음악 가사 데이터베이스 프로젝트

서문

일반적으로 노래는 멜로디와 가사로 이루어져있다. 포털사이트를 비롯하여 음악의 가사를 찾는건 어렵지 않지만 현재 듣고 있는 멜로디와 가사가 싱크된 싱크가사가 쉽게 이용 가능하게 제공되지 않는다.

실시간 싱크가사를 제공하는 비영리의 거대한 데이터베이스는 존재하지 않으며,(일부 싱크가사 제공를 사이트는 있으나 일반적으로 소규모), 음원사이트나 musixmatch나 알송과 같은 상용 서비스를 이용하거나 이러한 서버의 비공식 api 통해 싱크가사를 얻어올 수 있지만 이 경우 합법성과 지속 가능성에 의문이 제기된다.

일부 음원 사이트는 mp3를 다운 받을 때 싱크가사를 제공하지만, CD로 직접 리핑한 mp3를 소유중인 경우 혹은 이러한 가사를 제공하지 않는 사이트를 통해 mp3를 구매후 이용하는 경우는 이러한 싱크가사 서비스에 의존할 수 밖에 없다.

사례연구

한국에서 가사를 제공하는 음악플레이어는 알송과 곰오디오가 있으며,

두 소프트웨어는 공통적으로 메타데이터의 가수와 곡이름 정보를 이용하여 가사를 표시하거나,

태그데이터를 skip후 음악데이터의 시작부분에서 정해진 byte 만큼의 데이터를 md5로 hash하여 해당하는 싱크 가사를 찾아 표시한다.

또한, 가사 등록시 수집한 음악의 태그정보와 사진을 이용한 자동태그를 지원하며, 자동태그 메뉴를 통해 파일의 md5에 해당하는 태그를 가져오거나 직접 데이터베이스를 검색하여 파일에 태그정보(메타데이터)를 삽입할 수 있다.

구현목표

사례에서 제시된 방식과 마찬가지로 파일의 해쉬를 기반으로 싱크가사와 자동태그를 지원하는 데이터베이스를 구현하는것이다.

가사를 제공하는 lyric테이블, 음악에 대한 메타데이터를 담고 있는 song 테이블, 파일의 해쉬와 가사를 일치시키는 hash\_to\_table, 그리고 lyric테이블의 히스토리를 추적하기 위한 lyric\_revision 테이블로 구성된다.

테이블 설계시 먼저 고려한 사항은 단순히 가사 데이터베이스인가, 음악데이터베이스로서 가사를 제공하는 방식에 가깝냐는 것이냐는 것이다.

