빅데이터프로그래밍 Team Project

60182180 안성표

60191948 이재혁

1. 문제 정의

현대 사회에서 음악은 각 시대의 정서, 문화, 사회적 변화를 반영하는 중요한 매개체이다. 노래는 특히 그 시대의 감정, 가치관, 그리고 예술적 표현을 담아내기 때문에 노래 데이터를 통한 분석은 그 시대를 더 깊이 이해하는 데 도움이 될 수 있습니다. 본 프로젝트에서는 멜론 웹사이트에서 수집한 노래 데이터를 활용하여 다음과 같은 목표를 달성하고자 한다.

* 감정 분석: 노래 가사 텍스트를 활용하여 연도별 노래의 감정을 분석한다. 이를 통해 각 시대별로 어떠한 감정 표현이 주를 이루는지 분석하고 이렇게 나타나는 이유를 생각해본다.
* 트렌드 분석: 노래 데이터를 통해 시대별 음악 트렌드를 분석한다. 어떠한 음악적 장르가 특정 시기에 유행하고 있는지를 조사하여 음악 문화의 시대별 트렌드 변화를 파악한다.

2. 시스템 아키텍처

텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

먼저 멜론에서 데이터를 크롤링한다. 이때, 멜론 웹사이트에서 데이터를 크롤링하므로 Selenium 라이브러리를 사용한다. 그리고 Selenium의 driver를 사용하여 웹페이지의 html을 가져오고 이를 BeautifulSoup 라이브러리를 통해 데이터를 수집하기 쉬운 구조로 파싱한다. 수집된 데이터를 HDFS에 저장하고 Spark를 사용하여 데이터 전처리 및 분석하고 결과를 Zepplin을 사용하여 시각화한다. Spark를 사용한 이유는 MapRuduce, pig, Hive 등에 비해 데이터프레임 형태의 데이터를 다루기 훨씬 편리하고 속도와 효율성 측면에서도 좋기 때문이다.

3. 데이터 수집 방법

데이터 크롤링 코드가 매우 길기 때문에 나눠서 설명한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

먼저 필요한 라이브러리 및 패키지를 import한다. 웹페이지를 크롤링할 때 chrome 브라우저를 사용하여 크롤링하기 때문에 이에 필요한 옵션을 설정한다.

chrome\_options.add\_argument("--headless") : 브라우저 창을 띄우지 않고 백그라운드에서 실행하는 headless 모드를 활성화한다. 리눅스 환경에서는 실행할 때 반드시 필요하다.

chrome\_options.add\_argument("--disable-gpu") : GPU 가속을 사용하지 않도록 설정한다. 일부 환경에서 GPU 가속이 문제를 일으키는데, 이 버그는 크롬 자체에 있는 문제이다. 따라서 해당 옵션을 설정해줘야 한다.

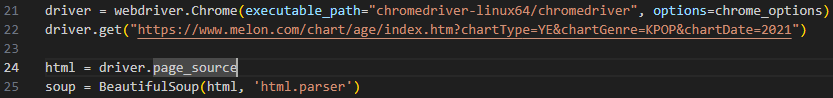
chrome\_options.add\_argument("--disable-dev-shm-usage") : 이는 일부 리눅스 시스템에서 필요한 설정으로, 브라우저를 실행할 때 메모리 사용을 최적화하는 데 도움이 된다.

chrome\_options.add\_argument("--no-sandbox") : 브라우저의 샌드박스 모드를 비활성화한다. 일부 환경에서는 샌드박스 모드가 제대로 동작하지 않을 수 있어 이를 비활성화할 필요가 있다.

my\_user\_agent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36"

chrome\_options.add\_argument(f"--user-agent={my\_user\_agent}")

사용자 에이전트(User Agent)는 브라우저가 서버에게 전송하는 정보로, 이를 변경하여 서버에 접속할 때 사용할 브라우저 정보를 설정할 수 있습니다. 이를 사용하지 않으면 ip나 방화벽 차단이 걸릴 수 있으므로 작동하는 브라우저를 나의 Chrome 브라우저로 속여 차단으로부터 방지한다.



