**Pusan National University Computer Science and Engineering Technical Report 2023-10**

**음악 추천 모델의 음악 데이터 전처리 성능 분석**



진현

임우영

김영수

지도교수 채흥석

목 차

[1. 서론 1](#_Toc148694024)

[1.1. 연구 동기 1](#_Toc148694025)

[1.2. 연구 목표 1](#_Toc148694026)

[1.3. 배경지식 2](#_Toc148694027)

[1.3.1. NLP(Natural Language Processing) 2](#_Toc148694028)

[1.3.2. 유사도 측정 방식 3](#_Toc148694029)

[1.3.3. 추천 알고리즘 3](#_Toc148694030)

[1.4. 입력 데이터셋 4](#_Toc148694031)

[2. 연구 내용 5](#_Toc148694032)

[2.1. 음악 추천 모델 구축 5](#_Toc148694033)

[2.2. 전처리 방식 5](#_Toc148694034)

[2.2.1. 태그 임베딩 벡터 생성 방식 6](#_Toc148694035)

[2.2.2. 태그 자연어 처리 7](#_Toc148694036)

[2.2.3. 태그 불균형 처리 8](#_Toc148694037)

[2.2.4. 장르 임베딩 벡터 생성 방식 9](#_Toc148694038)

[2.2.5. 장르 불균형 처리 11](#_Toc148694039)

[2.3. 모델 다양화 12](#_Toc148694040)

[2.4. 웹 페이지 구현 13](#_Toc148694041)

[3. 실험 과정 및 결과 17](#_Toc148694042)

[3.1. 실험 설계 및 과정 17](#_Toc148694043)

[3.2. 실험 결과 21](#_Toc148694044)

[3.2.1. 태그 자연어 처리에 따른 성능 비교 21](#_Toc148694045)

[3.2.2. 태그 불균형 처리에 따른 성능 비교 22](#_Toc148694046)

[3.2.3. 장르 임베딩 벡터 생성 방식에 따른 성능 비교 26](#_Toc148694047)

[3.2.4. 장르 불균형 처리에 따른 성능 비교 27](#_Toc148694048)

[3.2.5. 최적 전처리 적용 결과 29](#_Toc148694049)

[4. 결론 및 향후 연구 방향 30](#_Toc148694050)

[4.1. 결론 30](#_Toc148694051)

[4.2. 향후 연구 방향 31](#_Toc148694052)

[5. 구성원 별 역할 및 개발 일정 31](#_Toc148694053)

[5.1. 구성원 별 역할 31](#_Toc148694054)

[5.2. 개발 일정 32](#_Toc148694055)

[6. 참고 문헌 32](#_Toc148694056)

# 서론

## 연구 동기

수많은 데이터를 수집하고 가공하여 새로운 정보의 가치를 얻어내는 빅 데이터 시대를 뛰어넘어, 현재는 이러한 데이터를 기반으로 한 추천 시스템, 스마트 자동차와 같은 머신러닝 서비스 기술이 보편화되고 있다. 특히 특정 사용자가 관심을 가질만한 정보를 추천하는 추천 시스템은, 넷플릭스의 온라인 스트리밍 서비스나 멜론의 음악 스트리밍 서비스 등에서 필수적인 역할을 하고 있다. 하지만 낮은 추천 시스템의 성능은 사용자의 취향이 충분히 반영되지 않은 컨텐츠 제공을 유발할 수 있으며, 이는 사용자의 서비스 만족도 감소로 이어질 수 있다. 이러한 추천 시스템의 성능은 학습 데이터의 품질에 크게 의존하며, 데이터 전처리를 통해 머신 러닝 모델의 성능을 향상시킬 수 있다. 따라서 본 과제는 음악 추천 시스템을 대상으로 다양한 데이터 전처리 기법을 적용하고 성능을 비교하려 한다.

