Creative Integrated Design, Fall 2016

Project Kalexa

with Amazon Web Service

D조

최영진 김준혁 이상훈

윤석찬(AWS, Tech Evangelist)

Table of Contents

1. Abstract

본 프로젝트에서는 Amazon Alexa에서 동작하는 음성 인식 기반의 응용 프로그램을 제작하였다. Alexa는 아마존에서 개발한 음성 인식 시스템으로, 간단한 음성 비서 기능에 더하여 사용자가 다양한 응용 프로그램을 추가할 수 있도록 되어 있다.

그러나 Alexa는 한국어를 지원하지 않아 한국어로 된 질문을 하거나 응답을 받을 수가 없다. 본 프로젝트는 이 점에 착안하여 Alexa에서 한국어 관련 서비스를 제공하는 것을 목표로 하였다. 한국어 번역 기능, 한국어 학습 기능, K-POP 컨텐츠 제공 등을 주요 요소로 정하여, 어플리케이션을 디자인하고 구현하였다.

해당 프로그램을 제작하기 위하여 AWS(아마존 웹 서비스)내의 다양한 서비스를 활용하였다. 주요 로직을 실행하는 Lambda, 파일 저장소의 역할을 하는 S3, 데이터 저장소의 역할을 하는 dynamoDB 등을 융합하여 서버를 구현하였다. 그리고 네이버의 기계번역 API, 음성합성 API를 사용하여 외국인들이 한국어 음성을 들을 수 있도록 하였다. 또한 멜론의 차트, 앨범, 아티스트 등의 API를 활용하여 K-POP 관련 컨텐츠를 제공하였다. 이 과정에서 여러 프레임워크들이 쓰였으며, 정확도 및 속도를 개선하기 위해 다양한 기법을 적용하였다.

2. Introduction

A. 프로젝트의 배경 및 중요성

최근 K-POP 스타들의 많은 해외 진출과 소셜 미디어의 발달로 인해 해외에서의 K-POP에 대한 열기가 뜨겁다. 해외 팬들이 다양한 한국어 컨텐츠를 접하면서 한국어 학습에 대한 수요 또한 증가하고 있는 추세이다. 이러한 측면에서 외국인들에게 K-POP에 대한 정보를 제공해 줄 뿐만 아니라 한국어를 간단하게 학습할 수 있고 올바른 발음을 들려주는 기능을 제공하는 것은 충분히 필요가치가 있다.

따라서 본 프로젝트의 목표는 크게 한국어 학습 제공, K-POP 정보 제공의 두 가지로 나뉜다. 현재 Amazon Alexa가 한국어 인식/발음을 모두 지원하지 않고 있는 점이 한국어 기반의 정보를 제공하는 프로젝트에 가장 큰 한계와 문제점이다. 이 제약 조건 속에서 어떤 방식으로 문제를 해결할지 설계하고, 완성도 있는 결과물을 구현하는 것이 본 프로젝트의 일차적 목표이다. 더 나아가서는 K-POP 음악 추천과 같은 심도 있는 기능을 제공하고, 정량적으로 측정 가능한 지표들을 개선하는 작업을 하고자 한다.

B. 프로젝트 개요

본 프로젝트에서 제공하려는 기능들을 정리하면 다음과 같다.

- 한국어 학습
  - 한국어 번역 및 원어민 발음 제공
  - 한국어 퀴즈
- K-POP 정보
  - K-POP 차트 정보 제공
  - 가사, 아티스트 및 그에 관련된 정보 제공
  - 음악 재생
- 음악 추천
  - 감정 기반 추천
  - 사용자 데이터 기반 추천

3. Background Study

A. 관련 접근 방법 / 기술 장단점 분석

A.1. Amazon Alexa

Figure 1 Amazon Echo

아마존 알렉사(Alexa)는 2015년 출시된 일종의 음성 비서이다. 에코(Echo)라는 원통형 스피커 형태로 출시되었는데, 이 안에는 음성 인식 프로그램인 Alexa가 들어 있다. Alexa는 시각, 날씨 알림 등 기본 기능은 물론이고 주요 뉴스나 특정 정보를 브리핑하는 기능을 제공한다. 아마존은 장차 Alexa가 집 및 사무실을 자동화하는 데 핵심적인 역할을 할 것이라고 기대하고 있다.

Alexa는 현재 한국어를 지원하지 않는다. Apple 사의 Siri 등 다른 음성 인식 인공지능 역시 한국어를 지원하는 경우를 찾기 어렵다. 본 프로젝트에서는 Alexa를 타 서비스들과 연계하여 한국어 관련 기능을 제공하는 것을 목적으로 한다. 한국어를 인식하고 한국어

를 구사하는 것은 가능하지도 않거니와 본 프로젝트의 범위에서 벗어난다. 그러나 영어 전용 프로그램인 Alexa에 한국어 번역 질의 응답이나 K-POP 등 인기 있는 한국 컨텐츠를 제공하는 기능을 추가하여 Alexa의 가용 범위를 늘리는 데 의의가 있다.

A.2. Amazon Web Service

Amazon은 세계적인 전자 상거래 회사이나, 지난 2006년 Amazon Web Service(AWS)라는 이름으로 클라우드 컴퓨팅 서비스를 시작하였다. AWS는 Amazon.com에서 축적된 기술과 저렴한 가격을 바탕으로 급속도로 성장하였고, 전 세계적으로 가장 널리 이용되는 클라우드 서비스가 되었다. 현재는 Amazon.com이 AWS의 가장 큰 고객이라는 주객이 전도된 듯한 말까지 나오는 상황이다.

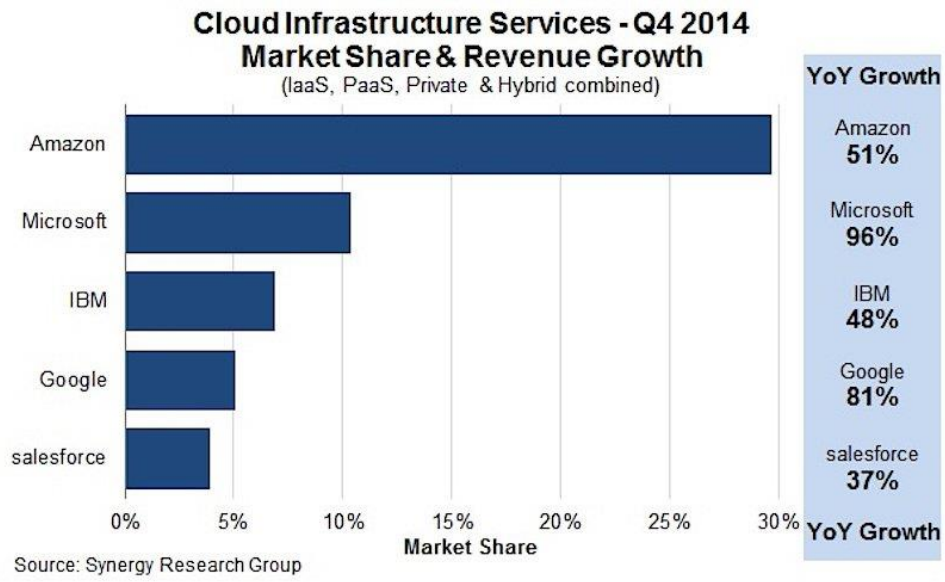

Figure 2 Cloud Service 시장 점유율

본 프로젝트는 AWS 코리아와 진행하는 것이기도 하고, Alexa가 아마존의 서비스이기 때문에 AWS와의 연동이 용이하다는 장점 때문에 AWS의 솔루션들을 사용하기로 결정하였다.

A.3. AWS Lambda

아마존 웹 서비스의 람다(Lambda)는 컴퓨터과학에서 익명 함수(Anonymous Function)을 지칭하는 용어에서 유래하였다. 프로그래밍 언어에서 익명 함수를 복잡한 선언 과정 없이 사용하듯이, 프로그래머가 개발 환경에 신경쓰지 않고 로직에만 신경쓸 수 있도록 한다. 예를 들어, 간단한 HTTP 동작을 처리하기 위해서도 일반적으로 서버를 구축하고 기본적인 웹 개발 환경을 갖추어야 한다. 그러나 아마존 람다는 해당 기능만 구현하면 서버를 관리할 필요 없이 AWS의 환경에서 실행이 가능하다. 기능에 맞는 환경을 편리하

게 선택 가능하며, 호출하는 시점만 서버 자원을 사용하므로 가격 면에서도 합리적이다.

본 프로젝트는 기능 자체는 많으나 외부 인터페이스와의 연결이 큰 부분을 차지하고 있어 코드 자체의 규모는 그리 크지 않을 것으로 예상된다. 또한 서버 머신이나 운영 체제 등에 종속될 경우도 없다고 판단되기에, 간편하게 로직을 구현하고 실행할 수 있는 람다가 적합하다고 판단하였다. EC2 등 서버를 사용하면 서버 환경 설정 및 관리에 불필요한 시간이 들어가고, 비용도 더욱 발생하는 데 반해 유리한 점이 많다고 할 수 있다.

A.4. AWS S3

AWS S3은 Simple Storage Service의 약자로, 빠르고 안정적인 저장 공간을 제공하는 솔루션이다. 또한 서비스가 성장함에 따라 필연적으로 저장되는 데이터도 늘어나는데, 이를 자동으로 처리해 줌으로서 Scalablity에 영향을 준다. 본 서비스에서는 한국어 재생을 위한 음성 MP3 파일의 저장에 필수적이다.

A.5. AWS DynamoDB

DynamoDB는 NoSQL 기반 데이터베이스이다. 본 프로젝트의 경우 복잡한 테이블 구조가 필요하지 않아, 빠르게 구현할 수 있는 DynamoDB를 선택하였다. 유저 정보, 곡 정보 등이 저장되게 된다.

A.6. 네이버 기계 번역 / 음성 합성 API

국내 유명 포털인 네이버는 개발자 저변을 확대하면서 여러 공개 API들을 제공하고 있다. 본 프로젝트에서는 그 중 기계 번역 및 음성 합성 API를 사용할 것이다. 기계 번역은 사람의 손을 거치지 않아 아직까지는 번역의 질이 떨어지나, Alexa와 같은 실시간 서비스에서 즉시 응답을 줄 수 있다는 장점이 있다. 또한 네이버의 기계 번역은 한국어 품질이 수준급이라고 알려져 있다. Alexa에서 한국어 번역 질의가 들어왔을 때, 이를 한국어 응답으로 되돌려 주기 위해 사용한다.

음성 합성 API란 TTS(Text to Speech) 서비스라고도 불리는데, 텍스트를 음성으로 변환한다. Alexa는 영어 응답을 지정할 수 있으나 한국어는 발음 자체가 불가능하다. 음성 합성 사용은 이를 해결하기 위한 아이디어로, 음성 합성 API에서 반환된 MP3 파일은 언어와 관계 없이 재생이 가능하기에 Alexa가 한국어를 말하도록 할 수 있다.

A.7. 멜론 실시간 차트 / 곡 검색 API

Melon은 국내 대형 온라인 음악서비스이다. SK 플래닛 개발자 센터에서는 Melon에 질의할 수 있는 API를 제공하여, 이를 통해 Melon의 여러 정보를 요청할 수 있다. 이 중에서 실시간 차트 정보 및 곡 검색 API를 사용할 것이다. 실시간 차트는 실시간으로 반

영되는 가요 순위이다. 이를 이용하면 어디에서나 한국 가요의 실시간 순위를 그대로 받아 볼 수 있다. 차트의 순위 범위는 파라미터 값을 통해 조정할 수 있다. 곡 검색 API는 파라미터에 search keyword를 추가해서 호출하면 그 keyword를 가지고 검색된 곡의 정보를 반환한다. 본 프로젝트에서는 특정 가수의 차트 순위 곡 이외에 어떤 곡이 있는지에 대한 정보를 제공하기 위해 사용하였다.

B. 프로젝트 개발환경

여러 명의 조원으로 구성된 프로젝트이니만큼 협업 도구가 필수적이다. 소스 버전 관리 도구로 git을 사용하였는데, 본 프로젝트의 경우 소스 파일 외에도 관리해야 할 텍스트 정보 파일들이 여럿 존재하여 이 역시 git을 통해 기록이 남도록 하였다. 원활한 커뮤니케이션을 위해서는 현재 전 세계적으로 널리 쓰이는 업무용 메신저인 slack을 사용했다. 또한 git과 slack의 연동을 통해 실시간으로 상대의 작업내역에 대한 알림을 받을 수 있도록 하였다.

Figure 3 개발 Platform & Tools

서버로는 AWS의 Lambda를 사용하였고, 개발 언어는 NodeJS(버전 4.3)이다. DB로는 AWS에서 제공하는 noSQL DB인 DynamoDB를 사용한다. 데이터 저장소로는 S3를, 추후 데이터 캐싱에는 ElastiCache 를 사용할 예정이다. Alexa는 개발 플랫폼인 Alexa Skill Set을 제공한다. 이를 통해 본 프로젝트에서 Alexa가 인식해야 하는 문자열 및 변수들을 설정할 수 있다. Alexa의 테스트는 문자열 입력 방식 및 웹 상에서의 음성 인식 방식이 모두 가능하다. 또한 네이버의 음성 합성 및 기계 번역 API를 사용한다. 이들의 개발은 주로 각자의 로컬 환경에서 이루어졌으나, AWS Console에서 테스트하는 경우도 많았다.

Amazon Web Service는 상용 서비스이다. Lambda의 경우 사용자가 서버에 접근할 수 없으나, 내부적으로는 서버 자원을 사용하므로 이를 누적하여 과금하고, DynamoDB나 S3

의 경우도 저장 공간과 이용 시간에 따라 요금이 청구된다. 본 프로젝트에서는 한국 AWS의 개발용 계정을 받는 것을 검토하였으나, 상용화 전에는 AWS Free tier 및 학생 지원 프로그램으로 충분하다고 판단되어 이를 이용하였다. 네이버 API의 경우도 보통 10,000글자/일에 대하여 무료로 제공되고 있다.

4. Goal/Problem & Requirement

A. 제공 언어의 한계 극복

Alexa가 지원하는 언어는 영어와 독일어이다. 따라서 한국어 인식은 불가능하며, 한국어 발음을 위하여 여러 방안을 강구하였다. 우선 한국어 발음을 알파벳으로 표현하여 Alexa가 그대로 읽게 하는 방법이 있다. 그러나 이 방법은 Alexa 응답의 발음이 매우 부자연스러우며 유저 입장에서 올바른 음성을 들을 수 없다. 따라서 이 문제점의 해결을 위해 음성 합성을 사용하기로 하였다. 네이버 등에서 제공하는 음성 합성 API는 한국어 문자열을 전달하면 매끄러운 발음으로 변환한 MP3 File을 반환한다. 이 방법으로 부자연스러움을 최소화하고 올바른 한국어 발음을 출력할 수 있다.