현재 시스템은 파일의 메타데이터를 제공하는 기능과, 가사를 제공하는 두가지 기능이 존재하며,

1.이를 별개의 기능으로 간주하여 서로 연관없는 테이블로 간주하는 안

2.기본적으로 Lyric으로만 이루어진 데이터베이스로서 Lyric 테이블에 모두 저장하는 방식

3.. song이라는 테이블을 먼저 만들고 가사테이블이 참조하는 안

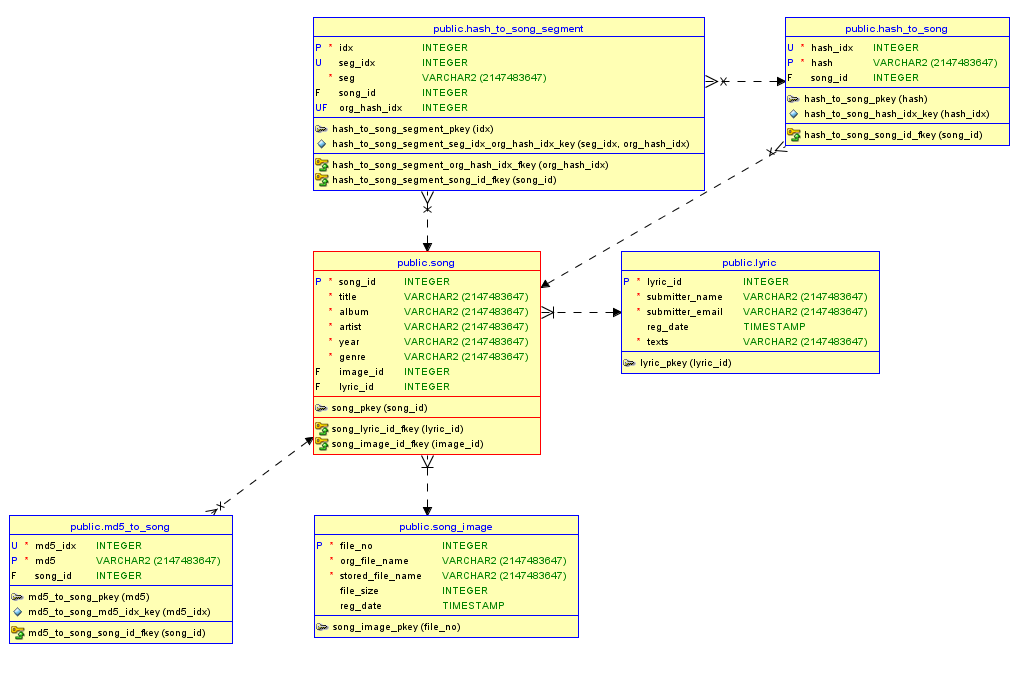
곡마다 하나의 가사를 가진다면, song테이블에서 Lyric을 참조하고, (혹은 그 반대도 가능 1:1 관계이므로) 곡마다 여러가사를 보유한다면, Lyric 테이블이 부모 Song테이블을 참고할 것이다.

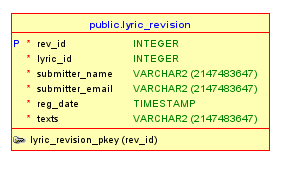
실제로 같은 음악이 여러 개의 Lyric을 가질 이유가 없기 때문에, 3번안의 Song테이블이 Lyric을 가르키는 방식으로 결정하였다.

song 테이블은 album, artist등의 필드가 별도의 테이블이 분리되어야 할 필요성이 있으나, 검증용 프로토타입으로 진행되는 만큼 나중에 확장을 고려하기로 한다. 대신 사용자의 음악 파일에 있는 태그를 받아 그대로 들어가게 된다.

Lyric의 경우 오수정으로 인한 데이터손실을 방지하기 위해 revision 테이블을 구성하였지만, Song 테이블은 재설계 가능성이 크고, 변경을 염두에 두지 않았으므로, 별도의 추적 테이블은 구성하지 않았다. 만일 이러한 테이블에도 추적이 필요하다면, 통합적으로 Audit history을 관리할 수 있는 테이블을 설계해야 할것으로 보여진다.

[현재 SQL 구조]





개선 가능한 점

현재 알송/곰오디오 방식에는 개선 가능이 필요한 부분은 두가지가 있다.

1. 하나는 같은 음악임에도 인코딩 파일 포맷등의 인해 같은 음악임을 인식할 수 없는점

2. 싱크가사를 사람이 직접 작성해야 하는 점

이중, 상대적으로 난이도가 낮은 1번을 개선하고자 하며, 2번의 경우 약간의 연구가 필요한다고 보여진다. 따라서 이에 대한 내용은 한계점에 서술해 두었다.

음악파일에 해당하는 음악에 대한 메타데이터, 가사를 찾는 방식에는

1.먼저 원하는 곡에 대한 정보를 찾는 방식에는 태그를 이용하여 검색하는 방식이 있다.

문제는 실제로 태그가 매겨지는 방식은 태거에 따라 다를 수 있는 위험이 있다는 것이다.

2. 두번째 방식은 파일의 hash를 이용하는 방식이며, 메타데이터를 스킵후 음악데이터를 해쉬하여검색하는 방식이며, 현재 사용하는 방식이다.

3. 세번째 방식은 오디오 핑거프린트를 사용하는 방식이다.

이 경우 인코딩이나 파일포맷에 상관없이 인식이 가능하지만, 잘못 인식되거나 Remix나 intro의 구분이 어려울 수 있다. 또한 상대적으로 처리에 필요한 시간이 많다.