## 연구 목표

본 졸업과제는 음악 추천 모델의 성능을 향상시키기 위한 효과적인 전처리 방법을 분석한다. 음악 추천을 위한 플레이리스트 및 음악 데이터에 다양한 전처리를 적용하고, 각 전처리 방식에 따른 추천 시스템의 성능을 비교한다. 플레이리스트 및 음악 데이터의 태그 및 장르는 자연어 처리, 불균형 데이터 처리 등 다양한 전처리를 통해 각 전처리 조합 별 데이터셋을 생성한다. 이후 이러한 태그 및 장르 텍스트 데이터를 기계가 이해하고 처리할 수 있는 벡터 형태로 변환한다. 본 과제의 음악 추천 모델은 앞서 생성한 두 벡터 값의 유사도를 계산하여 가장 유사한 10곡의 음악을 추천하고, 비지도 학습 평가 지표를 통해 모델의 성능을 평가한다. 이 과정을 각 전처리 조합 및 유사도 측정 방식에 따라 서로 다른 조건으로 수행하여 그 결과 및 모델 성능을 비교한다. 이를 통해 음악 데이터 전처리 별 효율성을 비교하고 음악 추천 모델의 성능을 보편적으로 향상시킬 수 있는 전처리 방식을 찾아내는 것이 본 과제의 최종 목표이다.

## 배경지식

### NLP(Natural Language Processing)

NLP는 컴퓨터가 인간의 언어를 해석 및 이해할 수 있도록 하는 머신러닝 기술이다. NLP 모델은 텍스트 토큰화, 어휘 구축, 임베딩 벡터화와 같은 과정을 거쳐 구축된다.

NLP 모델 구축 과정에서 텍스트 토큰화는 주어진 텍스트를 단어 단위로 분할하는 역할을 수행한다. 또한 어휘 구축은 이렇게 분할된 단어들로 어휘 집합을 만들어, NLP 모델이 자연어를 학습하는데 필요한 언어 정보의 기반을 만드는 역할을 수행한다.

특히 임베딩 벡터화 과정은 위에서 구축된 어휘 집합을 사용하여 각 단어를 고정된 길이의 실수 벡터로 변환하는데, 이 때 벡터화 방식에 따라 NLP 모델의 출력이 결정된다. 이러한 임베딩 벡터화 방식으로는 One-Hot Encoding, CountVectorizer, TF-IDF, Word2Vec 과 같은 대표적인 알고리즘이 존재한다[1]. 표 1은 대표적인 임베딩 벡터화 방식을 설명한다.

|  |  |
| --- | --- |
| 임베딩 벡터화 방식 | 설명 |
| One-Hot Encoding | * 범주의 개수만큼 크기를 가지는 벡터를 만들어 표현하고 싶은 인덱스에는 1, 그 외 인덱스에는 0을 부여하는 벡터 표현 방식 |
| CountVectorizer | * 단어의 빈도 정보를 활용해 텍스트 데이터를 수치화하는 방법 * 각 문서를 단어의 등장 횟수로 표현 |
| TF-IDF  (Term Frequency - Inverse Document Frequency) | * CountVectorizer와 마찬가지로 단어의 빈도 정보를 활용해 텍스트 데이터를 수치화하는 방법 * 모든 문서에서 자주 등장하는 단어일수록 중요도를 낮게 부여하고, 이러한 상대적 중요도를 고려해 각 문서를 벡터로 표현 |
| Word2Vec | * 단어 간의 의미적 유사성을 반영하는 임베딩 벡터를 생성하는 방법 * 대규모 텍스트 데이터에서 단어의 의미를 학습하고 각 단어를 벡터 형태로 표현 |

*표 1. 임베딩 벡터화 방식*

### 유사도 측정 방식

벡터 간 관련성을 수치화하기 위해서는 각 벡터의 유사도를 계산하는 과정이 필요하다. 벡터 간 유사도를 계산하기 위한 방식으로는 대표적으로 코사인 유사도[2], 피어슨 유사도[3]가 존재한다. 표 2는 대표적인 유사도 측정 방식을 설명한다.

|  |  |
| --- | --- |
| 유사도 측정 방식 | 설명 |
| 코사인 유사도 | * 두 벡터 간 각도를 측정해 유사도를 계산하는 방법 * 1에 가까울수록 두 아이템이 유사하고, -1에 가까울수록 두 아이템이 유사하지 않음을 의미 |
| 피어슨 유사도 | * 두 벡터 간의 선형 관계를 측정하여 얼마나 유사한 패턴을 가지는지를 파악하는 방법 * 1과 -1에 가까울수록 각각 양과 음의 선형 관계를 가지고, 0에 가까울수록 선형 관계가 없음을 의미 |

*표 2. 유사도 측정 방식*

라인, 디자인이(가) 표시된 사진

자동 생성된 설명

*그림 1. 코사인 유사도*

### 추천 알고리즘

데이터의 Feature를 활용하는 방식에 따라, 추천 시스템의 설계 및 과정이 달라질 수 있다. 이를 추천 알고리즘이라고 하며, 표 3은 대표적인 추천 알고리즘[4]을 설명한다.