B. 기본 기능 구현

본 어플리케이션에서 제공할 기본적인 기능들을 구현한다. 첫째로, 한국어 학습 관련 기능이 있다. 유저는 영어로 번역 질의를 하여 그에 해당하는 한국어 문장을 들을 수 있어야 한다. 또한 간단한 한국어 퀴즈를 만들어 학습의 재미를 늘릴 수 있도록 한다. 둘째로, K-POP 정보 제공 기능이 있다. 유저는 K-POP 차트에 대한 질문을 할 수 있으며, 특정 곡의 가사나 아티스트 정보 등을 얻을 수 있다. K-POP 음악의 재생 역시 가능하다. 궁극적으로는 유저 정보에 따라 음악을 추천할 것이다.

자세한 스펙의 경우 6. Implementation Spec에서 다루기로 한다.

C. 추천 알고리즘 구현

해외의 K-POP 팬은 본인이 좋아하는, 혹은 해당 언어권에서 유명한 가수나 곡 이외의 다른 한국 음악에 대한 접근성이 떨어질 것으로 사료된다. 그런 점을 고려할 때 본인의 선택 외에 시스템 상에서 자동으로 추천해주는 기능이 있다면 사용자 입장에서 매우 유용할 것이라고 생각한다. 보다 취향에 맞는 곡을 추천해주기 위해 유저 기반의 데이터를 수집, 활용하여 추천하는 Collaborative Filtering을 적용하고자 한다. 예를 들면 현재 노래를 들은 사용자가 직전에 들은 노래, 현재 노래를 들은 사용자들이 많이 들었던 다른 노래, 나와 가장 많이 들은 곡이 겹치는 사용자의 플레이리스트 중 랜덤 추천 등이

가능하다. 또한 현재의 기분에 따라 미리 정해진 음악을 재생하는 방식의 추천을 할 수 있다. 이러한 추천 알고리즘을 극대화하고자 한다.

D. 정량적 달성 목표

D.1. 정확성(Accuracy)

Prototype 구현 단계에서 Alexa 테스트 결과, 엉뚱한 답을 내놓거나 아예 답을 하지 않는 경우가 많았다. 초기에는 영어 음성 인식의 문제라고 판단하여 본 프로젝트에서 개선할 여지가 없다고 보았으나, 리서치 결과 정확한 응답에 미치는 요소들이 몇 가지 있었다. 이 부분들을 개선하여 거의 대부분의 경우에 정확한 응답을 얻을 수 있도록 목표를 세웠다.

주요 이슈들과 개선 결과는 7. Solution 및 8. Results에서 서술할 것이다.

D.2. 응답 시간(Response time, Performance)

일반 유저 상대로 하는 실시간 어플리케이션인 만큼 응답 시간이 매우 중요한 요소이다. 본 프로젝트는 기능이 다양하고 그 중 일부는 여러 단계의 로직을 필요로 한다. 또한 여러 가지 솔루션들을 같이 사용하여 Performance에 영향을 미치는 요인들이 다양하다. 따라서 기능 구현과 동시에 늘 속도의 개선을 염두에 두어야 했다. Prototype 단계에서부터 최종 결과물까지 각 기능들의 속도를 최대한 증가시키려고 하였다.

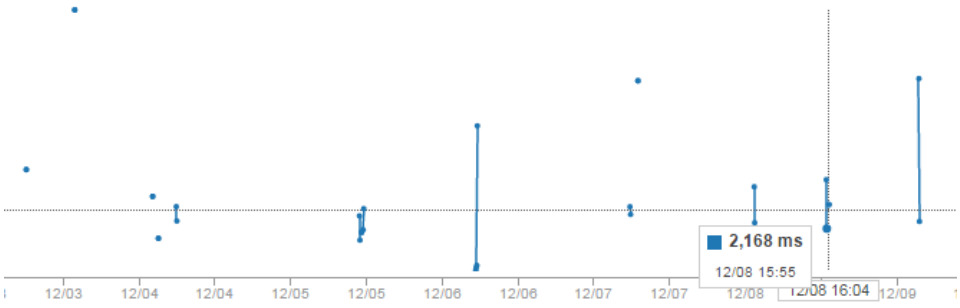

Figure 4 CloudWatch의 응답 시간 그래프

D.3. 처리량(Throughput, Scalability)

본 프로젝트에서 또 하나의 목표를 세운 것은 확장 가능성이다. 개발, 테스트 및 데모 시연에서는 유저가 많지 않지만, 유저가 기하급수적으로 늘어났을 때의 대응을 하나의 도전 과제로 삼았다. 수많은 Alexa에서 동시에 요청이 들어오더라도 속도 저하 없이 처리할 수 있도록 노력하였다. 다만 처리량의 경우 대부분 사용 솔루션의 선택 및 설계에 의존하기 때문에, 특별히 측정하여 비교하지는 않았다. 이론적으로 시뮬레이션을 하기로 했다.

5. Project Architecture

A. Architecture Diagram

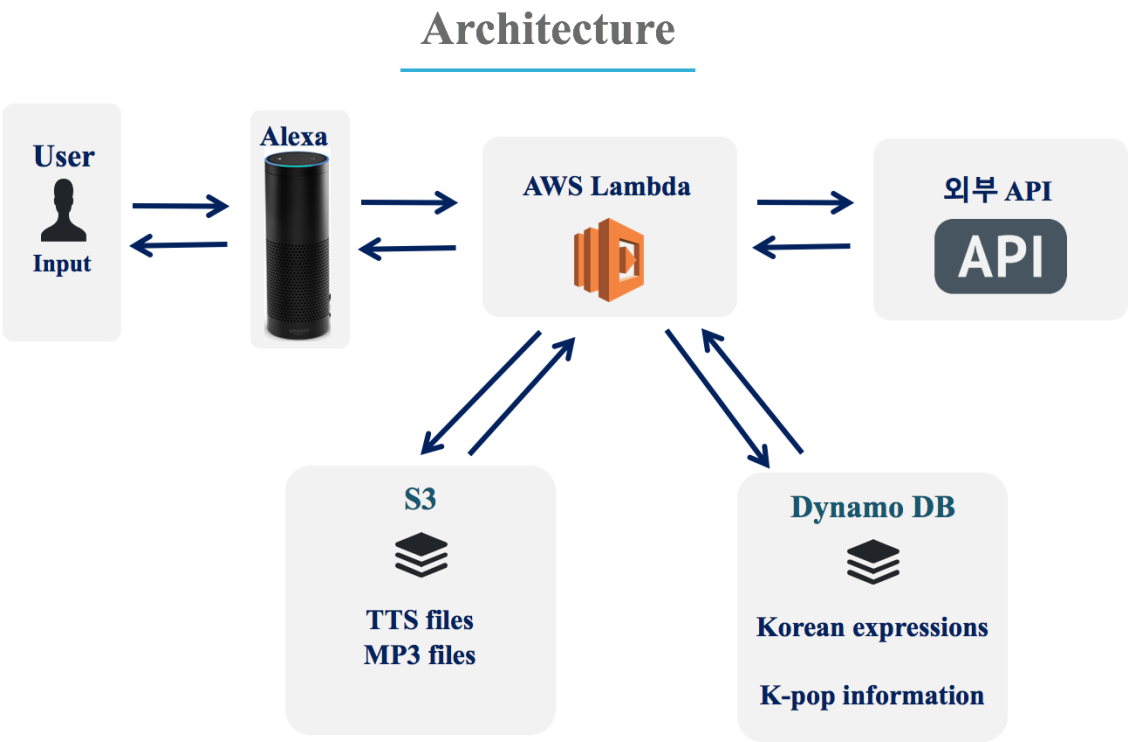

Figure 5 Architecture Diagram

B. Architecture Description

위의 그림은 본 프로젝트의 기본 구조를 보여준다.

사용자가 Alexa에 말을 하면, Alexa는 미리 지정된 Utterance에 따라 이를 해석한다. 해석된 문장을 분류한 뒤 그에 맞는 Parameter와 함께 서버 역할을 하는 AWS Lambda에 전달한다. Lambda 함수는 모든 로직을 처리하는 역할을 한다. 기능에 따라 세부 흐름은 상이하지만, 기본적으로 HTTP 통신, DB 접근, MP3 및 Speaking 문자열 전달 등의 역할을 한다. Lambda가 다른 Lambda 혹은 외부의 API를 호출할 수도 있다.

S3는 파일들을 저장하는 Storage이다. 본 프로젝트에서는 한국어 문장을 TTS를 이용하여 만든 MP3 파일, K-POP 음악 MP3 파일들을 저장한다. Alexa에 이를 URL 기반으로 전달하면 음성 파일을 재생할 수 있다. 다만 임의의 MP3 파일을 그대로 재생할 수는 없으며, Alexa에 맞는 포맷으로 변환해주어야 했다.

DynamoDB는 noSQL 형태의 데이터 저장소이다. 가장 최근에 요청된 차트 정보 혹은 곡 정보 등이 필요에 따라 저장된다. 또한 유저 상태를 저장하여 유용하게 쓰거나, 중복

된 API 호출을 방지한다. 이 때 API 호출을 하지 않음으로써 속도의 개선효과도 기대할 수 있다.

외부 API 또한 빠질 수 없는 요소이다. 본 프로젝트에서 제공되는 컨텐츠인 한국어 학습 컨텐츠에서 네이버의 기계 번역 API 및 음성합성 API가 사용된다. 그리고 K-POP 관련 정보는 Melon API를 통해 가져온다.

6. Implementation Spec

A. Input/Output Interface

입력과 출력은 모두 마이크와 스피커가 내장되어 있는 아마존 에코에 의해 이루어진다. 음성으로 입력이 들어오면 에코 내의 음성 인식 모듈이 Text로 변환하여 미리 정의된 Utterance를 통해 Parsing한다. 이 정보를 인자로 하여 Lambda API를 호출한다. 출력은 두 가지 형식이 가능한데, 일반 평문을 읽도록 할 수 있고, SSML 태그를 이용한 음성 파일 재생이 가능하다. 실제 음성의 출력은 아마존 에코 내의 TTS 엔진에 의해 동작한다.

B. Inter Module Communication Interface

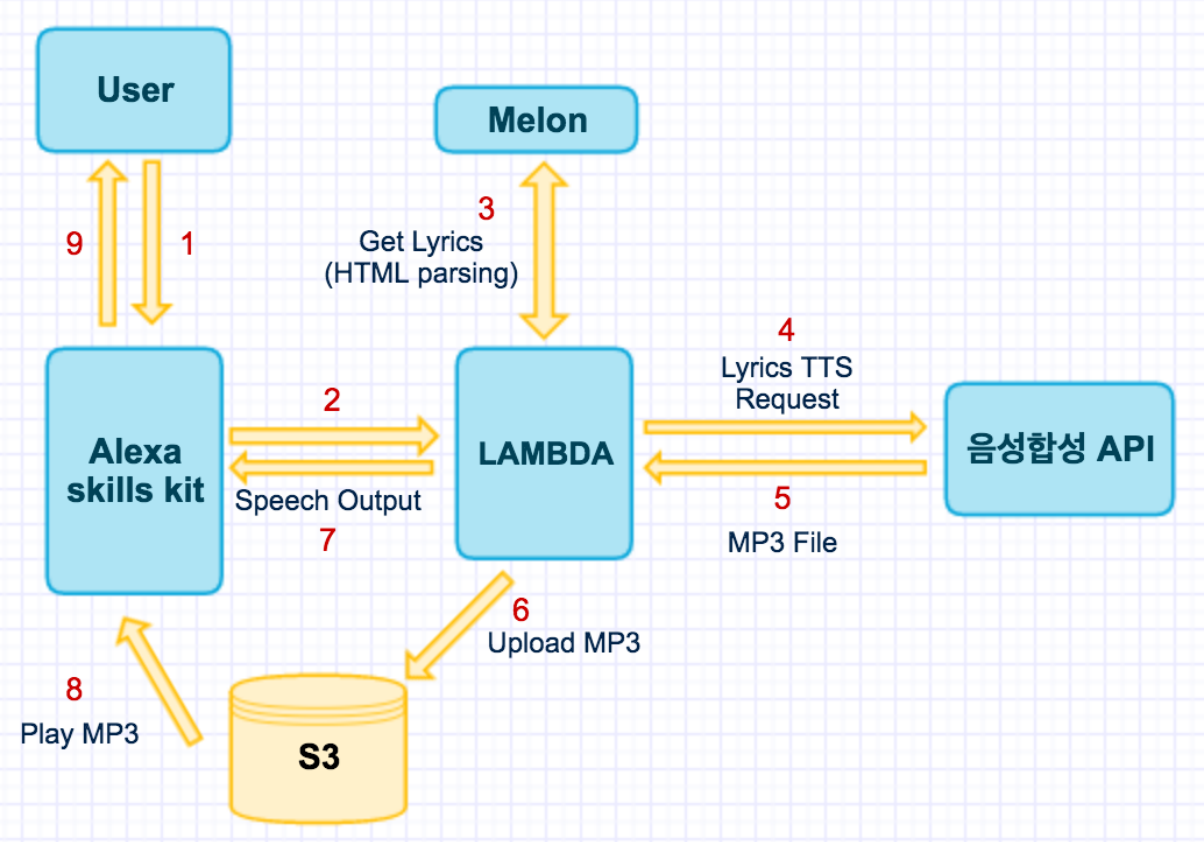

Figure 6 Module간 통신 Diagram

본 프로젝트를 구성하는 다양한 모듈은 기능에 따라 사용되기도 하고, 사용되지 않기도 한다. 또한 순서도 조금씩 다를 수 있는데, 대부분의 모듈을 사용하는 하나의 기능을 예로 들어 설명하고자 한다. 위의 그림은 가사를 재생하는 기능의 순서도를 나타낸다.

위의 그림에서 1번은 사용자가 Amazon Echo에 말을 하여 입력이 들어오는 부분이다. Alexa는 음성을 텍스트로 변환하여 텍스트를 람다 함수에 가사를 들려달라고 요청한다(순서 2). 람다는 서버의 역할을 하는 모듈로 대부분의 로직을 처리한다. 우선 람다는 해당 곡의 가사를 Melon으로부터 받아온다(순서 3). 이렇게 가져온 가사는 대부분이 한글로 이루어져 있을것이다. 하지만 alexa는 한국어를 지원하지 않기 때문에 한국어 텍스트를 음성으로 변환하는 작업이 필요하다. 이 역할은 NAVER의 음성합성 API가 담당한다. 람다는 음성합성 API에 요청을 보내고(순서 4) 응답으로 MP3 파일을 받는다(순서 5). 람다는 다시 응답으로 반환받은 MP3 파일을 S3에 업로드한다(순서 6). 람다는 S3에 파일이 올라간 것을 확인하고 alexa에게 그 파일을 읽어주라고 응답을 보낸다(순서 7). alexa는 건네 받은 MP3 파일 주소를 통해 그 파일을 로드하여(순서 8) 사용자에게 들려준다(순서 9).

