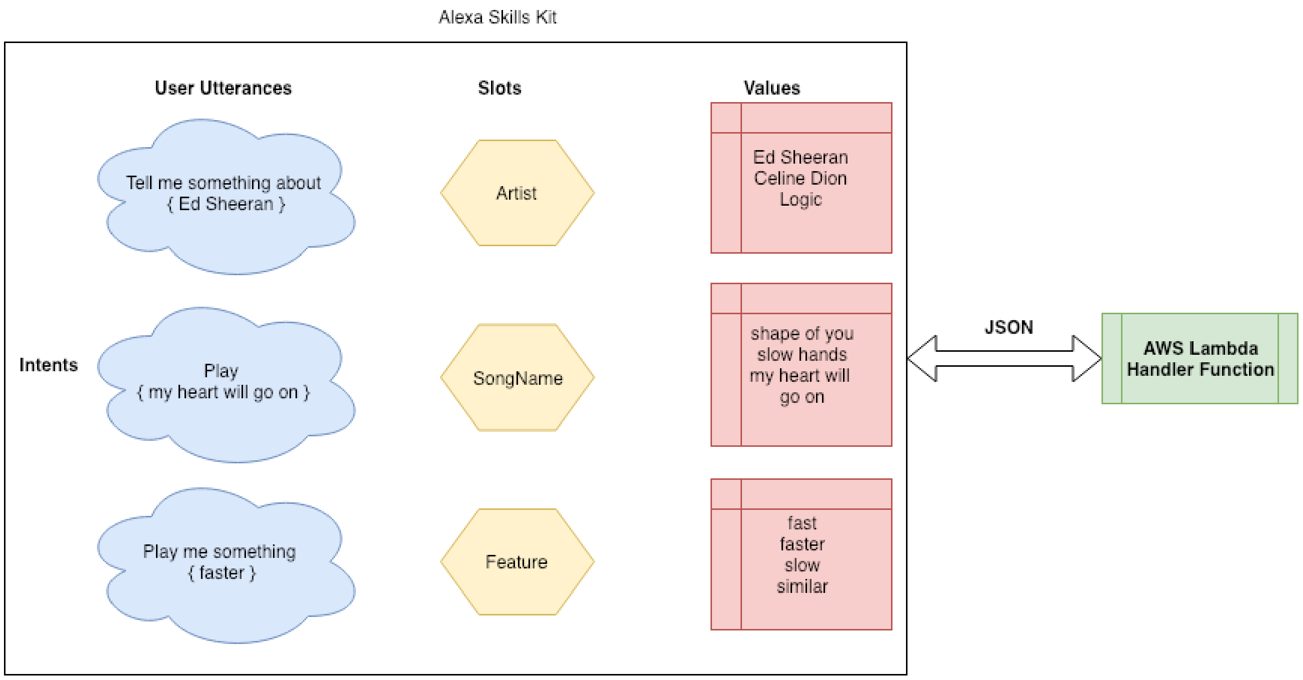
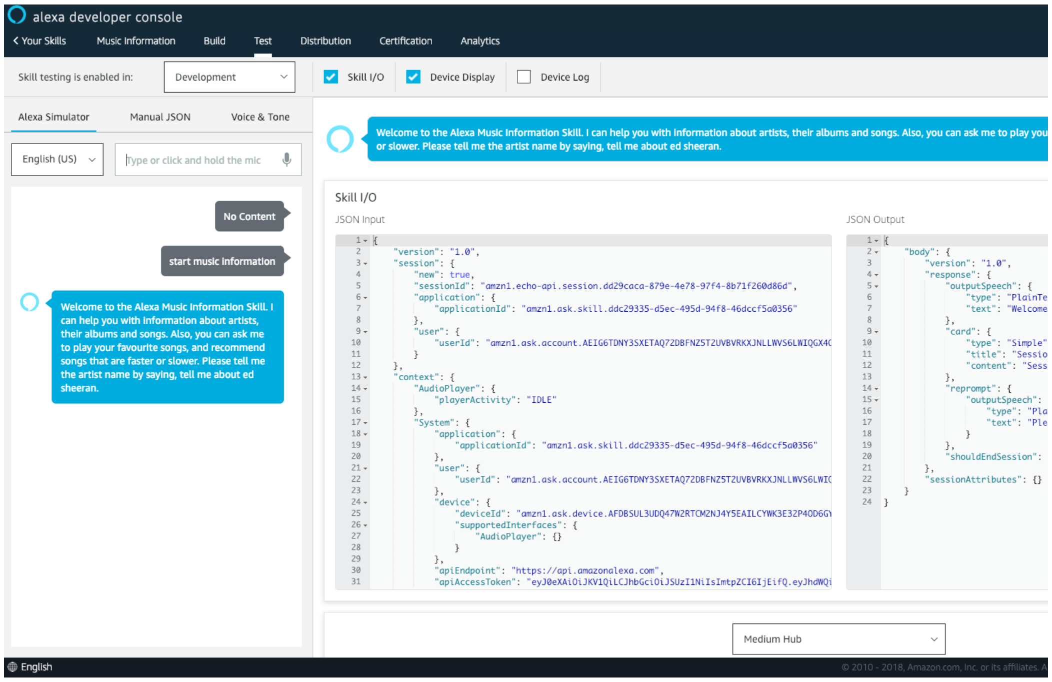