|  |  |
| --- | --- |
| 추천 알고리즘 | 설명 |
| Content Based Filtering  (CBF) | * 사용자가 소비한 아이템과 비슷한 아이템을 추천하는 방식 * 아이템 유사도를 비교하기 위해, Feature를 수치화하는 전처리 작업이 필요 |
| Collaborative Filtering  (CF) | * 유사한 성향을 가진 사용자는 같은 아이템을 선호할 것이라는 가정하에, 다른 사용자가 소비한 아이템을 추천하는 방식 * 아이템에 대한 평점 데이터가 추가적으로 필요 |
| Hybrid Filtering  (HF) | * Collaborative Filtering과 Content Based Filtering을 조합한 방식 |

*표 3. 추천 알고리즘*

## 입력 데이터셋

음악 추천 모델을 생성하기 위한 데이터로는 2020년 kakao arena 3회 대회[5]에서 제공된 Melon Playlist 의 데이터셋을 사용한다.

|  |  |  |
| --- | --- | --- |
| 데이터셋 | 속성 | 설명 |
| 플레이리스트 | Id | 플레이리스트 ID |
| plylist\_title | 플레이리스트 제목 |
| Tags | 태그 리스트 |
| Songs | 곡 리스트 |
| like\_cnt | 좋아요 개수 |
| updt\_date | 수정 날짜 |
| 곡 | \_id | 곡 ID |
| album\_id | 앨범 ID |
| artist\_id\_basket | 아티스트 ID 리스트 |
| artist\_name\_basket | 아티스트 리스트 |
| song\_name | 곡 제목 |
| song\_gn\_gnr\_basket | 곡 장르 리스트 |
| song\_gn\_dtl\_gnr\_basket | 곡 세부 장르 리스트 |
| issue\_date | 발매일 |

*표 4. Melon Playlist 데이터셋*

# 연구 내용

연구 내용은 음악 추천 모델 구축, 전처리 방식, 모델 다양화와 웹 페이지 구현으로 구성된다. 음악 추천 모델 구축은 추천 모델에 사용할 알고리즘 및 주요 Feature의 선정 배경, 전처리 방식은 연구에서 사용할 전처리 활용 방법을 설명한다. 모델 다양화는 연구 결과를 일반화하기 위해 사용한 4가지 모델의 구조, 웹 페이지 구현은 음악 추천 기능을 구현한 웹 페이지의 상세 기능을 설명한다.

## 음악 추천 모델 구축

추천 시스템에서 대표적으로 사용되는 알고리즘은 사용자가 소비한 아이템과 비슷한 아이템을 추천하는 방식인 Content Based Filtering(CBF), 사용자와 유사한 성향을 가진 다른 사용자가 소비한 아이템을 추천하는 방식인 Collaborative Filtering(CF), CBF와 CF를 결합한 방식인 Hybrid Filtering(HF)이 존재한다.

본 과제는 이 중 CBF[6]를 추천 시스템의 주요 알고리즘으로 선정했다. CF와 HF는 데이터의 Feature를 고려하지 않고 단지 플레이리스트 내 음악의 존재 여부로 아이템을 추천한다. 이와 달리 CBF는 데이터의 Feature로 유사도를 계산해 음악을 추천하며, 이때 데이터 Feature를 사용함으로써 추천 결과에 근거를 제시할 수 있다는 장점이 존재한다. 따라서 데이터 Feature에 적절한 전처리 과정을 수행하면 추천한 아이템의 신뢰도를 높일 수 있다고 판단하여 이에 부합하는 알고리즘인 CBF를 본 과제에서 사용하게 되었다.

CBF 알고리즘으로 설계된 모델 중 2020년 kakao arena 3회 대회의 공개 리더보드 5위팀의 모델[7]을 기반으로 다양한 모델을 설계할 수 있다고 판단하여 이 모델을 참고해 모델을 설계했다. 이 모델의 주요 Feature로는 태그와 장르가 있으며, 태그는 사용자의 취향을, 장르는 음악적 특성을 반영하여 사용자와 음악의 관계를 찾아내 사용자에 맞는 음악을 추천할 수 있다. 이러한 점에서 본 과제는 태그, 장르를 음악 추천 모델의 주요 Feature로 선정했다.