C. Modules

NAME	DESCRIPTION
RequestHandler	AlexaSkillsKit과의 통신에서 모든 입력과 출력을 관리, 간단한 로직 즉시 처리
PlayLyrics	특정 곡 ID를 인자로 받아 해당 곡의 가사를 재생
EmotionBasedRecommendation	감정 상태를 인자로 받아 해당 감정에 따라 적절한 곡을 추천
Translate	영문 텍스트를 인자로 받아 네이버 기계 번역 API 혹은 구글 Translate API를 호출하여 번역된 한국어 텍스트를 반환
TTS	한국어 텍스트를 인자로 받아 네이버 음성 합성 API를 호출, 반환된 MP3 파일을 Alexa Playable Format으로 변환, 변환된 파일을 S3에 업로드 후 URL 리턴

TranslateWrapper	Translate 및 TTS 기능을 Wrapping Translate, TTS는 여러 기능에서 복합적으로 쓰이기 때문에 분리하여 구현
KPopChart	Melon API를 호출하여 1-10위 까지의 차트 정보를 가져와서 DB에 저장
SpeakKPopChart	차트 정보를 기반으로 유저에게 전달할 정보 문자열을 만들어 TTS Lambda를 호출, 해당 MP3의 URL을 반환
GetOtherSongs	가수 이름을 인자로 받아 Melon Search API 호출, 다른 곡 정보 반환
ArtistBasedRecommendation	가수 이름을 인자로 받아 해당 가수의 다른 곡 재생
Quiz	한국어 퀴즈 요청 및 각 퀴즈에 대하여 정답 체크
_QuizInserter	퀴즈 리스트를 삽입하는 One Time 로직 (Invisible to User)

Table 1 Module List

7. Solution

A. Implementation Details

A.1. Alexa Skills Kit

Alexa Skills Kit에서 미리 설정해 두어야 할 것은 크게 3종류로 나뉜다.

첫째는 Intent이다. Intent란 Alexa에 들어온 입력이 어떤 종류의 기능에 대한 입력인지 구별할 수 있도록 해주는 String 값이다. 예를 들어, 가사를 재생해 주는 기능은 PlayLyricsIntent, 감정 기반의 곡 추천 기능은 EmotionBasedRecommendationIntent등으

로 등록이 가능하다. 또 각 Intent에서 추후에 설명될 Slot이 쓰인다면 그 부분도 명시가 되어야 한다. 이러한 Intent설정을 json형태로 만들어 둔다.

둘째는 슬롯(Slot)이라는 개념이다. 슬롯은 여러 가지의 어휘 혹은 문장을 하나의 카테고리로 묶는 것이다. 예를 들어, 사용자가 본인의 감정상태를 말하고자 할 때 사용될 수 있는 어휘는 수백 혹은 수천가지에 달할 것이다. 이러한 수백가지의 감정들을 'Emotion'이라는 슬롯으로 등록해 둘 수 있다. 어휘 리스트는 개행 문자로 구분된 텍스트 형태로 저장된다.

마지막으로는 Utterances다. 이는 각 기능별로, 즉 Intent별로 입력으로 들어와 매칭될 수 있는 모든 문장을 저장해 둔다. 예를 들어, 현재 슬프다고 감정표현을 한다고 하면 'I am sad', 'I feel sad'등이 가능할 것이다. 이러한 문장들을 Intent에 매칭 시켜 저장해두는 설정이 utterances설정이다. 위의 예에서 sad대신에 들어올 수 있는 수많은 감정들은 {Emotion}과 같은 파싱 구분자로 모든 입력에 대해 일괄 처리가 가능하다.

A.2. RequestHandler Lambda

Alexa Skills Kit의 모든 입력은 하나의 람다가 받아서 해당 람다가 기능에 따라 다른 람다를 호출하는 형식으로 설계하였다. 이러한 람다를 'ReqeustHandler(이하 핸들러)'라고 명명하였는데 Alexa는 응답으로 평문 혹은 SSML태그, 두가지의 형식을 지원한다. 평문은 단순히 람다함수에서 텍스트를 리턴하면 그대로 읽어주므로 큰 문제는 없다. 본 프로젝트에서는 Naver 음성합성 API를 사용하여 생성된 MP3 파일 혹은 K-POP 관련 MP3 파일들을 재생해야 하는 경우가 꽤 있었다. 이를 위해서 SSML 태그를 사용해야만 한다. 이런 출력 형식은 모든 람다에서 쓰이지만 중복된 코드가 모든 람다에 있는 것은 비효율적이므로, 핸들러에서 각 Intent별로 다른 형식의 응답을 반환할 수 있도록 설계하였다.

A.3. Naver API Wrapping

A.3. (1) 기계 번역

네이버 기계 번역 API를 람다 함수에서 호출할 수 있도록 구현하였다. 네이버 비로그인 API의 경우 복잡한 인증 과정 없이 Client-ID, Client-Secret 상수값만 파라미터로 전달하면 동작하도록 되어 있다. Node.js의 HTTPS 모듈을 사용하여 Lambda가 Trigger될 때 그 인자값과 함께 기계 번역 API를 호출하고, 응답을 받아 콜백 함수에서 그대로 리턴한다. 인자와 리턴 모두 Plain Text로 큰 어려움은 없었다.

A.3. (2) 음성 합성

음성 합성의 경우도 호출 자체는 기계 번역과 같이 쉽게 구현되었다. 그러나 음성 합성 API는 응답값이 MP3 바이너리 파일로, 이를 처리하는 데 있어 여러 기술적 이슈가

발생하였다.

우선 Lambda에서 MP3 파일을 받기 때문에, 로컬 환경처럼 즉시 파일의 무결성을 확인할 수 없었다. 그래서 파일 처리 메커니즘이 모두 구현된 뒤에야 디버깅을 할 수 있었는데, 버그 발생 시 어느 부분의 문제인지 확신할 수 없는 상황이 많았다.

Lambda는 서버가 아니므로 State를 가질 수 없고, 파일도 영구적으로 저장할 수 없다. 따라서 위에서 서술한 S3에 Upload하여야만 했다. 초기에는 MP3 스트림을 즉시 S3으로 돌림으로써 시간과 자원을 절약하였으나, 추후 Alexa Research가 진행되면서 Alexa에서 MP3을 재생하기 위해서는 특별한 변환 과정이 필요함을 알게 되었다.

Lambda에서 유일하게 접근 가능한 디렉토리는 /tmp/이다. (물론 State를 가지지 않으며 한 번 호출된 후 원칙적으로 사라질 수 있다.) 이곳에 MP3 파일을 저장한 후 ffmpeg라는 미디어 처리 라이브러리를 활용하여 변환 과정을 거쳤다. Lambda에서도 ffmpeg 같은 External Library를 사용할 수 있는데, 패키지에 같이 묶어서 업로드한 후 특정 경로로 접근하는 방식이었다. 물론 그 라이브러리에서 참조하는 라이브러리들은 Static으로 함께 빌드해야 한다.

```
25 function uploadAndClean(tempName, fileName, callback) {
26     var s3 = new aws.S3();
27     var file = fs.createReadStream(DEFAULT_PATH + fileName);
28     var s3param = {
29         Bucket: 'koreantts',
30         Key: fileName,
31         Body: file,
32         ACL: 'public-read'
33     };
34     s3.upload(s3param, function(err, data) {
35         if (err) console.log('S3 upload error : ' + err, err.stack);
36         else console.log('S3 upload succeed : ' + data);
37
38         cp.exec('rm -rf ' + DEFAULT_PATH  + tempName + ' ' + DEFAULT_PATH + fileName,
39             function(error, stdout, stderr) {
40                 if(error)
41                     console.log('deleting temp error : ' + error);
42                 else
43                     console.log('deleting temp succeed');
44                 callback(null, 'https://s3.amazonaws.com/koreantts/' + fileName);
45             }
46         );
47     });
48 }
49
50 function convertMP3(tempName, fileName, callbackAfterUpload) {
51     cp.exec(
52         DEFAULT_PATH + 'ffmpeg -i ' + DEFAULT_PATH + tempName
53         + ' -analyzeduration 10000000 -probesize 100000000 '
54         + ' -ac 2 -codec:a libmp3lame -b:a 48k -ar 16000 -ss 00:00:00 -t 00:01:25 '
55         + DEFAULT_PATH + fileName,
56         function(error, stdout, stderr) {
57             if(error) {
58                 console.log('executing ffmpeg error : RETRY');
59                 convertMP3(tempName, fileName, callbackAfterUpload);
60             } else {
61                 console.log('executing ffmpeg');
62                 uploadAndClean(tempName, fileName, callbackAfterUpload);
63             }
64         }
65     );
<s.js   CWD: /Users/yeongjin/Dropbox/SNU/2016F/Project/kalexa   Line: 58   Column: 4   49%
```

Figure 7 FFMPEG의 실행

A.4. Melon API Calling

A.4. (1) 실시간 차트

멜론 실시간 차트 API를 람다 함수에서 호출할 수 있도록 구현하였다. SK플래닛 개발자 센터에서 제공받은 App key 로 인증이 가능하고 파라미터로 version(현재 1 고정값), page(받을 page 번호), count(받을 곡의 수) 를 전달하여 API를 호출하면 그에 따른 실시간 차트 정보를 응답받는다. 프로젝트에서 제공할 차트범위는 1위부터 10위까지(page = 1, count = 10)로 정했다. 응답받는 파라미터에는 곡에 대한 여러가지 정보가 모두 포함되어 있지만 그중에서 이후 기능 구현에 필요한 songId, songName, artistId, artistName을 담은 song object의 list로 정리하여 JSON 형태로 리턴하도록 했다. 또한 이번 프로젝트에서 K-pop 관련 기능을 구현할 때 실시간 차트를 이용하는 부분이 상대적으로 많이 발생한다. 따라서 실시간 차트는 실시간으로 순위가 반영이 되는 data지만 실제 현실에서 차트 순위의 변경이 실시간으로 계속 뒤바뀌지 않고 일정 기간은 순위정보가 유지된다는 것을 고려하여 매번 실시간 차트 API를 호출하지 않고 DB에 저장된 차트의 정보를 활용할 수 있도록 차트 정보를 리턴하기 전에 실시간 차트 정보를 DynamoDB에 업데이트하도록 구현하였다.

A.4. (2) 곡 검색

멜론 곡 검색 API를 호출하는 람다함수를 구현하였다. 실시간차트와 호출방법은 유사하며 다른점은 search key 파라미터가 추가되어야 한다는 것이다. search key의 값은 임의의 String 값이 들어갈 수 있다. 이후 목차 F에서 설명할 특정 가수의 다른 노래는 어떤 것들이 있는지에 대한 정보를 제공하는 기능을 구현하기 위해 해당 API를 사용한다. count는 5로 설정하여 검색된 목록에서 5개의 곡의 정보를 응답 받도록 구현하였다.

A.5. Chart Information

Figure 8 멜론 차트

실시간 1~10위 차트에 올라온 곡의 제목, 가수 이름을 Alexa가 응답 해주도록 한다. 곡의 제목과 가수의 이름은 대부분 한글로 되어 있으며 이를 Alexa가 응답해주기 위해서는 음성합성을 이용하여 곡 제목과 해당 가수의 이름을 mp3 파일로 저장하여 이를 재생하는 방법으로 진행되어야 한다.

곡 제목과 가수이름에는 부연 설명을 위해 특수문자가 포함되어 있거나 중복되는 정보가 있는 경우도 있다. 예를 들면 "돌아오지마 (Feat. 용준형 Of 비스트)"와 같은 곡 제목 또는 "헤이즈 (Heize)"와 같은 가수이름이 있다. 이런 경우 그대로 음성 합성을 하면 실제 곡 제목인 "돌아오지마" 뿐만 아니라 뒤의 내용 모두 음성 합성이 되어 이를 출력할 경우 의아한 곡 제목으로 들릴 수 있다. 후자의 경우 음성 합성을 하면 같은 내용이 두 번 반복되어 출력되므로 부자연스러워진다. 따라서 음성 합성을 하기 전에 곡 제목과 가수 이름 String에 대해 적절한 subsrting을 만드는 작업이 필요하다. 이에 대한 현재 구현 방법은 "("가 포함되어 있을 경우 그 전까지의 String을 받아들이도록 구현하였다. 이러면 중복되는 정보는 제거할 수 있다. 하지만 피처링을 한 가수 정보가 괄호 안에 있었다면 그 정보는 손실된다. 보통 어떤 곡에 여러 가수가 참여할 경우 가수들의 정보가 배열로 들어있다. 하지만 위에 예시와 같이 곡 제목에만 그 정보를 써 두고 가수정보에는 없는 경우도 있다. 이럴 경우 '피처링을 누가 했느냐에 대한 정보는 차트 정보 제공의 큰 틀에서 보았을 때 생략할 만한 정보이다'라는 가정을 하고 구현하였다.

곡 제목과 가수 이름을 음성합성 할 때 이미 S3에 저장된 정보가 있는지 판별하기

위해 mp3 파일 제목인 key값을 정한다. songId와 artistId를 활용하여 unique한 파일 이름을 만든다. 따라서 음성합성을 하기 전에 Id값으로 MP3 파일 이름을 도출하고 S3에 파일 존재여부를 확인한 뒤 존재 하지 않을 경우에만 음성합성 람다 함수를 호출하여 음성변환 후 S3에 새로운 mp3 파일을 업로드 한다. 이미 존재하는 파일이라면 음성 합성을 호출할 필요없이 바로 S3에 접근하여 MP3 파일을 가져온다.

처음에는 Alexa에게 '차트를 알려줘'와 같은 미리 설정해 놓았던 말을 하면 Alexa가 1위부터 10위까지 순서대로 곡 제목과 해당 곡의 가수이름을 순차적으로 응답하도록 구현하였다. 하지만 Alexa는 한 응답에 최대 5개의 MP3 파일만 재생시킬 수 있다는 한계가 있다. 따라서 곡 제목 10개, 가수이름 10개 총 20개의 MP3 파일을 재생해야 하는 차트 정보 제공 기능은 불가능하다. 이를 해결하기 위한 방법으로는 두가지가 있다. 첫번째는 곡 제목과 가수이름을 각각 20개의 mp3 파일로 만드는 것이 아니라 차트정보는 알려주는 문장 전체를 하나의 mp3 파일로 만드는 것이다. 이 방법을 사용하면 음성 출력해 줄 수 있는 정보의 양은 제약이 없다. 하지만 두 가지의 단점이 있다. 차트 정보를 제공하는 방식은 the first song is ~~ the artist is ~~ 이며 '~~' 이 부분이 음성 합성 된 한국어 노래제목과 가수이름이 들어 갈 자리이다. 이 것을 하나의 mp3 파일로 음성 합성을 할 경우, 영어로 된 부분이 영어 원어민 발음이 아닌 어색한 발음으로 합성 되는 것이다. 또 하나의 단점은 이렇게 만들어진 음성 파일은 caching 하기가 사실상 힘들다는 것이다. 차트 정보를 통째로 합성했기 때문에 차트 내 순서가 조금만 바뀌어도 다시 TTS 콜을 해야하며 하나하나의 음성 파일을 저장하여 같은 정보일 경우 DB에서 가져오도록 구현한다고 한다면 key 값이 차트 전체 정보이어야 하므로 상대적으로 비효율적이다.

따라서 곡제목과 가수이름을 각각 음성합성 하는 방향으로 정하였고 응답하는 종류를 세분화 하는 방식을 선택하였다. 사용자의 별다른 요청이 없는 단순 차트정보 질의에는 곡의 제목만 알려주고, 해당 가수의 이름을 요청하는 질의에서 해당 가수명을 알려주도록 카테고리를 나누었다. {fromSixth}라는 slot과 {Nth}라는 Slot을 추가하고 사용자가 요청한 문장에 두 slot이 없을 경우는 1~5위의 곡 제목만 출력하도록 하고 {fromSixth} slot이 있을 경우 6~10위의 곡 제목을 출력한다. {Nth}라는 slot이 있으면 해당 순위에 있는 곡 제목과 가수이름을 출력하도록 설계하였다.