Figure 4.7: Alexa Skills Kit - Intents, Utterances, Slots and Values.



Welcome to the Alexa Music Information Skill. I can help you with information about artists, their albums and songs. Also, you can ask me to play your favourite songs, and recommend songs that are faster or slower. Please tell me the artist name by saying, tell me about ed sheeran.

JSON Input

```
{
    "version": "1.0",
    "session": {
        "new": true,
        "sessionId": "amzn1.echo-api.session.dd29caca-879e-4e78-97f4-8b71f260d86d",
        "application": {
            "applicationId": "amzn1.ask.skill.ddc29335-d5ec-495d-94f8-46dccf5a0356"
        },
        "user": {
            "userId": "amzn1.ask.account.AEIG6TDNY3SXETAQ72DBFNZ5T2UVBVRKXJNLLWVS6LWIQGX4C
        }
    },
    "context": {
        "AudioPlayer": {
            "playerActivity": "IDLE"
        },
        "System": {
            "application": {
                "applicationId": "amzn1.ask.skill.ddc29335-d5ec-495d-94f8-46dccf5a0356"
            },
            "user": {
                "userId": "amzn1.ask.account.AEIG6TDNY3SXETAQ72DBFNZ5T2UVBVRKXJNLLWVS6LWIC
            },
            "device": {
                "deviceId": "amzn1.ask.device.AFDBSUL3UDQ47W2RTCM2NJ4Y5EAILCYWK3E32P4OD6GY
                "supportedInterfaces": {
                    "AudioPlayer": {}
                }
            },
            "apiEndpoint": "https://api.amazonalexa.com",
            "apiAccessToken": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6IjEifQ.eyJhdWQi
```

JSON Output

```
{
    "body": {
        "version": "1.0",
        "response": {
            "outputSpeech": {
                "type": "PlainTe
                "text": "Welcome
            },
            "card": {
                "type": "Simple"
                "title": "Sessio
                "content": "Sess
            },
            "reprompt": {
                "outputSpeech":
                    "type": "Pla
                    "text": "Ple
                }
            },
            "shouldEndSession":
        },
        "sessionAttributes": {}
    }
}
```

Figure 4.8: Alexa developer console.

Alexa is Amazon's smart conversational agent. Adding capabilities or skills to it is easily done via Alexa Skills Kit. This kit includes a collection of self-serve API's, documentation and tools that help in the skill development. Figure 4.7 shows various components required to design an Alexa skill. We will discuss them in detail:

4.3.1 Intents and Utterances

Intents are actions that a skill should perform to fulfil a user's spoken request. Intents act as triggers that call a specific function in AWS Lambda. A user intent for "play me something similar" will call the comparer function from AWS Lambda. Utterances are sample phrases that a user may say to invoke a particular intent. Once you define the utterances, Alexa Skills kit links sample utterances to intents and sends triggers to AWS function with a JSON object containing the request parameters.