따라서 세가지 방식 모두 적절히 조합되서 사용되어야 할 것이다.

음악을 통해 곡의 이름을 찾는 방식에는

1.동일한 음악이 재생하였을 때 검색가능한 방식과

2.허밍으로 검색할 수 있는 QBH(Query by Humming)등이 있으며,

이에 해당하는 대표적으로 사용되는 서비스 및 오픈소스 프로젝트에는

1. Shazam, Chromaprint, Echoprint, Libofa, Lastfm의 fplib, openfp, emysound

SoundFingerprinting, nelemans1971/AudioFingerprinting, fdmf 등이 있으며

2. SoundHound나 구글의 Hum to Search가 대표적이다.

Music Retriaval 분야의 고전적인 주제인 만큼, 많은 상용화된 서비스가 존재하며, Mirex대회에서도 관련 주제로 여러번 다뤄진바 있다.

이중에서 가장 우수하고, 자료가 많은 Shazam[[1]](#footnote-1) 알고리즘을 바탕으로 한 검색 시스템을 사용하고자 하였지만, 구현상의 어려움으로 인해 Libfooid로 검증후 필요시 다른 알고리즘을 활용하기로 하였다.

(Shazam 알고리즘의 경우 특허 문제가 존재하며, 곧 만료되거나 만료될 예정이다. 한국의 경우 이미 만료되었고, 미국은 2025년에 만료될것으로 보인다.)

Shazam 알고리즘 자체는 상당히 간단한데 먼저, 스펙트로그램에서 peak값을 추출한다.

그리고 일정 거리안의 두 peak값의 주파수, 시간, 거리로 hash값을 만든다.

이 peak값들은 노이즈에도 다수가 유지되기 때문에 이를 통해 곡을 찾아 낼 수 있다.

현재 사용중인 Libfooid의 구현은 다음과 같은 알고리즘으로 이루어져있다,.

먼저 노래의 시작으로 부터 90초를 가져오고 8khz의 모노로 변환하고 FFT를 수행후 16개의 바크 척도로 변환한다. 그리고 이름 424바이트의 지문으로 변환한다.

87프레임 데이터로부터 추출된 프레임당 (척도당 2비트를 가지는) 4바이트 fit와 프레임당 6비트의 dom line(프레임의 주요 멜로디 라인)으로부터 생성된 424바이트의 핑거프린트는 10바이트의 헤더와 348바이트의 fit데이터와 66바이트의 dom으로 이루어져있으며 데이터베이스에 존재하는 핑거프린트와 음악파일에 핑거프린트를 비교함으로써 일정 기준 이상이면 같은 음악으로 판단한다.

이 핑거프린트와 유사한 곡을 어떻게 데이터베이스에서 가져오냐는 문제가 존재하는데

1. 데이터베이스에 있는 모든 곡의 핑거프린트와 비교하는 방식

의 경우 곡의 크기가 늘어남에 따라 상당히 비효율적일 수 있으며, 이를 해결하기 위해

1. 핑거프린트를 일정크기로 잘라서 구성된 역인덱스를 만들고 역인덱스에 있는 곡들만을 비교하는 방식
2. 핑거프린트를 일정길이로 잘라서 넣고 인덱스에 맞는 가져온후 count를 센후 같은 곡으로 판단

의 방식을 생각했으나 역시 나름의 문제가 존재한다.

현재 고려중인 3번의 경우 곡이 적은 경우 1번 방식의 비해 많은 저장공간과 연산이 필요하며, Select구문이 여러 번 수행되는 만큼 처리속도가 상당히 느릴것으로 예상된다. 하지만, 곡이 늘어남에 따라 더 효율적일 수 있다.

따라서 어떤 방식이 더 효율적일지는 적절한 고려가 필요하다.

한계점

저작권 문제

가사에는 저작권이 존재하며, 이를 무단으로 제공하는 것은 문제가 될 수 있다.