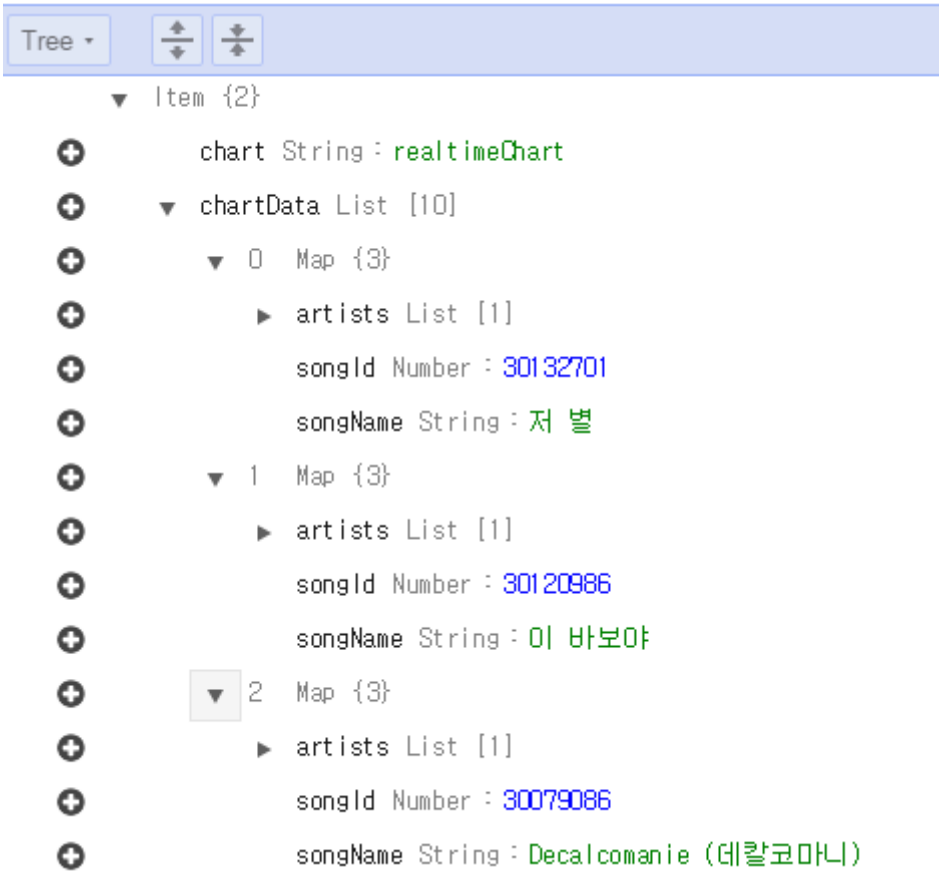

Figure 9 DB의 차트 데이터 예시

A.6. Other Songs Information

사용자가 '몇 번째 가수의 다른 곡 알려줘'와 같은 형태의 요청을 하면 Alexa는 해당 가수의 노래 최대 4개를 제공한다. 노래가 최대 4개인 이유는 이전 항목에서 설명한 Alexa의 재생가능한 최대 MP3 파일 수가 5개이기 때문이다. 곡 제목 4개와 나머지 1개는 해당 가수의 이름이다.

이를 구현하기 위해서는 곡 검색 API 호출 람다 함수를 사용한다. Intent에 {Nth} slot 옵션을 추가하여 사용자가 몇 번째를 원하는지에 대한 정보를 파라미터로 받고 DynamoDB에 저장되어 있는 차트 정보를 불러와서 해당 순위의 곡 정보를 가져온다. 곡 정보를 곡 검색 API 람다 함수로 넘겨주고 artistName을 search key로 하여 곡 검색 API 호출을 한다. 곡 검색 API호출 람다 함수에서 최대 5개의 곡을 가져오는 이유는 차트에 있는 곡도 검색 목록에 포함될 수 있기 때문에 넘겨받은 songId와의 비교를 통해서 이를 제외하고 4개 목록을 가져오기 위함이다. 이렇게 해당 가수의 곡 4개의 정보를 받으면 각 노래의 제목을 음성 합성한다. 이 때에도 마찬가지로 S3에 존재하지 않는 파일일 경우에만 업로드하도록 구현했다. 만일 해당 가수의 검색 결과가 없을 경우 Alexa는 '다른 곡이 없다'와 같은 응답을 하고 그렇지 않을 경우 가수이름과 4개의 곡 제목을 순차적으로 응답해준다.

A.7. Playing Music

차트 정보나 가수의 다른 곡을 alexa가 재생시켜주는 기능이다. '차트 몇번째 노래 재생해줘' 또는 '몇번째 순위 가수의 다른 곡 재생해줘'와 같은 요청을 하면 alexa는 그에 대한 곡을 재생한다. 이 기능에서는 {Nth} slot 과 {NotinChart} slot 옵션을 가진다. {Nth} slot 정보만 있으면 사용자가 몇번 째를 요청하는건지에 대한 정보를 받고 DynamoDB에 저장되 있는 차트 정보를 가져와서 해당 순위 songId를 이용해 곡을 재생한다. {Nth} ,{NotinChart} slot 정보가 모두 있는 경우 차트 정보에서 해당 순위 가수이름에 대해 곡 검색을 한 후 제일 검색된 곡중 제일 상위 곡의 songId를 받아 곡을 재생시킨다.

A.8. Emotion Based Recommendation

사용자가 현재 자신의 감정 상태를 입력했을 때 alexa가 감정 상태에 따라 지정된 곡들을 추천하는 기능이다. 약 200여 가지의 감정을 지원하고, 실제 처리부분에서는 4가지(희, 로, 애, 락)로 구분한다. 선곡은 미리 감정 별로 선별해서 만들어 둔 리스트에서 랜덤 재생된다. 또한, 해당 추천곡이 사용자의 마음에 들어, 가사 재생 기능을 요청할 수 있는 상황을 대비하여 마지막으로 재생된 곡의 정보를 DB에 저장해둔다. 만약 사용자가 '이 곡의 가사를 알려줘'와 같은 말을 Alexa에게 한다면 가사를 재생한다.

A.9. Telling Lyrics

특정 곡을 골라 가사를 재생하는 기능이다. 유저별로 플레이리스트를 저장하여 직전 재생 곡을 선택하거나, 차트 상에서 몇 번째 곡을 지정하는 방식 등으로 특정 곡을 선정할 수 있다. 멜론 API에서 가사를 알려주는 기능은 제공하지 않기 때문에 곡의 고유 ID인 songId가 입력으로 들어오면 해당 곡의 가사 정보를 멜론 홈페이지에서 크롤링하는 방식으로 가사를 가져온다. 이렇게 얻어진 가사를 음성합성 API를 호출하여 MP3 파일로 만들어 내고 Alexa에게는 해당 파일을 재생한다. 다만 이미 가사 MP3 파일이 있는데 음성합성 API를 호출하게 되면 불필요하게 중복된 파일이 S3에 쌓이게 되고, 속도 또한 느릴 수 밖에 없다. 이를 방지하기 위해 해당 곡 가사 재생 최초 요청시에 songId와 url 정보를 dynamoDB에 저장한다. 추후에 같은 요청이 들어오게 되면 DB에 저장된 정보가 있는지 확인하고 없을 경우에만 음성 합성 API를 호출한다.

A.10. User Based Recommendation

본 프로젝트에서는 기본적으로 곡이 재생되는 모든 경우에 대하여 로그를 남긴다. 즉, 유저 별로 플레이리스트를 저장하고 있다. 플레이리스트는 곡의 고유 ID와 해당 곡의 아티스트의 고유 ID를 함께 저장한다. 사용자 DB기반 추천 기능은 이러한 재생 내역 리스트를 기반으로 유저가 좋아할 만한 곡을 임의로 추천해주는 것을 목표로 한다.

우선 플레이리스트를 선택하는 데 있어 2가지 옵션이 있고, 임의로 선택된다. 첫째는 가장 많이 들은 곡이 같은 사용자의 플레이리스트고, 둘째는 플레이리스트 내에서 가장 큰 비중을 차지하는 아티스트가 같은 플레이리스트이다. 두 가지 경우 모두 추천 대상 플레이리스트가 없을 경우 임의 사용자의 플레이리스트를 선택한다.

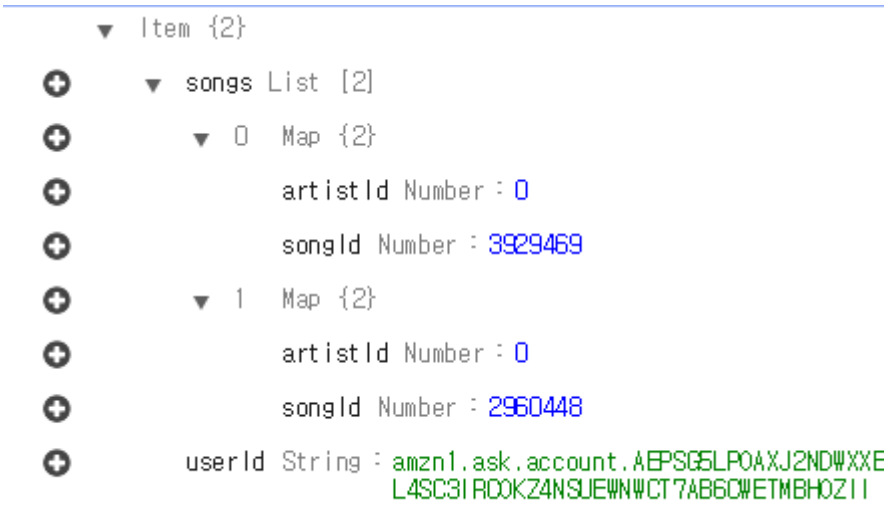

Figure 10 플레이리스트의 예시

이렇게 일단 현 사용자의 취향과 가장 비슷한 플레이리스트를 고르고 나면, 그 중에서 최종적으로 곡을 선택한다. 곡을 선택하는 방식은 총 4가지로 그 중 한가지 방식이 임의로 선택된다. 4가지 방식은 플레이리스트 내에서 가장 최근에 재생된 곡, 플레이리스트 내에서 가장 많이 재생된 곡, 가장 선호되는 아티스트의 임의 곡, 임의의 곡으로 구성된다.

따라서 총 8가지의 조합이 가능한데, 이 중에서 공통적으로 가장 많이 들은 곡이 같은 플레이리스트에서 가장 많이 재생된 곡을 택하면 늘 같은 곡이 나오기 때문에, 이 경우에는 추천 플레이리스트에서 2번째로 많이 재생된 곡이 선택되도록 별도의 처리를 하였다.

A.11. Translation

한국어 번역은 Alexa에서 영어 표현의 한국어 번역을 질의할 수 있는 기능으로, 현재 부분적으로 완료되었다. Alexa에게 어떤 표현의 한국어 번역이 무엇인지 물어보면, 내부적으로는 해당하는 람다 함수가 호출된다. Lambda에서는 네이버 기계 번역을 통해 문자열을 한국어로 번역하고, 이를 다시 음성 합성으로 한국어가 재생되는 MP3을 생성한다. Alexa에 이 MP3 파일을 전달하여 마침내 유저는 한국어 번역 결과를 들을 수 있다.

기계 번역 및 음성 합성 API 부분을 모듈화하여 따로 Lambda 함수로 만들어 놓았기에, 이를 가져다 쓰는 것은 간단한 일이었다. 기본 동작을 구현한 뒤 Alexa가 알아들어야 하는 질의들을 추가했다. Alexa Interaction Model에서 질의에 해당하는 Intent와, 그 내부

번역 대상에 해당하는 Slot들을 채워주었다.

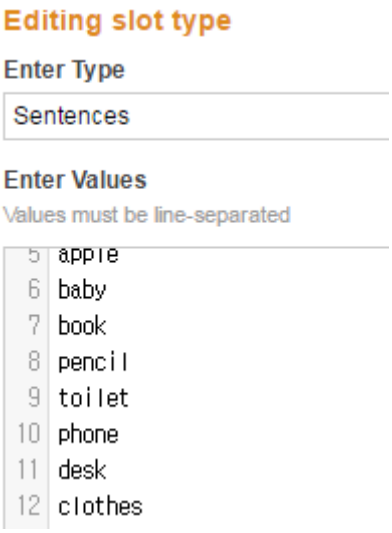

Figure 11 Slot의 예시

그러나 현재 Alexa는, 미리 정의된 문자들만 변수로 받을 수 있다. 즉 한국어 번역을 위해서는, 사용자가 물어볼 만한 단어와 문장들을 '모두' 정리해서 저장해 놓고 있어야 하는 것이다. 심지어 단어들을 저장한 후 이에 기반한 문장들을 받는 형식도 지원하지 않는데, 이론적으로 이 서비스를 상용화하기 위해서는 천문학적인 리스트가 필요할 것이다. 이 부분은 본 프로젝트의 난점으로, Future Plan에서 아마존 측에 제안해볼 수 있는 내용이다.

지금은 동작 테스트를 위한 기초 영어 단어 1000여 개로 이루어져 있다.

A.12. Korean Quiz

한국어 퀴즈의 경우 유저가 Alexa에게 퀴즈를 달라고 질의하는 것부터 시작한다. Alexa는 퀴즈 목록에서 영어 문장 하나와 한국어 음성 MP3 URL을 랜덤하게 추출한다. 이 때 랜덤하게 정답을 O/X로 설정하여, X인 경우 영어에 해당하는 한국어가 아닌 다른 문장을 내보낸다. 이렇게 Alexa에 두 가지 Input Source를 전달한 후 유저 응답을 기다린다.

유저가 답을 말하면 Alexa는 그 답을 채점한 뒤, 남은 퀴즈 수가 있는 경우 퀴즈를 더 출제하도록 한다. 이를 위해서는 각 유저마다 데이터들을 저장해둘 필요가 있다. 유저 데이터는 DynamoDB의 quiz_status Table에 저장된다. 이 때 각 유저를 구분하는 방식은, User Based Recommendation에서 사용한 것처럼 Alexa Session의 userId 필드를 사용한다.

A.13. Others

기능 구현 시 써드 파티 라이브러리를 사용해야 할 경우가 많았다. Lambda가 아마존에서 제공하는 기본 패키지 외의 패키지를 사용하고자 하는 경우에는 소스파일과 해당

패키지를 같이 압축해서 Lambda를 생성해야 한다. 이와 같은 상황을 고려하여 하나의 람다별로 하나의 폴더를 갖는 구조를 갖게 하였다. 이 과정에서 생성되는 패키지나, 압축 파일은 공유될 필요가 없으므로 gitignore에 추가하여 버전 관리가 되지 않도록 하였다. 이러한 방식으로 사용되는 라이브러리들은 ffmpeg과 npm(NodeJS)의 request, async 등이 있다.

B. Implementation Issues

B.1. Accuracy

Alexa의 응답 실패에 영향을 미치는 요소는 다음과 같다.