4.3.2 Slots and Values

Slots act as variables in sample utterances, they can take different values depending on their type and defined values. In figure 4.7, for utterance "Tell me something about Ed Sheeran" the slot type is Artist and can take different values like Drake, Eminem and Celine Dion.

4.3.3 Custom Slots

Slot values are expected to be finite and listed before building the skill model. Custom slot types were introduced in 2017 when they replaced Amazon.Literal to give developers the freedom to add any number of values for a slot during run-time. Custom slot requires some initial values to help Alexa direct that user request to right intent. In our case we have used custom slot types for the intents "List top songs by artist x" and "List artist like x" where x can take any run-time value.

Alexa Skills kit provide developers with a testing console that can take user requests just like an Alexa device in text or speech format. It also displays the JSON object that is sent as a request to our AWS Lambda functions and the response returned from them. Figure 4.8 shows Alexa developer console.

4.4 Spotify Web API Server with Heroku

The prototype implementation supports fetching user music data from two different sources, user music library and Spotify. Spotify is a paid music subscription service that provides a Web API for requesting music information. To speed up the process of development we used a Python-based client library for Spotify Web API that provides full access to all the music data on the Spotify platform. A web API server was developed using Flask which is a light-weight framework for Python web applications. Flask applications are written in Python with templating done using HTML. We designed three different API functions hosted on Heroku to provide information on 'related artist', 'artist top songs' and 'song URL'. These functions were accessed from the AWS Lambda functions using simple API GET request. GET method is the most common HTTP method used to request data from a specified resource. The GET request includes 'artist name' which when passed as a parameter to these functions returned a JSON object as response. The response was trimmed to top 5 and narrated by the Alexa skill.

4.5 Serverless Deployment

The last step in the implementation process is to deploy the local running code on the servers. Serverless deployments are different from the conventional ones because they do not have control over the environment that will run their code in production or staging servers. We will outline the process of deployment used in our prototype implementation (Figure 4.9).

1. Preparing local environment - Working with complex prototypes require specific versions of various software libraries. Projects are developed within their virtual environment created on the local machine by tools like virtualenv. These tools install all the required libraries in the environment that makes it easy for the deployment tools to zip contents and deploy.

2. Adding configuration files - Different deployment frameworks have different configuration file name and format. Serverless uses .yml format to get user details like AWS access key ID and secret key. Configuration files also include information about the application name, it's version and access permissions. We used