## 전처리 방식

전처리 방식은 태그 임베딩 벡터 생성 방식, 태그 자연어 처리, 태그 불균형 처리, 장르 임베딩 벡터 생성 방식과 장르 불균형 처리로 구성된다. 태그 임베딩 벡터 생성 방식, 태그 자연어 처리와 태그 불균형 처리는 태그 데이터의 전처리 과정이며 장르 임베딩 벡터 생성 방식, 장르 불균형 처리는 장르 데이터의 전처리 과정이다.

### 태그 임베딩 벡터 생성 방식

임베딩 벡터는 텍스트 데이터를 기계가 이해하고 처리할 수 있도록 수치화하여 생성한 데이터로 음악 간 유사도 계산에 활용할 수 있다. 태그 데이터는 텍스트로 이루어져 있어 임베딩 벡터로 변환하여 사용해야 한다. 그림 2는 태그 데이터의 임베딩 벡터 생성 과정을 표현한다.

텍스트, 도표, 폰트, 스케치이(가) 표시된 사진

자동 생성된 설명

*그림 2. 태그 데이터의 임베딩 벡터 생성 과정*

태그 데이터는 완전한 자연어로 구성되어 있으므로, 임베딩 벡터화 방법 중 자연어 간의 의미적 유사성을 반영하는 Word2Vec을 통해 데이터를 변환한다. 표 5는 Word2Vec의 주요 매개변수[8]를 설명한다.

|  |  |
| --- | --- |
| 매개변수 | 설명 |
| vector\_size | * 벡터의 크기(설정 값 : 100) |
| window | * 학습시 앞 뒤로 고려하는 단어의 개수 (설정 값 : 5) |
| min\_count | * 학습에 사용되기 위한 최소 빈도수 (설정 값 : 15) |
| workers | * 모델을 만들 때의 스레드 개수 (설정 값 : 4) |
| sg | * 예측 방법 설정 (설정 값 : 1) |

*표 5. Word2Vec에 사용되는 매개변수*

vector\_size, window, workers의 설정 값은 Word2Vec 사용시 일반적으로 선택하는 값을 적용했다. min\_count는 다양한 값을 적용해 실험하여 그 중 가장 유의미한 성능 차이가 나타난 값인 15로 설정했다. sg는 0 일 경우 주변 단어를 통해 해당 단어를 예측하는 CBOW, 1 일 경우 해당 단어를 통해 주변 단어를 예측하는 skip-gram 방식을 적용한다. skip-gram 의 경우가 CBOW 의 경우보다 보편적 성능이 뛰어난 것으로 알려져 있어 sg 값은 1 로 설정했다.

### 태그 자연어 처리

태그 자연어 처리는 자연어에서 나타나는 동의어와 불용어와 같은 문제를 처리하는 과정이다. 태그 자연어 처리는 한국어 전처리 또는 영어 전처리를 통해 수행할 수 있다. 그림 3은 태그의 자연어 전처리 방식 별 변환 결과를 표현한다.

**텍스트, 폰트, 도표, 번호이(가) 표시된 사진

자동 생성된 설명**

*그림 3. 같은 의미를 가진 태그의 자연어 전처리 방식 별 변환 결과*

플레이리스트에 적용된 태그는 사용자의 임의 설정이 가능하기 때문에, 태그를 학습하고 벡터값으로 변환하는 Word2Vec에서 ['사랑', 'love', ‘Love’]를 서로 다른 단어로 간주하여 학습한다. 이로 인해 태그 유사도 측정 시 같은 의미를 가진 단어 간에도 유사도가 낮은 문제가 발생한다. 이를 해결하기 위해 태그 데이터에 대한 자연어 전처리 작업이 필요하다.

본 과제는 태그 데이터 전처리 방식으로 한국어 전처리, 영어 전처리 두 방식을 사용했다. 한국어 전처리 방식으로는 형태소 분석을 통해 명사, 동사, 형용사의 어간 추출을 진행했다[9]. 영어 전처리 방식으로는 기존 태그들을 googletrans API 를 통해 번역한 후, 소문자로 변환했다. 또한 'is', 'not' 과 같은 불용어를 제거 후, 한국어와 마찬가지로 형태소 분석을 통해 태그들의 어간 추출을 진행했다[10]**.**

### 태그 불균형 처리

태그 불균형 처리는 등장 빈도가 적은 태그를 가진 음악일수록 높은 가중치를 부여하는 과정이다. 그림 4는 태그 불균형 처리의 과정을 표현한다.