1. 음성 인식 자체의 실패
2. Utterance의 충돌
3. 타임아웃
    A. API Latency
    B. 복잡하고 비효율적인 로직
    C. ffmpeg 용량

우선 영어 발음에 의하여 음성 인식이 실패하는 경우가 있는데, 이는 아마존 음성 인식 모듈 자체의 문제이기 때문에 어플리케이션 개발자 입장에서는 개선할 수 있는 여지가 없다. 다만 Web에서의 테스트인 Echosim에 비하여 실제 디바이스는 훨씬 더 좋은 인식률을 보여주었다.

Utterance는 어플리케이션에서 Alexa가 인식하도록 미리 정의한 문자열들인데, 비슷한 발음의 문장들이 많은 경우 충돌이 일어났다. 특정 문장으로 확실하게 신뢰할 수 없는 경우에는 한 쪽을 선택하지 않고 아예 인식을 하지 못하는 경우도 발생하였다. 따라서 본 프로젝트에서는 가능한 한 비슷한 발음의 문장들을 없애도록 하였다.

Alexa의 개발 키트에서는 실제 말을 하지 않고 Text를 입력하여 가상 Input으로 주어서 테스트하는 기능이 있다. 특히 실제 디바이스가 없을 때는 이 기능을 통하여 많은 테스트와 개발을 진행하였다. 그러나 실제 디바이스에서 말할 때는 타임아웃으로 인하여 응답을 못하는 경우가 발생하였다. 아무래도 End-User와 직접 닿는 서비스인 만큼 짧은 타임아웃을 설정해둔 것이라고 추측된다. 따라서 오래 걸리는 로직들을 개선한 후에는 정확도가 현저하게 좋아졌다. 속도 개선은 B.2. Performance에서 자세히 살펴볼 것이다.

정확도 개선 수치는 8.Results에서 확인할 수 있다.

B.2. Performance

본 프로젝트에서 가장 큰 주안점을 둔 목표는 Performance의 개선이었다. 이는 유저 경험(User Experience)에 직결되며, Alexa에서는 특히 응답 구성에 대한 타임아웃이 존재

하기 때문에 더 중요한 이슈가 되었다. 이 타임아웃은 3초였는데, 서비스 특성 상 3초를 초과하는 로직이 많았기 때문이다.

속도에 대한 문제점과 개선 방식을 정리하면 다음과 같다.

PROBLEM	DESCRIPTION	SOLUTION
Latency	AWS는 전 세계에 데이터 센터를 보유하고 있으며, 각각을 Region이라고 한다. 그런데 Alexa의 경우 미국 내 Region의 Lambda에만 접근이 가능했다. 따라서 일단 미국과의 통신 Latency가 발생하고, 본 프로젝트에서는 네이버/멜론 API를 사용하기에 그 문제는 심화되었다.	API 중 Google로 대체할 수 있는 번역 API는 Google로 바꾸었다. Google은 미국에 데이터 센터가 있으므로 한 번의 긴 Latency를 줄일 수 있었다. 또한 데이터들을 캐싱함으로써 API 호출을 최소화하였다.
Lambda Initialization	Lambda는 서버 구현체 없는 동작 기능을 제공한다. 이는 대부분의 경우 장점이지만, Lambda가 실행될 때마다 코드와 관련 라이브러리가 업로드되어야 한다는 단점이 있다. 특히 ffmpeg 라이브러리의 경우 static build가 46MB에 이르러서 큰 병목이 되었다.	우선 npm 라이브러리들의 사용을 최소화하고, 압축함으로써 업로딩 시간을 줄였다. ffmpeg의 경우 거의 대부분의 기능에서 쓰이기 때문에 결정적인 영향을 미쳤다. 그러나 우리는 ffmpeg 기능 중 audio, 그 중에서도 mp3 관련 코덱만 사용하기 때문에, ffmpeg 빌드 옵션을 조정하여 용량을 줄일 수 있었다. 이 때 Lambda는 Amazon Linux에서 돌아가기 때문에 그에 맞춰서 컴파일하였고, 같이 필요한 라이브러리들을 모두 .a 형태로 Static Link하였다. 결과적으로 용

		량을 2MB까지 줄일 수 있었다.
Redundant Logic	같은 동작을 반복하거나, 비슷한 API 호출을 반복하는 코드들이 있었다.	Code Refactoring 단계에서 코드를 최대한 효율적으로 수정하였다. 또한 API 호출 횟수를 줄일 수 있는 경우들을 찾아 간소화하였다.
Caching	똑같은 요청 처리를 위해 매번 똑같은 로직을 실행해야 했다.	MP3 파일의 경우 본 프로젝트에서 있어서 절대적으로 필요한데, 텍스트 통신보다 훨씬 더 큰 용량을 차지한다. 따라서 가능한 한 같은 요청에 대한 MP3 파일을 캐싱하여 재사용할 수 있도록 하였다. 이는 다른 유저가 요청했을 때에도 공용으로 사용할 수 있다.
HDD Based Database	현대 DBMS는 Hard Disk를 기본적인 Storage로 사용한다. 물론 메모리에 캐시하는 경우도 많지만 일반적으로 요청 시 HDD에 접근하는 시간이 소요된다.	몇 가지 기능들에 In-memory Cache인 Redis를 적용하였다. Redis는 메모리에 저장되는 데이터 캐시로 빠른 속도를 자랑한다. AWS에서는 ElastiCache라는 솔루션을 제공하여 이를 사용하였다.

Performance 개선 수치는 8.Results에서 확인할 수 있다.

B.3. Scalability

본 프로젝트에서는 초기 디자인 시부터 확장 가능성을 염두에 두었다. 유저가 기하급수적으로 증가했을 때의 상황을 가정하여 Architecture를 설계하였다.

우선 Lambda를 사용함으로써 메인 로직이 동시다발적으로 실행되는 데 문제가 없도록 하였다. 일반적인 웹 서버의 경우 요청이 많아지면 컴퓨팅 자원을 늘려야 하는 상황이 발생한다. 그러나 AWS Lambda의 경우 매 요청마다 새로운 Lambda Instance가 만들어지기 때문에, 처리량이 많아지는 것을 신경 쓸 필요가 없다. 또한 과금 역시 사용한 양만 결제하면 된다.

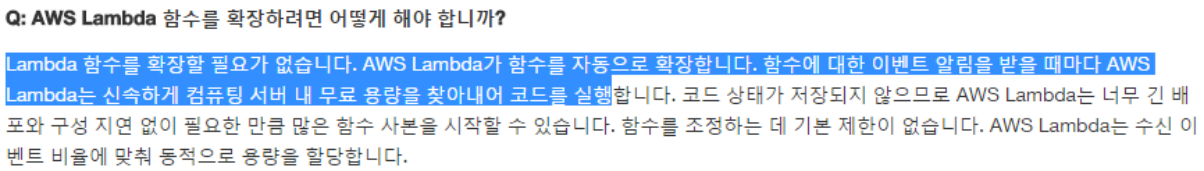

Figure 12 Lambda의 확장성

그리고 유저가 많아졌을 때의 효율성을 위하여 공유할 수 있는 데이터들은 모두 공유하도록 구현하였다. 이를테면 같은 문장에 대한 번역 MP3 파일은, S3에 저장한 뒤 다른 유저가 요청하더라도 그 파일을 사용할 수 있다. K-POP Chart 정보 역시 한 번만 얻어오면 변경되기 전까지는 캐싱된 데이터를 사용할 수 있다. 이와 같은 방법으로 Scalability를 확보하였다.

B.4. Lambda에서의 ffmpeg 비정상 동작

Alexa에서의 MP3 파일 재생에 ffmpeg를 통한 변환이 필요하다는 것은 앞서 살펴보았다. 그런데 팀원들의 Mac/Linux 환경에서는 잘 변환되는 MP3 파일이 Lambda에서는 간헐적으로 변환 실패하는 현상이 나타났다. 서버에 직접 접근할 수 있었으면 문제 해결이 손쉬웠을텐데, Lambda의 로그를 통해서만 간접적으로 상황을 추측할 수 있어 굉장히 오랜 시간이 소요되었다. 이 부분은 Lambda 사용의 단점이라고 할 수 있겠다. 결과적으로 드러난 문제는 크게 두 가지였는데, Output file이 이미 존재할 경우 무한정 Hang이 걸리는 현상 및 정상 파일 변환에서 ffmpeg crash가 발생하고, Lambda 실행도 멈추는 현상이었다.

두 가지 모두 로컬 환경에서는 전혀 발생하지 않는 이슈이고, ffmpeg 역시 다양한 미디어 플레이어에서 사용되는 유서 깊은 라이브러리라 예상치 못한 상황이었다. 그러나 당장 Amazon Linux에서 디버깅을 진행할 수 없었기에 이 부분은 우회하여 해결한 뒤 Future Plan으로 남기고자 한다. Output file이 이미 존재하는 경우 지워주었고, ffmpeg이 crash가 난 경우에는 반복해서 실행하도록 하였다. 결과적으로 거의 모든 경우에 1-2번의 변환 과정 안에 해결되었다.

8. Results

A. Experiments

본 프로젝트의 경우 유저가 질의할 수 있는 문장이 많으며, 임의의 Slot과 결합될 수 있기 때문에 엄청난 수의 질의 문장이 존재한다. 따라서 예시로 그 중 일부만 테스트한 결과를 싣는다. 테스트는 Text 입력을 통해 할 수 있고, Echosim에서 Web 상에서 Labtop 등의 마이크로 음성을 전달하여 할 수 있고, 실 디바이스로 할 수 있다.

A.1. Text 전달

특정 질의를 날렸을 때, 왼쪽 텍스트박스는 Alexa가 그 문장을 파싱하여 Lambda에게 전달하는 요청이고, 오른쪽 텍스트박스는 Lambda의 결과로, Alexa가 말할 문장이 된다. 이 때 SSML은 MP3 파일 등 Plain text가 아닌 데이터를 전달할 때 사용한다.