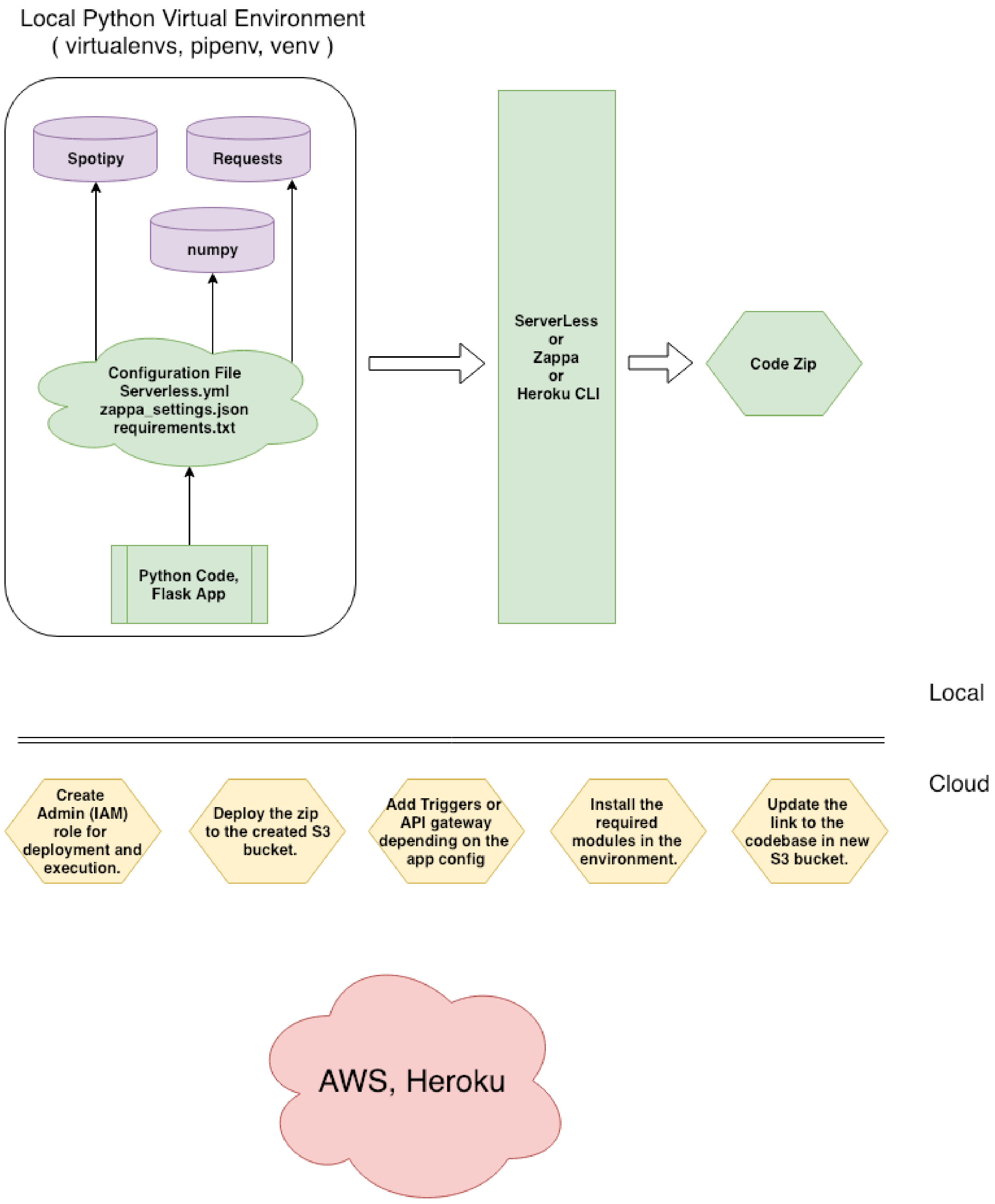

Local Python Virtual Environment
(virtualenvs, pipenv, venv)
Spotipy
Requests
numpy
Configuration File
Serverless.yml
zappa_settings.json
requirements.txt
Python Code,
Flask App
ServerLess
or
Zappa
or
Heroku CLI
Code Zip
Local
Cloud
Create Admin (IAM) role for deployment and execution.
Deploy the zip to the created S3 bucket.
Add Triggers or API gateway depending on the app config
Install the required modules in the environment.
Update the link to the codebase in new S3 bucket.
AWS, Heroku

Figure 4.9: Deployment process.

a serverless framework to deploy the AWS function for Alexa skill and Heroku CLI for Spotify Web API.

3. Using deployment frameworks - Deployment tools can be run from the command line, they use the configuration file to get project information and generate a zip file containing project code. A series of steps are performed on the cloud environment like making required roles and permissions for deployment, copying the code zip to temporary cloud storage like S3, installing required modules from the configuration files on the cloud environment and adding other configurations for Triggers and API Gateway.

Chapter 5

Limitations and Future Work

An important part of the music recommendation is to listen to user audio and recommend songs accordingly. Unfortunately, the available voice to text smart home agents like Alexa and Google Home do not support this type of functionality. Considering privacy and security reasons, these devices do not allow processing of raw audio from the user microphone and thus resulting in blocking access to raw user music. Also, certain features have been compromised due to the fact that using a number of components in the system, limits their compatibility with each other and introduce a number of integration and data flow problems. Certainly, we have tried to accommodate as much as possible by using various wrappers at their interfaces.

The prototype works with user music library and external music sources like Spotify. Music is copyrighted which makes it hard to store and process on the cloud. We recorded 20 seconds clips from the microphone by playing the music on YouTube to avoid the copyright violations. The resulted audio had poor quality and maximum length of 30 seconds.