Selenium을 사용하여 Chrome 브라우저를 초기화 및 실행하고, 해당 페이지의 소스 코드를 가져와서 BeautifulSoup으로 파싱한다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

수집하려는 노래 데이터의 연도와 이에 해당하는 url, 그리고 수집할 데이터의 개수를 지정하는 count를 함수의 파라메터로 받아서 해당 페이지에서 노래 데이터를 크롤링하는 메서드이다. 중간중간 time.sleep()을 사용하여 차단으로부터 방지한다. 웹페이지의 html을 파싱하고 내가 원하는 노래의 title과 singer를 얻기 위해 해당 부분을 find하고 리스트에 저장한다. 리스트에 append하고 delete하는 이유는 내가 원하는 순위까지만 데이터를 얻기 위함이다. 예를 들어 top10 노래 데이터만 수집할땐 count가 10이므로 해당 인덱스 이후의 데이터는 삭제한다. 마지막에 songid를 추출하는 이유는 현재 driver.get으로 한 url 웹페이지에는 노래 제목, 가수, 노래 id 등은 있지만 노래 가사와 장르에 해당하는 데이터는 없다. 따라서 songid를 가져와 해당 노래의 가사와 장르가 있는 웹페이지로 이동하여 노래 가사와 장르를 수집한다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

위에서 말했듯, songid에 해당하는 웹페이지를 새로 가져와 파싱하고, 노래 가사와 장르를 가지고 있는 부분을 찾아 데이터를 수집한다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

수집한 데이터를 데이터프레임 형태로 바꾸고 해당 데이터를 csv파일로 HDFS에 저장한다. 해당 데이터가 HDFS 뿐만 아니라 현재 로컬에도 저장되므로 로컬에선 삭제한다. 저장된 csv파일의 이름은 melon1964.csv , melon1965.csv, … , melon2022.csv 이다. 각 연도별로 hdfs에 파일이 저장된다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Melon\_collector 메서드를 실행하는 부분이다. 멜론 사이트에 1964년 이전의 데이터는 없으므로 1964년부터 시작하여 2022년까지 데이터를 수집한다. 2023년이 아직 끝나지 않아 2023년에 대한 노래 순위는 아직 집계되지 않았다. 또한 1990년 이전의 노래 데이터는 너무 오래 전이라 그런지 top100은 커녕 top50에 대한 데이터도 없다. 따라서 1990년 이전에 대해선 top10 데이터만 수집하고 이후 데이터는 top50까지 수집한다. Top100까지 수집할 수 있지만 시간이 데이터 크롤링이라 시간이 매우 오래걸리고, 특히 ip 및 방화벽 차단의 위험이 있기 때문에(실제로 제가 차단당해서 프로젝트를 진행하지 못할 뻔하여 줄였습니다.) top50까지 수집한다.

4. 데이터 분석 방법

데이터 분석은 연도별 노래 데이터 감정 분석과 연도별 장르 트렌드 분석으로 총 2가지가 있다.

4-1. 노래 데이터 감성 분석

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

감성 분석에 필요한 라이브러리 및 패키지를 import 한다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Categorize\_decades 메서드는 현재 데이터프레임의 연도 컬럼에는 각 노래가 발매된 연도인 1964, 1965, … 2022까지 할당되있으므로 이를 1960, 1970, … 2020으로 바꿔주는 메서드이다. 즉 연도를 10년을 기준으로 범위를 나눈다.

Directory\_path는 크롤링한 데이터가 저장되있는 파일의 hdfs 경로이며 file\_names은 각 파일의 이름이다. 따라서 각 파일을 spark.read를 통해 읽고 리스트에 추가한 뒤 데이터프레임형태로 변환한다.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

노래 가사의 감정을 분석하므로 감정이 없는 행은 제거한다. 그리고 앞에서 언급한 categorize\_decades 메서드를 통해 연도 범주 컬럼을 새로 생성하고 노래 가사 텍스트의 감정을 분석하기 위해 translator를 통해 번역하고 감성을 분석한다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

nltk.download('vader\_lexicon') : NLTK(Natural Language Toolkit) 라이브러리에서 VADER Lexicon 데이터를 다운로드한다. VADER Lexicon은 VADER (Valence Aware Dictionary and sEntiment Reasoner) 감정 분석 모델을 지원하기 위한 감성 사전으로, 단어의 감성 점수 및 극성 정보가 포함되어 있습니다. VADER 감정 분석을 사용한 이유는 VADER 감정 분석은 특히 텍스트 데이터에서 감정을 감지하고 분류하는 데 사용되는 빠르고 효과적인 방법 중 하나이며, 문맥을 고려한 감정 분석, 감정 강도 및 극성 분류가 가능하여 해당 모델을 사용했다.