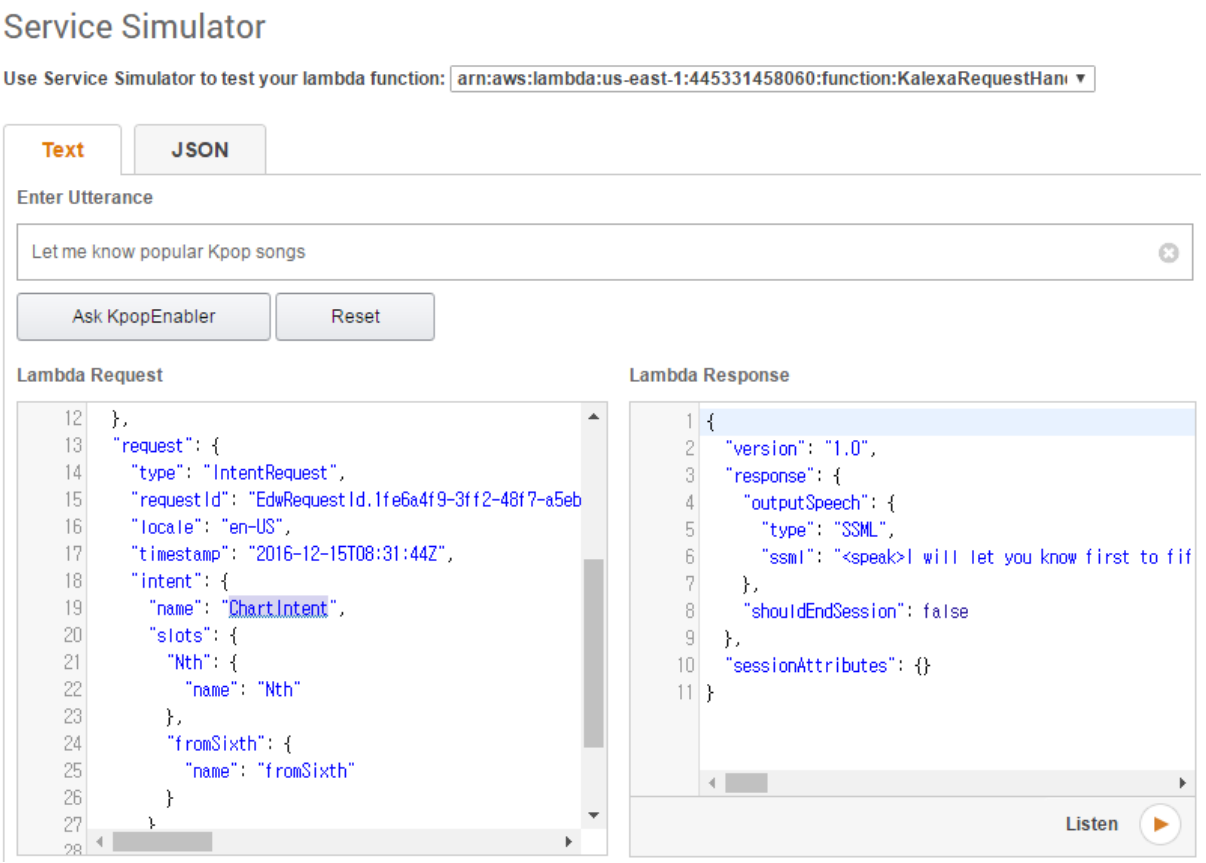

Figure 13 K-POP 차트 정보 테스트

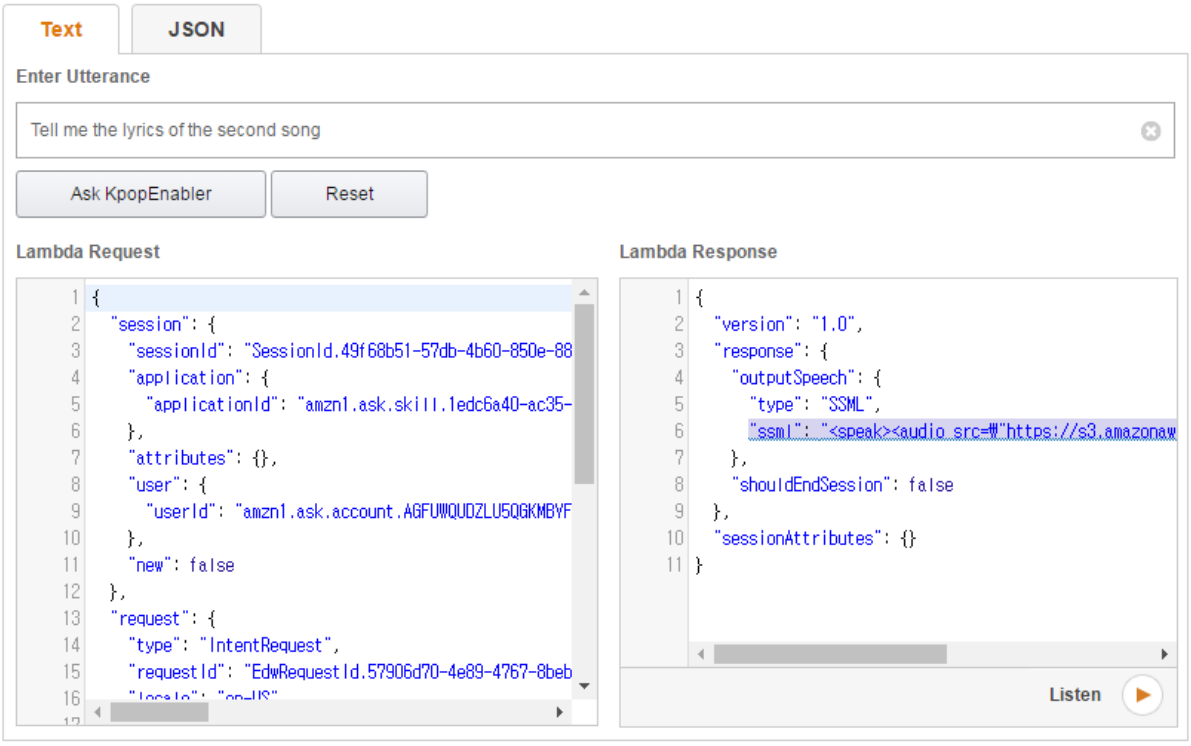

Figure 14 가사 질의 테스트

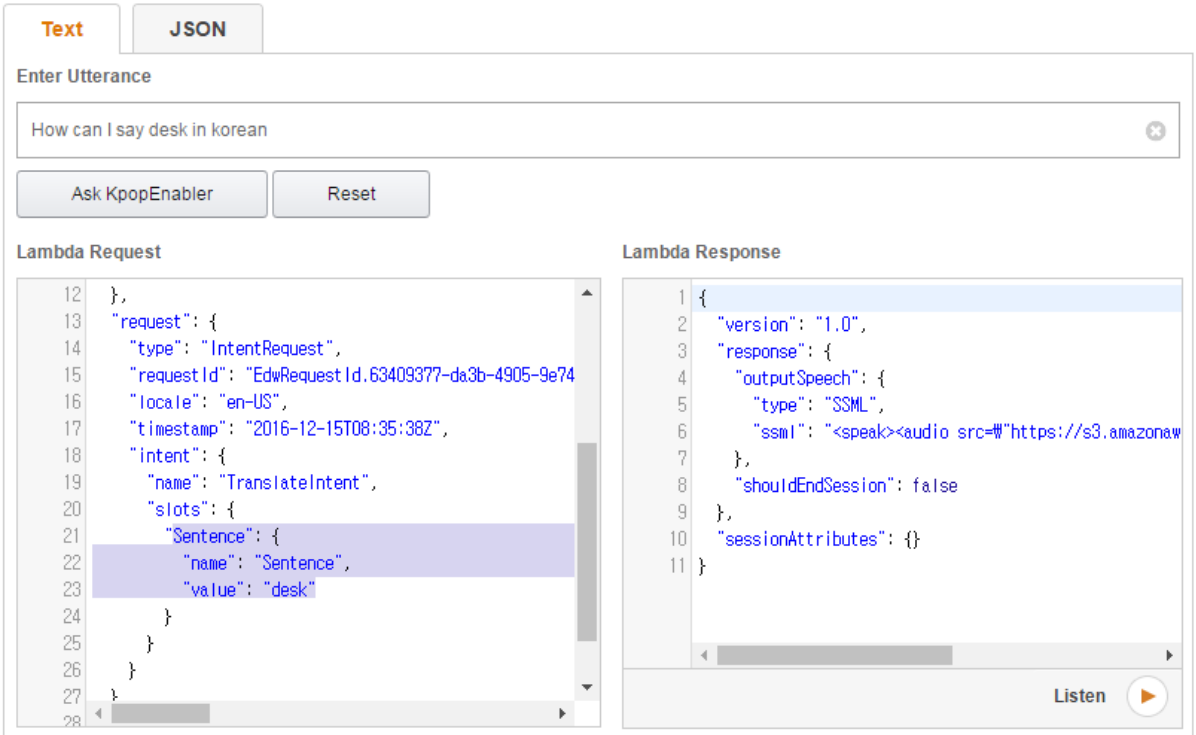

Figure 15 번역 질의 테스트

A.2. 실 디바이스 음성 질의

(실제 음성 응답을 Text로 옮김, 2016년 12월 기준)

TYPE	QUESTION	ANSWER
번역	Translate 'What is your name?' into Korean	당신의 이름은 무엇입니까?
퀴즈	Give me two quizzes. The answer is true.	Quiz start. Was this intentional? 이것이 고의적이었습니까? Correct. Next quiz. …
K-POP 차트 정보	Let me know the chart.	I will let you know first to fifth songs of K-POP chart. First song is …
가사 정보	Tell me the lyrics of first song.	… 일단 시작부터 제일 센 걸로 부탁해 … (가사 내용)
아티스트 정보	Let me know the other songs of first artist.	I recommend some songs of '빅뱅' …
음악 재생	Play the first song.	(1번 곡 재생)
감정 기반 추천	I feel sad.	(슬픈 곡 재생)
유저 데이터 기반 추천	Recommend me any song.	(추천된 곡 재생)
어플리케이션 시작 및 종료	Open project kei. We're done.	Welcome to the project kei. Ask me. Bye.

Table 2 실제 디바이스 음성 테스트

B. Result Analysis and Discussion

B.1. 기능 구현

Experiment에서 살펴보았듯이, 본래 계획했던 스펙대로 여러 기능들이 완성되었다. 유저들은 Project Kalexa의 여러 기능을 문제 없이 사용할 수 있다.

B.2. 정확도 개선

정확도의 측정은 특정 질의에 대하여 여러 번 반복하여 몇 번이나 정확한 응답을 주는지로 하였다.

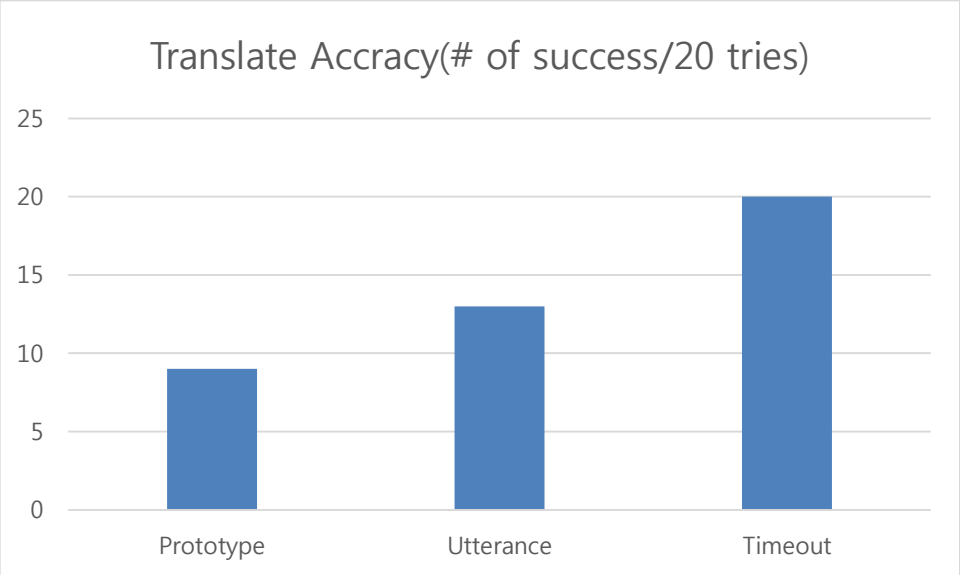

Figure 16 개선 작업에 따른 번역 응답 정확도 증가

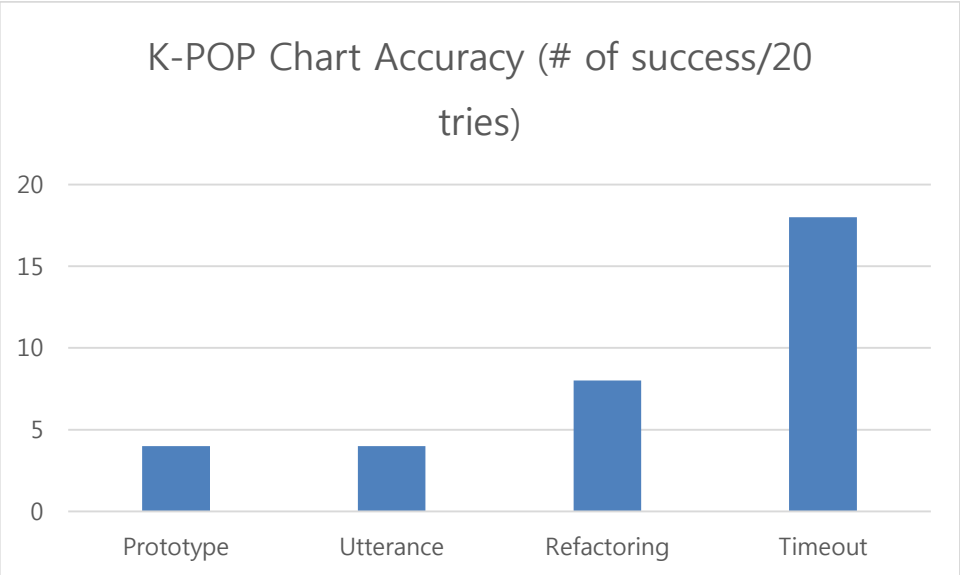

Figure 17 개선 작업에 따른 K-POP 차트 응답 정확도 증가

위 그래프에서 각각의 개선에 대해 정확도 변화를 알 수 있다. 우선 번역의 경우 Translate 명령 Prefix + 번역할 문자열 + Translate 명령 Postfix의 형태이므로, 번역할 문자열에 따라 인식 혼란이 생기는 경우가 빈번하였다. 따라서 Utterance가 충돌하는 경우를 해결하자 일정 부분 좋아졌다. 그 후 7.B.2 Performance 부분과 같이 Latency를 줄이고 ffmpeg 업로드 시간을 개선하여 타임아웃을 없애자, 모든 질의에 대하여 정확한 응답을 줌을 확인할 수 있었다.

K-POP 차트는 초기에 받아와야 하는 데이터가 상당히 많다. 차트 정보를 받아서 파싱한 뒤 아티스트명 등에 대해서는 각각 TTS를 통해 한국어 MP3 파일을 만들어야 한다. 이 경우 Utterance 충돌의 문제는 아예 없었던 것으로 추측된다. 해당 과정을 간소화하고 최대한 API 호출을 줄였을 때 상당 부분 개선되었고, 위의 TTS 시간이 줄어들자 대부분 쿼리에서 성공하는 모습을 볼 수 있었다. 그러나 차트가 많은 부분 바뀌는 경우 어쩔 수 없이 타임아웃으로 응답이 실패하였다.

B.3. 속도 개선

속도의 측정은 Lambda 함수의 로그를 통해 Elapsed Time을 추출하는 방법을 사용하였다. 이 때 각각에 대해서 10회 이상 실행하여 평균값을 추출하였다.

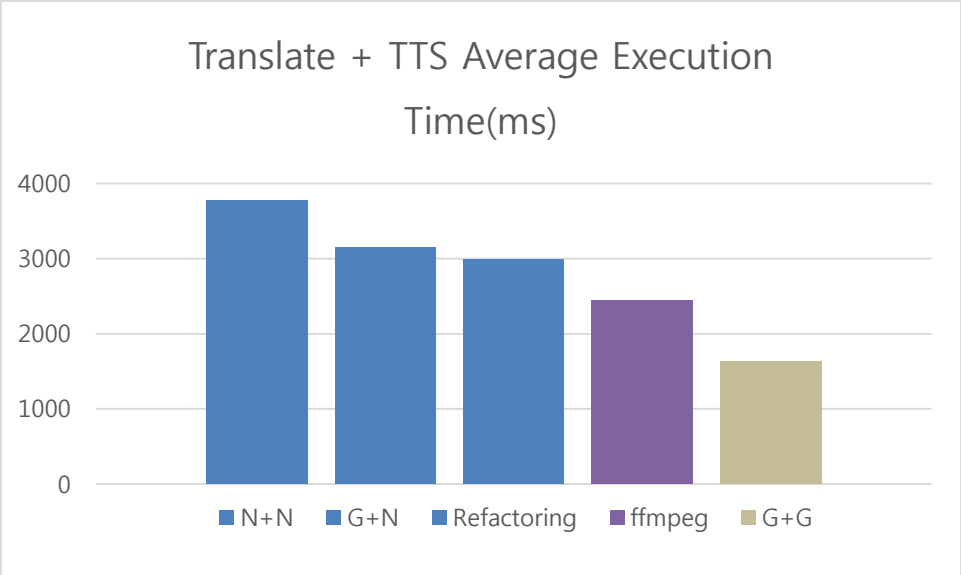

Figure 18 Translate + TTS 평균 실행 시간의 개선

Translate + TTS는 번역 기능에서 직접적으로 쓰이며, 각각의 기능은 다른 기능에서도 자주 쓰이는 기능이다. 우선 네이버 번역 API를 구글 API로 바꾸자, Latency의 감소(미국-한국의 통신이 한 번 줄어듦)로 실행 시간이 줄어들었다. 그 후 소스의 개선으로 약간의 실행 시간 감소가 있었다. 결정적인 부분은 ffmpeg의 최적화로, Amazon Linux에 맞게 우리가 필요한 기능만 선별적으로 빌드한 후 업로딩 시간이 짧아져서 많이 개선되었다. 그

래프 마지막은 음성 합성 API도 구글로 사용해본 것인데, 구글은 음성 합성 API를 공식적으로 공개하고 있지 않다. 따라서 내부 API로 테스트만 하였을 뿐 실제 사용은 불가능하다.

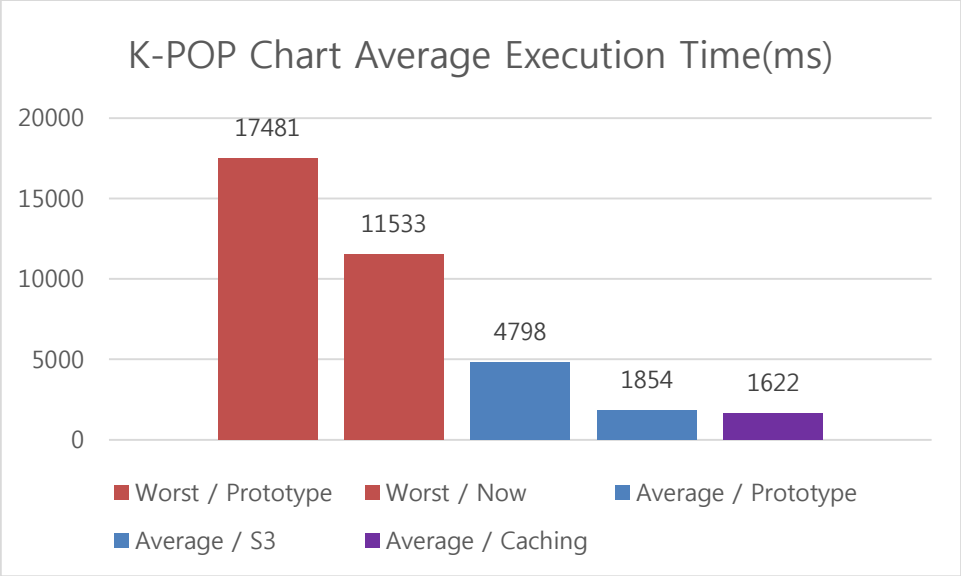

Figure 19 K-POP 차트 조회 응답 시간

K-POP 차트 정보 기능은 이론적으로 본 프로젝트에서 가장 오랜 시간이 소요되는 기능이다. 여러 가지 데이터를 가져온 뒤 그 데이터 각각을 TTS로 변환해야 한다. 이 때 데이터를 묶어서 TTS를 한 번만 호출하는 경우 음성 질이 하락해서, 여러 번에 걸쳐 호출할 수밖에 없었다.