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

*그림 4. 태그 불균형 데이터 가중치 처리 과정*

Word2Vec 는 같은 단어가 데이터 세트 내에서 더 자주 나타날 경우 다른 단어와의 유사도가 증가하며, 반대로 해당 단어가 적게 나타날 경우 다른 단어와의 유사도가 감소하는 문제가 발생할 수 있다[11]. 예를 들어, '사랑' 이라는 단어는 태그 목록에서 자주 등장하지만 '연민' 이라는 단어는 태그 목록에서 드물게 등장하는 경우, 사용자 플레이리스트에 '연민' 태그가 존재하더라도 오히려 '사랑' 태그가 포함된 음악이 추천될 수 있다. 이를 해결하기 위해 태그 데이터에 대한 불균형 전처리 작업이 필요하다.

본 과제는 불균형 전처리 작업을 위해 TF-IDF를 참고하여 데이터의 등장 횟수에 따라 가중치를 부여했다.

해당 가중치의 계산 공식은 다음과 같다.

*수식 1. 최대치 기반 태그 가중치 부여 방식*

또한 '가장 많이 등장한 데이터의 개수'를 '샘플 데이터에 존재하는 음악 개수'로 바꾸는 공식 역시 적용했다.

*수식 2. 전체 개수 기반 태그 가중치 부여 방식*

### 장르 임베딩 벡터 생성 방식

장르 데이터 또한 텍스트로 이루어져 있어 유사도 계산을 위해 임베딩 벡터로 변환해야 한다. 태그 데이터와 달리 장르 데이터는 카테고리가 정해져 있는 범주형 데이터로, 이러한 데이터를 벡터화하는 대표적인 기법으로 One-hot Encoding이 알려져 있다.

One-hot Encoding은 범주형 데이터를 수치형 데이터가 담긴 벡터로 변환하는 기법으로, 한 데이터가 여러 개의 클래스에 속하더라도 결과 벡터에 독립적으로 표현할 수 있다는 장점이 있다.

본 과제는 이러한 One-hot Encoding의 원리를 기반으로 하면서 임베딩 벡터 생성에 가장 널리 쓰이는 기법인 CountVectorizer와 TF-IDF를 선택했다. CountVectorizer와 TfidfVectorizer라는 두 모듈은 각각 CountVectorizer와 TF-IDF를 구현한 파이썬의 모듈이며 이를 이용해 각 음악이 가진 장르 데이터를 임베딩 벡터로 변환한다. 그림 5는 장르 데이터의 임베딩 벡터 생성 과정을 표현한다.

**텍스트, 도표, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

*그림 5. 장르 데이터의 임베딩 벡터 생성 과정*

CountVectorizer는 음악에서 장르의 빈도수로 음악의 장르 행렬을 생성하고 임베딩 벡터를 생성한다. 벡터 생성 시 복잡한 계산 과정이 없어 결과를 빠르게 도출할 수 있지만 클래스 불균형이 존재할 경우 정확도가 떨어진다는 단점이 있다.

TF-IDF는 음악의 장르 빈도수인 TF에, 특정 장르가 전체 음악에서 얼마나 희소하게 등장하는지 나타내는 값인 IDF를 곱해 행렬을 생성한다. IDF 값은 해당 장르가 여러 음악에서 나타날수록 낮아지고 반대의 경우 높아진다. IDF 값의 계산 과정은 다음과 같다.

*수식 3. IDF 값 계산 공식*

TF-IDF는 장르의 빈도와 전체 음악 내에서의 희귀도를 함께 고려하므로 일반적으로 CountVectorizer에 비해 정확도가 높다고 알려져 있지만 계산 과정이 복잡해 대규모 데이터에서 실행 속도가 매우 느려진다는 단점이 있다.

### 장르 불균형 처리

장르 불균형 처리는 등장 빈도가 적은 장르를 가진 음악일수록 높은 가중치를 부여하는 과정이다. 그림 6은 장르 불균형 처리의 과정을 표현한다.

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

*그림 6. 장르 불균형 데이터 가중치 처리 과정*

플레이리스트를 구성하는 음악들은 최소 한 가지 이상의 장르를 포함한다. 플레이리스트는 사용자가 선호하는 음악들을 담고 있으며, 그 결과 일부 장르는 다른 장르에 비해 더 많은 음악을 포함하고 있다. 이로 인해 장르 간 데이터 불균형이 발생하고 추천 시스템에서 주류 장르의 음악이 지나치게 강조될 수 있다. 이러한 문제를 해결하기 위해 장르 데이터에 불균형 전처리가 필요하다.