Recommendation works best if you have a large data-set. The prototype development was done using only 4 songs from the user music database that resulted in large similarity window and roughly correct recommendations. We had fewer songs to save time on audio processing done to avoid copyright violations.

The API web server for requesting Spotify user data was hosted using Heroku cloud platform. Heroku unloads the app from the server memory when it is not used for a while to save resources, this causes a delay in the first GET request. Also, resource and access management are easy if both the services run on the same cloud platform.

The prototype aims to simplify the task of listing and accessing information from the music data, this, when supported with a user study, will definitely be a great research and landmark in the field of conversational agents for music exploration. We are looking to develop and do a user study based on our prototype to get more information and improve the product. Secondly, we wish to replace Alexa with a conventional voice to text agent to have better control over the processing and achieve the functionality of receiving the raw user audio through the microphone. Thirdly, recommendation using multiple features is more accurate, we wish to extract more features from the audio files using native MIR methods. For example, in addition to BPM we can compare features like danceability, loudness, valence and energy.

Chapter 6

Results

Prototype testing after successful development was done using ten generic sentences gathered from a pilot user survey. The pilot user survey involved asking five close friends to list ten questions they would ask to a voice-controlled music information retrieval system. The obtained questions were compared and the most common ten questions were selected to evaluate the prototype which is listed below in the left column of the table. The prototype was then tested by asking these questions, if it answered something, we filled the response column with a 'Yes' otherwise we filled it with a 'No'. Prototype in some cases was able to answer but the response was not correct, we added a 'No' in the correct column for that question. The prototype had an accuracy of 8/10 with 9/10 response rate when answering these questions.

The prototypes can never fully satisfy user expectations because they are still in an early development stage. The last question about "when this song was released" was expected from the user survey but was not handled in the prototype. The sixth question "I want to hear something similar" gave an incorrect response because the window used to match similar songs was made big enough to accommodate a valid response from a smaller song database. The specified architecture can be further used for development of similar speech-based music applications with little modifications and the Prototype can be used for a user survey on a large scale to capture user-expectations from such a system.

Question	**Response**	**Correct**
Tell me something about Drake	Yes	Yes
Who is Ed Sheeran	Yes	Yes
I want to hear 'shape of you'	Yes	Yes
Play me 'my Heart will go on'	Yes	Yes
Play me something faster	Yes	Yes
I want to hear something similar	Yes	No
Play something slower	Yes	Yes
List top songs by this artist	Yes	Yes
Similar artists like Drake	Yes	Yes
When this song was released	No	No

Table 6.1: List of questions used to evaluate the prototype.

Bibliography

[1] Sameera A Abdul-Kader and JC Woods. Survey on chatbot design techniques in speech conversation systems. *International Journal of Advanced Computer Science and Applications*, 6(7), 2015.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge & Data Engineering*, (6):734–749, 2005.

[3] Maurizio Atzori, Ludovico Boratto, and Lucio Davide Spano. Towards chatbots as recommendation interfaces. In *CEUR WORKSHOP PROCEEDINGS*, volume 1945, pages 28–31. CEUR-WS, 2017.

[4] Jerome R Bellegarda. Large-scale personal assistant technology deployment: the siri experience. In *INTERSPEECH*, pages 2029–2033, 2013.

[5] Ronald J Brachman, Hector J Levesque, and Raymond Reiter. *Knowledge representation*. MIT press, 1992.

[6] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[7] Justine Cassell, Joseph Sullivan, Elizabeth Churchill, and Scott Prevost. *Embodied conversational agents*. MIT press, 2000.

[8] Oscar Celma. Music recommendation. In *Music recommendation and discovery*, pages 43–85. Springer, 2010.

[9] Hung-Chen Chen and Arbee LP Chen. A music recommendation system based on music data grouping and user interests. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 231–238. ACM, 2001.

[10] Leonard Chen, Sandra Cheng, Larry Birnbaum, and Kristian J Hammond. The interactive chef: a task-sensitive assistant. In *Proceedings of the 7th international conference on Intelligent user interfaces*, pages 234–234. ACM, 2002.