Worst case란 하나의 데이터도 없는 상태에서 모든 차트 정보를 가져올 때를 말한다. 이 부분은 많은 개선을 거쳐도 10초 이상으로 도저히 감당할 수 없는 시간이 나왔다. 따라서 본 프로젝트에서는 초기부터 저장된 데이터를 사용하는 방법을 강구하였다. Average case란 차트 데이터가 이미 있는 때를 말한다. 이 경우에도 초반에는 5초에 가까운 시간이 걸렸으나, 모든 TTS MP3 파일을 S3에 넣자 현저하게 빨라졌다. 여기에 캐싱 기법을 더하여 현재는 매우 짧은 시간 내에 응답을 줄 수 있다.

B.4. 확장 가능성 측면

Scalability의 경우 따로 테스트할 수는 없었다. 다만 위에서 서술한 구조를 볼 때, 유저가 늘어날 때 문제가 될 부분은 없을 것이라고 판단된다. API 및 컴퓨팅 자원 사용료는 증가할 것이나, 응답 속도가 느려지거나 서비스가 실패하는 경우는 없을 것이다.

C. Limit

C.1. 추후 Alexa 자체의 한국어 지원

Alexa가 업데이트되어 한국어까지 지원하는 경우, 본 프로젝트의 의의가 상당 부분 사라진다. 한국어를 인식할 수 있으며, 발음 역시 본 프로젝트에서 우회한 방식보다 좋을 가능성이 높다. 해당 시점에서는 K-POP 정보 제공 등의 의미를 지닐 것이다.

C.2. 인식 문자열을 미리 정의해야 함

이는 Alexa 자체의 문제로, 어플리케이션에서 받을 문자열은 모두 미리 정의해서 설정해 놓아야 한다. 이 때 Slot이라는 변수를 허용하지만, Slot에 들어갈 수 있는 값 역시 미리 정의해야 한다. 즉 이를테면 한국어 번역 서비스에서, 임의의 문자열에 대해서 모두 정의할 수가 없기 때문에 번역이 불가능한 상황이다. 이 부분은 Alexa 측에서 개선해야 할 문제점이라고 생각한다. 특정 Prefix, Postfix를 정의하고 그 사이에 들어가는 문자열은 임의의 것을 받아도 될 것이다.

D. Future Plan

추후에 다음과 같은 개선점들을 생각할 수 있다.

1. 다채로운 퀴즈
    A. 간단한 O/X 형태에서 벗어나 여러 가지 형태의 재미있는 퀴즈를 출제할 수 있다.
2. 정확한 추천
    A. 추천 알고리즘의 경우 Netflix, Amazon, Facebook 등 최고의 IT 기업에서도 뜨겁게 고민하는 주제이다. 본 프로젝트 역시 음악 추천 기능이 있기에, 이 추천 기능을 깊게 발전시키는 것에는 끝이 없을 것이다.
3. Refactoring
    A. 프로젝트를 진행하면서 여러 번 Code refactoring을 거쳤으나, 아직 개선점을 찾을 수 있을 것이다.
    B. 이를테면 Lambda Intent Interface인 KalexaRequestHandler에서 간단한 로직들을 작성하다보니 소스가 길어졌는데, 이를 분화할 수 있다.
4. Failover
    A. 본 프로젝트는 외부 API에 대한 의존성이 적지 않다. 상용 서비스를 하려면 이 API가 동작하지 않을 때의 처리를 필수적으로 해야 한다.
5. 다른 컨텐츠
    A. K-POP 외에 다른 한국 관련 컨텐츠를 제공할 수 있다.
    B. 현재 인기가 많은 드라마, 쇼핑 정보 등을 제공할 수 있다.

9. Division & Assignment of Work

WORK	ASSIGN
ALEXA SKILLS KIT INTENT 및 UTTERANCE 설정	팀 전원
기계 번역/음성 합성 API 호출 LAMBDA FUNCTION 구현	최영진
FFMPEG 라이브러리 포팅	최영진
DB MODELING	팀 전원
LEARNING KOREAN 서비스 구현	최영진
차트 정보 조회 기능	이상훈
아티스트의 다른 앨범 및 노래 조회 기능	이상훈
가사 출력	김준혁
감정 상태에 따른 곡 추천	김준혁
추천 알고리즘 구현	김준혁
DATA CACHE 구현	이상훈
문제 해결	팀 전원

Table 3 Assignment of Work

10. Demo Plan

A. 일반적인 데모 방법

Alexa가 탑재된 기기인 Amazon Echo 혹은 Amazon Tap을 사용한다. 이 디바이스가 Wireless Network에 연결되어 있기만 하면 어디서든 테스트가 가능하다. 실 디바이스가

없는 경우는 마이크가 설치된 노트북 혹은 PC에서 echosim.io라는 웹 서비스를 통해 테스트할 수 있다.

원래 Alexa Skills의 설치는 Market에서 진행하나, 본 프로젝트는 아직 Market에 올리지 않아 개발자 계정을 연결해야만 한다. 개발자 계정 연결 후 Open Project Kei 등의 명령으로 어플리케이션 실행이 가능하며, 이후 질의를 할 수 있다.

디바이스의 초기 설정 및 질의 방법은 [Appendix] User Manual을 참고하기 바란다.

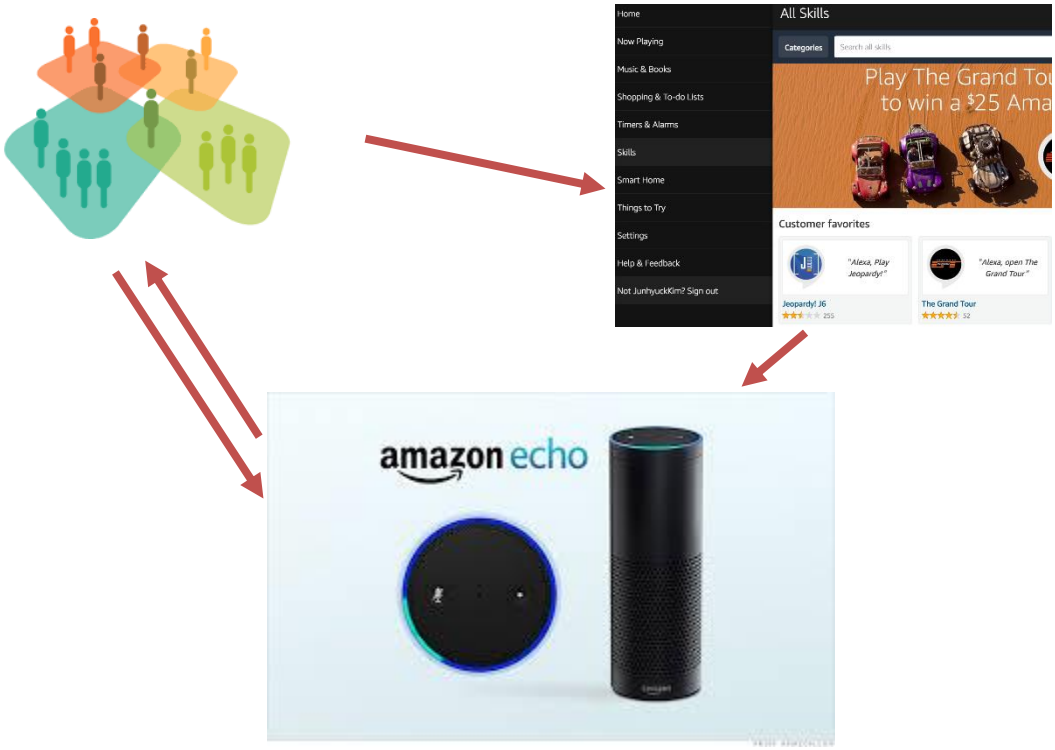

Figure 20 User 입장에서의 View

B. 시연 시 데모

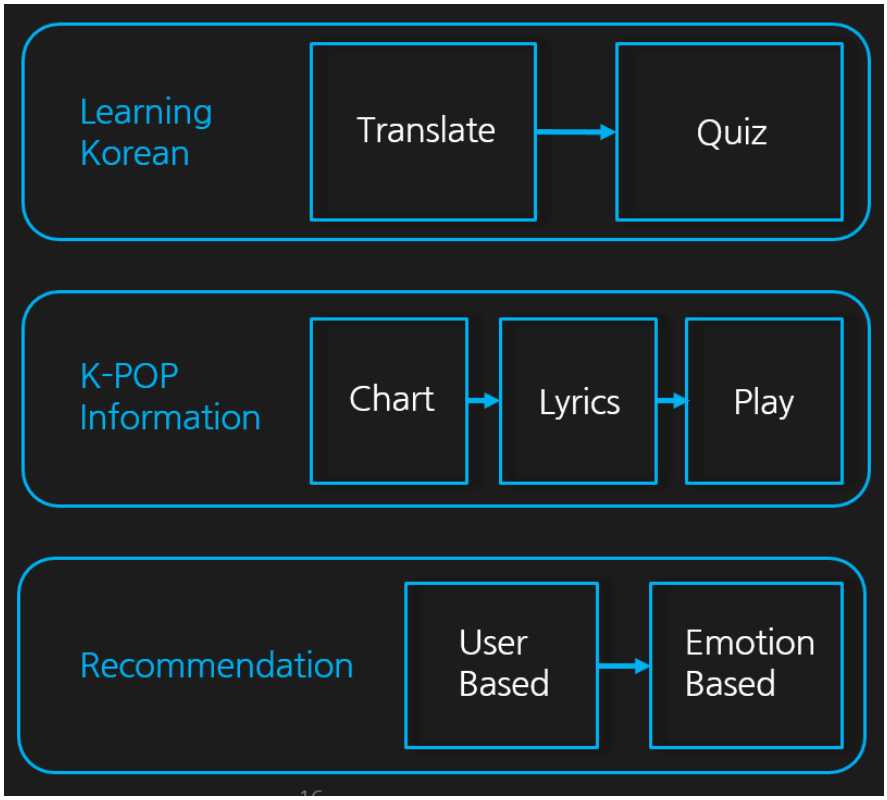

Figure 21 Demo Plan of Kalexa

실제 시연 시 질의는 다음과 같다.

[Button] Increase volume to ten.

[Button] Open project kei.

[Prompt] How can I say 'XX' in Korean?

[Prompt] Translate 'XX' into Korean.

[Prompt] Give me two quizzes.

[Prompt] The answer is True/False

[Prompt] The answer is True/False

[Prompt] We are done.

[Button] Ask project kei k-pop chart.

[Cancel]

[Prompt] Tell me the lyrics of first song.

[Cancel]

[Prompt] Play the second song.

[Cancel]

[Prompt] Let me know other song of third artist.

[Cancel]

[Prompt] Never mind.

[Button] Start project kei.

[Prompt] Recommend me any song.

[Cancel]

[Prompt] I feel happy.

[Cancel]

[Prompt] This is the end of our presentation.

11. Conclusion

현재 한국어를 지원하지 않는 알렉사에서는 부자연스러운 한국어 음성을 들을 수 밖에 없었던 상황이었지만 본 프로젝트를 통해 올바른 한국어 음성을 사용자에게 들려 줄 수 있게 되었다. 더불어 단순히 올바른 발음을 출력해 주는 것에 그치지 않고 외국인들이 관심을 가질 수 있는 한국어 학습기능과 세계적인 인기를 끌고 있는 K-POP의 정보를 제공한다.

프로젝트를 진행하는 과정에서 응답 시간과 정확성 부분에서 많은 시행착오가 있었지만 현재 접근 가능한 부분을 개선함으로써 응답 시간 단축과 정확성 향상 문제도 해결하였다. 이러한 도전 과제를 해결하는 과정에서 엄현상 교수님과 회사 담당자인 윤석찬 매니저님의 조언이 큰 도움이 되었다.

향후 많은 사용자가 이 어플리케이션을 사용하여 User data가 많이 쌓인다면 사용자 기반 추천 기능에 있어서 사용자가 정말 원하는 추천을 받았는가 즉, 추천의 신뢰도 측면에 대한 고찰이 필요하다고 생각한다.

[Appendix] User Manual

1. 기기 설정
   A. 본 프로젝트는 Alexa 음성 인식 기능이 탑재된 기기인 Amazon Echo 혹은 Amazon Tap에서 구동 가능하다. Amazon Echo는 디바이스를 Activate 시킬 때 "Alexa"라고 말하는 반면에 Tap은 기기의 마이크 버튼을 눌러야 한다. 이 점을 제외하면 디바이스 간 큰 차이가 없다.

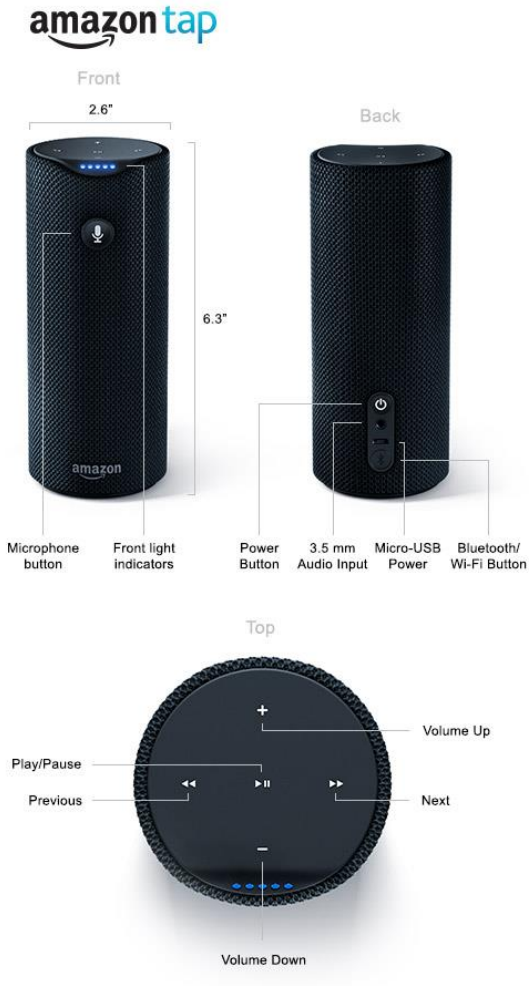

Figure 22 Amazon Tap

   B. 디바이스에 케이블을 연결하여 충전한다.
   C. 전원 버튼을 눌러 LED가 점등되는 것을 확인한다.
   D. Echo / Tap을 제대로 사용하기 위해서는 Wireless Network 또는 Bluetooth에 연결하여야 한다.
      i. PC의 alexa.amazon.com에 접속하거나, 스마트폰 Alexa 어플리케이션을 설치한다.

ii. 디바이스의 WiFi / Bluetooth 버튼을 길게 누른다. 이렇게 하면 잠시 동안 디바이스가 Wireless AP처럼 작동한다.

iii. PC 혹은 스마트폰을 그 AP에 연결한다.

iv. PC 혹은 스마트폰 메뉴에서 디바이스를 실제 Wireless Network에 연결할 수 있다.

v. 아래 화면과 같이 그 외에도 여러 가지 상태 정보 조회, 설정이 가능하다.