충돌가능성이 높은 알고리즘

현재 의존할 예정인 fooid는 처음 90초의 음악데이터를 가져와 핑거프린팅하는 간단한 알고리즘에 의존하고 있으며 간단한만큼 핑거프린트가 충돌하거나 false postive와 같은 문제가 발생할 수 있다. 또한 현재 상용화된 시스템과 같이 노래의 일부만을 듣고 곡을 인식하지 못한다

또한 핑거프린팅 기법으로 데이터베이스를 직접 구축 할 경우 데이터베이스에 있는 곡이 한정적일 뿐만 아니라 알고리즘 교체시 다시 데이터베이스를 구축해야 하는 위험성이 있다.

싱크가사를 위해서는 실제 사람에 의존해야 하는 비효율성

현재 싱크가사를 자동으로 생성하는 방법이 존재하지 않으며, 싱크가사를 제공하기 위해서는 수작업으로 싱크를 맞춰야 하는 비효율성이 존재한다.

가사와 오디오를 align하는 것 또한 Music retrieval 분야의 오래된 주제이며 MIREX 대회가 언급되는것을 알 수 있다.

음성인식 기술의 발달로 가사와 음악을 제공하여 싱크가사를 생성하는 기술이 상용화되었으며,

실제로 가우디오랩이라는 기업의 솔루션을 통해 벅스에서 자동 싱크가사를 제공하는 것으로 알려져있다.

이를 위해서는 음성인식을 통한 가사인식, 가사의 음절길이와 무음, 음악의 길이를 이용하여 가사를 일치시키는 방법, 가사에 해당하는 음성을 TTS를 통해 합성한후 DTW를 이용해 Forced Aligment를 사용하는 방법등도 제시되었지만, 실제로 가장 성공적인 방식은 Phoneme level(음소레벨) alignment를 이용 하는 것이다.

NUS AutoLyrixAlign[[2]](#footnote-2)를 비롯한 ALTA[[3]](#footnote-3) plla-tisvs[[4]](#footnote-4) Spotify의 논문[[5]](#footnote-5) dezzer의 논문[[6]](#footnote-6)등 Phoneme level를 이용하는 여러 논문이 제시되었으며, 일반적으로 Kaldi를 사용하거나 직접 개발된 커스텀 모델을 통해 이러한 가사일치를 달성한다.

일반적으로 kaldi를 통해 초보적인 한국어 인식을 훈련하기 위한 레시피의 최소 조건으로는 1080ti 혹은 3060에 해당하는 그래픽카드가 요구되는것으로 알려져있으며[[7]](#footnote-7) 한국어에 맞춘 추가적인 고려가 필요하다.

추가적으로 한국에서 제공되는 노래는 한국어 가사뿐만 아니라 영어가사 인식이 필수적이며, 가까운 나라인 Jpop 또한 많이 소비되는 만큼 일본어 가사의 인식도 높은 수요가 있다.

이를 위해서는 영어와 한국어의 혼합 훈련이 (혹은 불가능하다면, 문장을 한국어 발음으로 바꾼후 일치시키는 등의 처리가) 요구된다고 보여진다.

dezzer의 논문에 따라면 phoneme일치에 더해 가사를(espeak라이브리를 이용하여) IPA로 변환하여 훈련시켜 다국적 가사일치를 달성했다고 하지만 espeak의 한국어 합성 문장의 퀄리티를 볼 때 실제로 한국어 훈련에서도 IPA방식의 훈련이 도움이 될지는 의문점이 있다.

1. An Industrial-Strength Audio Search Algorithm https://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf [↑](#footnote-ref-1)
2. https://github.com/chitralekha18/AutoLyrixAlign [↑](#footnote-ref-2)
3. <https://github.com/emirdemirel/ALTA> https://github.com/emirdemirel/ASA\_ICASSP2021 [↑](#footnote-ref-3)
4. https://github.com/schufo/plla-tisvs [↑](#footnote-ref-4)
5. https://ieeexplore.ieee.org/document/8683470 [↑](#footnote-ref-5)
6. https://program.ismir2020.net/static/final\_papers/101.pdf [↑](#footnote-ref-6)
7. https://github.com/goodatlas/zeroth, [↑](#footnote-ref-7)