[11] Geumhwan Cho, Jusop Choi, Hyoungshick Kim, Sangwon Hyun, and Jungwoo Ryoo. Threat modeling and analysis of voice assistant applications.

[12] Gobinda G Chowdhury. Natural language processing. *Annual review of information science and technology*, 37(1):51–89, 2003.

[13] E Crawford. Bots are awesome! humans? not so much. chatbots magazine.(2016).

[14] Zachary Davis, Michael Hu, Shreyas Prasad, Michael Schuricht, PM Melliar-Smith, and Louise E Moser. A personal handheld multi-modal shopping assistant. In *Networking and Services, 2006. ICNS'06. International conference on*, pages 117–117. IEEE, 2006.

[15] John De Vet and Vincent Buil. A personal digital assistant as an advanced remote control for audio/video equipment. In *Proceedings of the Second Workshop on Human Computer Interaction with Mobile Devices*, pages 87–91, 1999.

[16] J Stephen Downie. Music information retrieval. *Annual review of information science and technology*, 37(1):295–340, 2003.

[17] Douglas Eck, Paul Lamere, Thierry Bertin-Mahieux, and Stephen Green. Automatic generation of social tags for music recommendation. In *Advances in neural information processing systems*, pages 385–392, 2008.

[18] Slim Essid, Yves Grenier, Mounira Maazaoui, Gaël Richard, and Robin Tournemenne. An audio-driven virtual dance-teaching assistant. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 675–678. ACM, 2011.

[19] Olivier Gouvert, Thomas Oberlin, and Cédric Févotte. Negative binomial matrix factorization for recommender systems. *arXiv preprint arXiv:1801.01708*, 2018.

[20] Mark Greene, Michael Hegarty, and Dermot Cantwell. System, method and computer program product for adding voice activation and voice control to a media player, September 9 2008. US Patent 7,424,431.

[21] Hui Guan, Wei Ru Chen, Han Li, and Jun Wang. Stride–based risk assessment for web application. In *Applied Mechanics and Materials*, volume 58, pages 1323–1328. Trans Tech Publ, 2011.

[22] Jeong-Hye Han, Dong-Ho Kim, and Jong-Won Kim. Physical learning activities with a teaching assistant robot in elementary school music class. In *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on*, pages 1406–1410. IEEE, 2009.

[23] Johann Hauswald, Michael A Laurenzano, Yunqi Zhang, Cheng Li, Austin Rovinski, Arjun Khurana, Ronald G Dreslinski, Trevor Mudge, Vinicius Petrucci, Lingjia Tang, et al. Sirius: An open end-to-end voice and vision personal assistant and its implications for future warehouse scale computers. In *ACM SIGPLAN Notices*, volume 50, pages 223–238. ACM, 2015.

[24] Wendell Hicken, Frode Holm, James Clune, and Marc Campbell. Music recommendation system and method, February 17 2005. US Patent App. 10/917,865.

[25] Jizhou Huang, Ming Zhou, and Dan Yang. Extracting chatbot knowledge from online discussion forums. In *IJCAI*, volume 7, pages 423–428, 2007.

[26] Chen Hung-Chen and Arbee LP Chen. A music recommendation system based on music and user grouping. *Journal of Intelligent Information Systems*, 24(2-3):113, 2005.

[27] Adrian Iftene and Jean Vanderdonckt. Moocbuddy: a chatbot for personalized learning with moocs. In *RoCHI–International Conference on Human-Computer Interaction*, page 91, 2016.

[28] Yuichiro Ikemoto, Varit Asawavetvutt, Kazuhiro Kuwabara, and Hung-Hsuan Huang. Conversation strategy of a chatbot for interactive recommendations. In *Asian Conference on Intelligent Information and Database Systems*, pages 117–126. Springer, 2018.

[29] Nicholas R Jennings and Michael Wooldridge. Applications of intelligent agents. In *Agent technology*, pages 3–28. Springer, 1998.

[30] Kyu-hong Kim, Jeong-Su Kim, and Ick-sang Han. Method and apparatus for searching for music based on speech recognition, October 9 2008. US Patent App. 11/892,137.