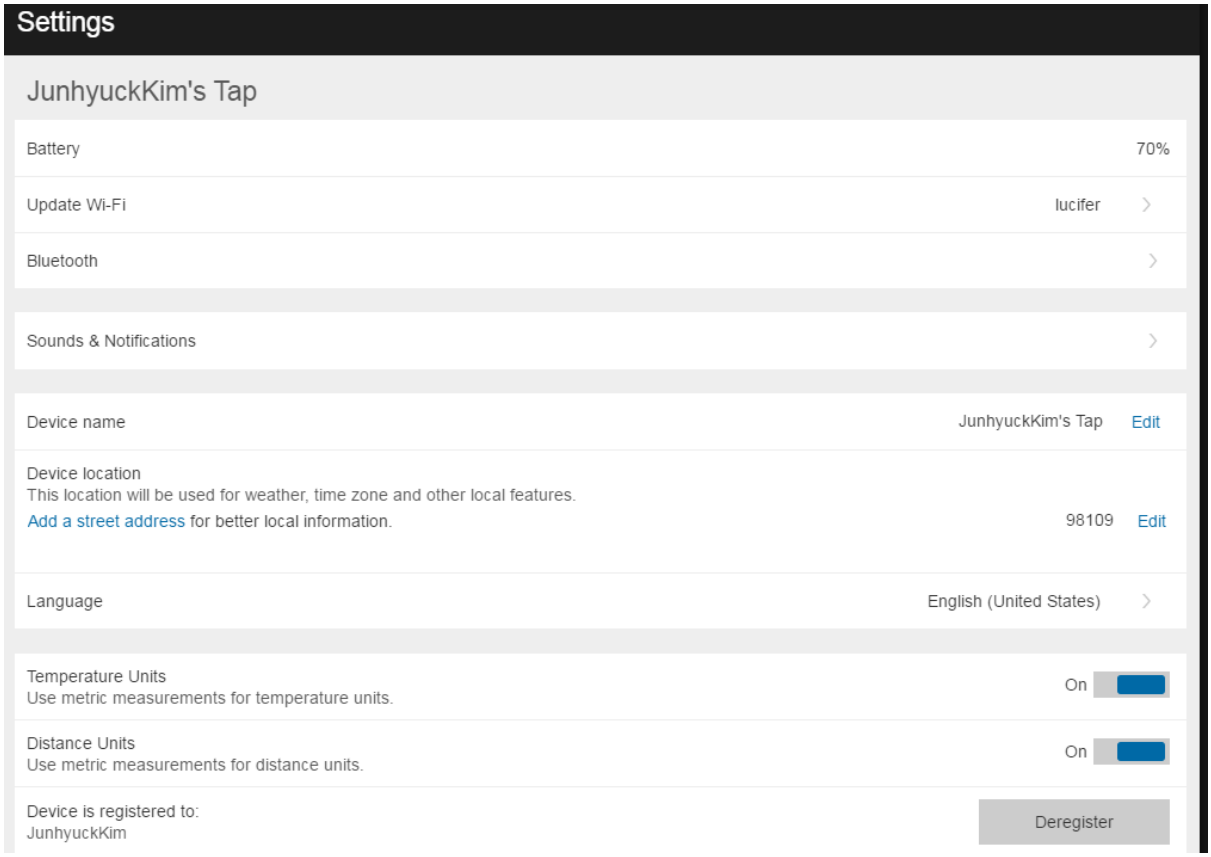

Figure 23 Amazon Tap을 PC에 연결하였을 때

E. Alexa의 어플리케이션은 Skill이라는 이름을 가지고 있다. Alexa에 연결된 PC나 스마트폰 Market에서 Skill을 Enable 시키면 디바이스에서 해당 어플리케이션을 사용할 수 있다.

F. 다만 Publish되지 않은 Skill의 경우 그 개발자 계정에서만 보이므로, 본 프로젝트의 경우 개발자 계정을 연결해야만 한다. 그 후 Skill List에서 Project Kalexa를 추가할 수 있다.

G. Project Kalexa를 Enable시키면 Alexa에서 본 프로젝트의 모든 명령을 수행할 수 있게 된다.

2. Application의 실행

A. Alexa Skill은 다음 세 가지 방법으로 실행 가능하다. 이 때 <application

name>은 "Project Kei" 이며, <command>는 본 프로젝트에서 질의할 수 있는 문장이다. Kalexa는 없는 단어로 인식률이 좋지 않기 때문에 편의상 Kei로 설정하였다.

i. One-time Execution
    1. <application name> 으로 실행, <command> 로 명령에 대한 응답을 받고 종료
ii. One-shot Phrase
    1. Ask/Tell/... <application name> to <command> 와 같은 명령으로 한 마디로 명령
iii. Prompting
    1. Open/Start/Execute/... <application name> 후 유저가 <stop> 명령을 내릴 때까지 계속해서 질의 가능
    2. 데모 시연 시 편의를 위해 이 방식으로 진행한다.

B. Alexa를 호출하여(Tap에서는 버튼을 눌러서) 음성 인식을 활성화시킨 뒤 위 방법으로 어플리케이션을 실행하고 명령을 내릴 수 있다.

C. 기능 별 명령 리스트 예시는 다음과 같다.

TYPE	DESCRIPTION	COMMAND(일부)
번역	주어진 영어 문자열을 한국어로 번역	Translate 'XX' into Korean. How can I say 'XX' in Korean?
퀴즈	한국어 OX 퀴즈를 통하여 한국어 학습	Give me [number] quizzes. The answer is true / false.
K-POP 차트 정보	K-POP 차트에 대한 정보 조회	Let me know the chart. Let me know the popular songs in K-POP. I want to know K-POP chart.
가사 정보	K-POP 가사 조회	Tell me the lyrics of [nth] song.
아티스트 정보	해당 아티스트의 다른 곡 조회 및 재생	Let me know other song of [nth] artist. Play [notInChart] song of [nth]

		artist in the chart.
음악 재생	K-POP 음악 감상	Play the [nth] song.
감정 기반 추천	현재 기분 상태에 따라 추천된 음악 감상	I feel [emotion]. I am [emotion]
유저 데이터 기반 추천	플레이 기록 및 다른 유저 데이터에 따라 추천된 음악 감상	Recommend me any song.
시작 및 종료	어플리케이션 실행 및 종료	Open / Start / Execute / Ask / … Project Kei. Stop. / Never mind. / We're done. / … (Wait until end)

Table 4 Kalexa 명령어

TYPE	DESCRIPTION	EXAMPLES
Emotion	감정 관련 표현	Sad, Happy, Upset, ….
Number, Nth	숫자	One, Two, … First, Second, …
Sentences	번역 문자열	Apple, Desk, Pencil, What is your name?, …
Quiz List	퀴즈	I can't believe you did that. / 당신이 그렇게 했다는 것을 믿을 수 없습니다.

Table 5 Slot 및 퀴즈 예시

[Appendix] Detailed Implementation Spec

A. KalexaRequestHandler

AlexaSkillsKit과의 기능별 람다 사이에서 게이트웨이 역할을 하는 함수이다. AlexaSkillsKit에서 보내는 request를 파싱하여 각 Intent별로 람다 호출을 한다. 또한, 기능별 람다가 리턴한 결과를 AlexaSkillsKit의 형식에 맞게 응답하는 wrapping함수를 포함한다.

B. PlayLyrics

Parameter	@int : 재생하고자 하는 곡의 songId
Return	String : 가사 파일의 url
Implementation	songId 에 해당하는 곡의 가사 파일이 이미 만들어있는지 DB 확인. - 있을 경우 url 그대로 리턴 - 없을 경우 songId 로 멜론 홈페이지에서 HTML 파싱하여 가사 획득. 획득한 가사로 TTS 람다 호출 결과로 리턴 받은 mp3 파일 url DB 에 저장 후 해당 url 리턴
Additional Info	

C. EmotionBasedRecommendation

Parameter	@text string : 감정을 나타내는 어휘 혹은 표현
Return	String : mp3 file url
Implementation	입력된 감정의 상위 분류(희로애락)의 선곡 리스트 중 랜덤 곡 리턴
Additional Info	

D. CollaborationFiltering

Parameter	@userId : 세션 유저 아이디
Return	String : mp3 file url
Implementation	현재 유저의 취향과 비슷한 다른 유저의 플레이리스트에서 추천곡 선정하여 리턴
Additional Info	

E. Translate

Parameter	@text string English string to translate into Korean
Return	Translated Korean string
Implementation	네이버 기계 번역 API 호출 (source: en, target: ko, text: @text) UTF-8 인코딩 사용
Additional Info	

F. TTS

Parameter	@text string Korean string to speech @fileName string (optional) S3 upload file name Default: temporary name by datetime
Return	URL of MP3 which reads given string parameter (Playable in Alexa)
Implementation	네이버 음성 합성 API 호출 (speaker: Mijin, speed: 0, text: @text) UTF-8 인코딩 사용 FFMPEG 라이브러리 사용 Alexa 플레이 가능한 포맷으로 변환 <FFMPEG Options> -ac 2 -codec:2 libmp3lame -b:a 48k -ar 16000 -ss 00:00:00 -t 00:01:25
Additional Info	내부적으로 디렉토리 처리 등 부가 로직 FFMPEG 실패 시 재시도

G. 한국어 번역

Parameter	What is {sentence} in Korean? 등의 한국어 번역 질의
Return	한국어 번역 결과 재생
Implementation	Alexa 에서 번역 질의를 받으면 번역 해당 부분 파싱하여 전달 Translate, TTS 를 거쳐 한국어 MP3 파일 생성 MP3 파일 URL Alexa 에 전달하여 재생
Additional Info	{sentence}는 Predefined Set

H. 한국어 퀴즈

Parameter	Give me {number} quizzes. 등의 한국어 퀴즈 질의 The answer is true/false 등의 응답

Return	영어-한국어 세트로 이루어진 퀴즈 출력 응답의 경우 채점 결과
Implementation	Alexa 에서 퀴즈 질의를 받으면 퀴즈 리스트에서 랜덤하게 퀴즈 전달 영문 텍스트 및 한글 MP3 파일 URL 은 저장되어 있음 이 때 유저 상태를 저장하여 유저의 응답 채점 가능
Additional Info	퀴즈 리스트 및 유저 상태 정보는 DB 에 저장

I. KpopChart

Parameter	
Return	차트 정보 - list of song objects { songId(int) , songName(string) , artists(list) } artists : list of artist objects { artistId(int), artistName(string) }
Implementation	멜론 실시간 차트 API 호출 (version: 1, page: 1, count: 10) 차트정보 DynamoDB 에 update
Additional Info	

J. SpeakKpopChart

Parameter	
Return	차트 정보
Implementation	KpopChart 호출하여 실시간 차트 정보 받음 차트 정보에서 songName 과 artistName 을 음성 변환에 적절한 String 으로 변경 TTS 에 넘겨줄 @fileName 값을 “songId” + “1”, “artistId” + “1”로 설정 TTS 람다 호출
Additional Info	S3 에 해당 MP3 파일이 없는 경우에만 TTS 람다 호출

K. 차트 정보 제공

Parameter	1. I want to know the chart 등의 1~5 위 차트 정보 요구 질의 2. Let me know the chart {fromSixth} 등의 6~10 위 차트 정보 요구 질의 3. Who is the {Nth} song’s artist 등의 가수 정보 요구 질의
Return	1. 차트 1~5 위 곡 제목 재생 2. 차트 6~10 위 곡 제목 재생 3. 해당 순위 곡 제목과 가수 이름 재생
Implementation	SpeakKpopChart 람다 함수를 호출하여 차트 곡 제목과 가수 이름 S3 에 갱신 리턴 받은 차트 정보로 곡 제목과 가수 이름 MP3 파일의 URL 도출

	질의 종류에 따라서 재생할 MP3 URL 을 Alexa 에 전달
Additional Info	{fromSixth}: 6 위부터 라는 의미를 가진 표현 {Nth}: 몇번째인지를 나타내는 표현

L. GetOtherSongs

Parameter	@songId (int), @artistId (int), @artistName(string)
Return	list of at most 4 songs
Implementation	멜론 곡 검색 API 호출 (version: 1, page: 1, count: 5, searchKeyword: @artistName) API 호출로 받은 5 개의 곡 리스트에서 해당 곡 songId 와 @songId 의 비교를 통해 차트에 있지 않은 노래 구별하고 4 개만 리턴 4 개 이하일 경우 songId 와 @songId 비교 후 모두 리턴
Additional Info	

M. ArtistBasedRecommendation

Parameter	@nth (int)
Return	list of at most 4 songs
Implementation	DynamoDB 에서 차트 정보 받아서 @nth index 에 있는 곡 정보 획득 GetOtherSongs 람다 함수 호출하여 추천 곡 list 받음 곡 정보에서 songName 과 artistName 을 음성 변환에 적절한 String 으로 변경 TTS 에 넘겨줄 @fileName 값을 “songId” + “1”, “artistId” + “1”로 설정 TTS 람다 호출
Additional Info	S3 에 해당 MP3 파일이 없는 경우에만 TTS 람다 호출

N. 순위권 가수의 차트에 없는 다른 곡 정보 제공

Parameter	I want to know other song of {Nth} artist 등의 다른 곡 추천 요구 질의
Return	해당 가수의 이름과 최대 4 개의 곡 제목 재생
Implementation	ArtistBasedRecommendation 람다 함수 호출 하여 곡 리스트 획득 곡 정보에서 songId 와 artistId 를 이용하여 해당 MP3 URL 도출 MP3 URL 을 Alexa 에게 전달
Additional Info	{Nth}: 몇번째인지를 나타내는 표현

O. I, L 에서 제공된 노래 재생

Parameter	1. Play {Nth} song in the chart 등의 차트 해당 순위 곡 재생 질의

	2. Play {NotinChart} song of {Nth} artist in the chart 등의 해당 가수의 차트에 없는 곡 재생 질의
Return	1. 해당 차트 순위 곡 재생 2. 해당 차트 순위 가수의 또다른 곡 재생
Implementation	{NotinChart}가 없는 경우 DynamoDB 에서 차트 정보 받아서 {Nth} 번째 곡 songId 로 음원 MP3 URL 도출 후 Alexa 에 전달 {NotinChart}가 있는 경우 {Nth} 정보를 해당 int 값으로 변환 후 ArtistBasedRecommendation 람다 함수 호출하여 곡 list 획득 곡 list 에서 임의의 songId 를 선택하여 음원 MP3 URL 도출 후 Alexa 에 전달 UserId 에 따른 곡 재생 정보 DB 에 업데이트 가장 최근에 들은 노래 정보 DB 업데이트
Additional Info	{Nth}: 몇번째인지를 나타내는 표현 {NotinChart}: 차트에 없는 다른 곡을 의미하는 표현