4-2. 데이터 분석 결과

감정 분석 완료 후, 최종 목표를 위해 연도\_범주 별로 그룹화하고 감정 점수의 평균을 구한다. 실행 결과는 다음과 같다.

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

결과를 시각화한 결과는 다음과 같다.

텍스트, 스크린샷, 도표, 그래프이(가) 표시된 사진

자동 생성된 설명

요즘 시기가 혐오의 시대라는 말이 있는데, 이러한 말과는 달리 노래의 감정 점수는 현대로 올수록 점점 긍정적으로 변하고 있다. 이러한 경향성을 띄게 된 이유를 유추해보자.

최근 멜론 노래 차트는 옛날과 달리, 노래도 중요하지만 그 노래를 부른 가수의 영향력이 매우 크다고 생각한다. 특히 최근 멜론 차트를 보면 결국 노래가 좋고 나쁨보다는 인기 많은 아이돌의 노래가 차트 상위권의 대부분을 차지하고 있다. 그리고 이러한 아이돌의 노래는 긍정적인 경우가 많으므로 이로 인한 영향으로 이러한 경향성을 띄게 된 것 같다고 생각한다.

4-3. 연도별 장르 트렌드 분석

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

트렌드 분석에 필요한 라이브러리 및 패키지를 import 한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

SparkSession.builder.appName("Music Trend Analysis").getOrCreate()를 통해 Spark 세션을 생성합니다.

load\_dataframe 함수는 HDFS(분산 파일 시스템)에서 특정 연도의 데이터를 로드합니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

여러 연도 데이터 통합

year\_range는 분석할 연도 범위를 정의합니다. 각 연도의 데이터를 로드하여 all\_years\_df에 통합합니다. 이는 모든 연도 데이터를 하나의 DataFrame으로 결합하는 과정입니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명장르 인기도 분석 groupBy를 사용하여 각 연도별로 장르의 출현 횟수를 계산합니다. 이는 장르별 인기도를 파악하는 데 사용됩니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

연도별 최고 인기 장르 추출: Window.partitionBy("Year").orderBy(col("count").desc())를 사용하여 각 연도별로 데이터를 그룹화하고, 출현 횟수(count)에 따라 내림차순으로 정렬합니다. filter(col("rank") == 1)를 사용하여 각 연도별로 가장 인기 있는 장르만 선택합니다.

분석 결과를 시각하여 얻은 결과는 아래와 같습니다.

텍스트, 그래프, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

1960년도부터 1980년 중반까지는 트로트나 포크/블루스 같은 성인 가요가 주를 이뤘는데 1980년대 후반부터 발라드의 점유율이 급상승하였는데 이는 1980년대부터 정부가 컬러TV 방영을 허용하면서 각종 드라마와 티비쇼에서 발라드 장르의 음악들이 큰 인기를 얻기 시작하여 2000년대 말까지 그 인기가 이어지다가 2000년대부터 아이돌 연예산업이 크게 성장하여 댄스음악이 발라드를 꺾고 높은 점유율을 보이게 되었습니다. 그 후 2010년대 중반에는 쇼미더머니와 같은 힙합 음악 프로그램의 성공으로 잠시 힙합음악의 점유율이 높았던 적이 있고 최근에는 다시 발라드 음악이 강세를 보이고 있습니다.

음악은 그 시대상을 담고 있으므로 그 시대에 어떤 일이 있었는지 시대별 트렌드 데이터 분석을 통해 확인할 수 있습니다.

깃허브 주소

안성표 : <https://github.com/aspyo/2023-2-BigDataProgramming-Project>

이재혁 : <https://github.com/LEEJaeHyeok97/2023-2-BigDataProgramming-Project>