[31] Pavel Kucherbaev, Achilleas Psyllidis, and Alessandro Bozzon. Chatbots as conversational recommender systems in urban contexts. In *Proceedings of the International Workshop on Recommender Systems for Citizens*, page 6. ACM, 2017.

[32] Ekaterina Kurdyukova. Inspire, guide, and entertain: designing a mobile assistant for runners. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, page 75. ACM, 2009.

[33] Gustavo López, Luis Quesada, and Luis A Guerrero. Alexa vs. siri vs. cortana vs. google assistant: a comparison of speech-based natural user interfaces. In *International Conference on Applied Human Factors and Ergonomics*, pages 241–250. Springer, 2017.

[34] Ewa Luger and Abigail Sellen. Like having a really bad pa: the gulf between user expectation and experience of conversational agents. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 5286–5297. ACM, 2016.

[35] Rodger J McNab, Lloyd A Smith, Ian H Witten, Clare L Henderson, and Sally Jo Cunningham. Towards the digital music library: Tune retrieval from acoustic input. In *Proceedings of the first ACM international conference on Digital libraries*, pages 11–18. ACM, 1996.

[36] Liselotte Mettler, M Ibrahim, and W Jonat. One year of experience working with the aid of a robotic assistant (the voice-controlled optic holder aesop) in gynaecological endoscopic surgery. *Human reproduction (Oxford, England)*, 13(10):2748–2750, 1998.

[37] VF Munoz, Carlos Vara-Thorbeck, JG DeGabriel, JF Lozano, E Sanchez-Badajoz, Alfonso Garcia-Cerezo, Rosario Toscano, and A Jimenez-Garrido. A medical robotic assistant for minimally invasive surgery. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 3, pages 2901–2906. IEEE, 2000.

[38] Fedelucio Narducci, Marco de Gemmis, Pasquale Lops, and Giovanni Semeraro. Recommender systems in the internet of talking things (iott).

[39] Donald A Norman. Cognitive engineering. *User centered system design*, 31:61, 1986.

[40] Marius M Punt, Coen N Stefels, Cornelis A Grimbergen, and Jenny Dankelman. Evaluation of voice control, touch panel control and assistant control during steering of an endoscope. *Minimally Invasive Therapy & Allied Technologies*, 14(3):181–187, 2005.

[41] Raymond F Rettig and Michelle Vickrey. Natural language user interface, December 3 2015. US Patent App. 14/728,365.

[42] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach.* Malaysia; Pearson Education Limited,, 2016.

[43] Yoshiaki Shindo and Hiroshi Matsuda. Prototype of cyber teaching assistant. In *Advanced Learning Technologies, 2001. Proceedings. IEEE International Conference on*, pages 70–73. IEEE, 2001.

[44] Ippei Torii, Kaoruko Ohtani, Nahoko Shirahama, Takahito Niwa, and Naohiro Ishii. Voice output communication aid application for personal digital assistant for autistic children. In *2012 IEEE/ACIS 11th International Conference on Computer and Information Science*, pages 329–333. IEEE, 2012.

[45] Mario Villamizar, Oscar Garces, Lina Ochoa, Harold Castro, Lorena Salamanca, Mauricio Verano, Rubby Casallas, Santiago Gil, Carlos Valencia, Angee Zambrano, et al. Infrastructure cost comparison of running web applications in the cloud using aws lambda and monolithic and microservice architectures. In *Cluster, Cloud and Grid Computing (CCGrid), 2016 16th IEEE/ACM International Symposium on*, pages 179–182. IEEE, 2016.

[46] Michael Wooldridge and Nicholas R Jennings. Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(2):115–152, 1995.

[47] Huaigu Wu and Yuri Natchetoi. Mobile shopping assistant: integration of mobile applications and web services. In *Proceedings of the 16th international conference on World Wide Web*, pages 1259–1260. ACM, 2007.

[48] Michael Zimmermann, René Krishnan, Andreas Raabe, and Volker Seifert. Robot-assisted navigated neuroendoscopy. *Neurosurgery*, 51(6):1446–1452, 2002.