불균형 전처리의 대표적인 방식으로는 소수 데이터의 양을 늘리는 오버샘플링과 다수 데이터의 양을 줄이는 언더샘플링이 있다. 그러나 이러한 방법을 사용하면 데이터가 정확히 표현되지 않고 왜곡되어 생성되거나 중요한 정보를 가진 데이터가 손실될 수 있다. 예를 들어, 오버샘플링 과정에서 '락' 장르 음악의 양이 부족한 경우, 임의의 음악에 '락' 장르를 할당하여 새로운 데이터를 생성하면 '락'과 관련 없는 음악에도 '락' 장르가 할당될 수 있어 음악의 장르적 특징이 왜곡될 우려가 있다. 또 다른 예로, 언더샘플링 과정에서 특정 장르가 음악에 지나치게 포함되어 있는 경우 해당 장르를 포함한 음악을 일부 제외한다. 이 때 음악 추천을 위해 사용되는 플레이리스트 내 음악이 제외되며 장르 데이터가 변경될 수 있고 이에 따라 기존 플레이리스트의 특징을 나타내던 중요한 정보가 손실될 수 있는 문제가 발생한다[12].

따라서 본 과제는 이러한 불균형 전처리 방식이 적합하지 않았다. 대신, 데이터의 등장 빈도를 고려하여 각 장르에 가중치를 할당하는 방식을 사용했다. 가중치의 계산 공식은 다음과 같다.

*수식 4. 최대치 기반 장르 가중치 부여 방식*

태그 불균형과 동일하게 '가장 많이 등장한 데이터의 개수'를 '샘플 데이터에 존재하는 음악 개수'로 바꾸는 공식 역시 적용했다.

*수식 5. 전체 개수 기반 장르 가중치 부여 방식*

## 모델 다양화

전처리 방식의 효과는 추천 모델의 로직에 따라 달라질 수 있으므로 단일 모델을 사용하는 것은 결과를 일반화하기 어렵다는 문제가 있어, 다양한 모델을 구축할 필요성이 있다.

모델 다양화를 위해 Feature 의 우선 순위와 유사도 측정 방식 두 가지 기준을 적용했다. Feature 의 우선 순위는 태그 또는 장르 중 어느 유사도를 먼저 고려할지에 따라 2가지로 나눌 수 있고, 유사도 측정 방식은 코사인 혹은 피어슨 유사도 2가지로 나눌 수 있다. 표 6은 최종적으로 생성된 모델 4가지의 음악 추천 과정을 설명한다.

|  |  |  |
| --- | --- | --- |
| 태그 우선  코사인 측정 모델 | Feature 우선순위 | 태그 유사도 > 장르 유사도 |
| 유사도 측정 방식 | 코사인 유사도 |
| 태그 우선  피어슨 측정 모델 | Feature 우선순위 | 태그 유사도 > 장르 유사도 |
| 유사도 측정 방식 | 피어슨 유사도 |
| 장르 우선  코사인 측정 모델 | Feature 우선순위 | 장르 유사도 > 태그 유사도 |
| 유사도 측정 방식 | 코사인 유사도 |
| 장르 우선  피어슨 측정 모델 | Feature 우선순위 | 장르 유사도 > 태그 유사도 |
| 유사도 측정 방식 | 피어슨 유사도 |

*표 6. 각 모델의 음악 추천 과정*

## 웹 페이지 구현

전처리 방식 별 유사도의 비교를 용이하게 하기 위해 음악 추천 웹 페이지를 구현했다. 그림 7은 음악 추천 웹 페이지의 초기 화면이다.

텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명

*그림 7. 음악 추천 웹 페이지 초기 화면*

음악 추천 웹 페이지는 음악 검색, 태그 추가와 음악 추천 영역으로 구성된다. 음악 검색은 이름을 검색해 음악을 추가하는 영역이다. 태그 추가는 태그를 입력해 현재 플레이리스트에 태그를 직접 추가하는 영역이다. 음악 추천은 선택한 전처리 방식과 현재 플레이리스트 데이터에 따라 추천된 음악 목록이 출력되는 영역이다. 그림 8은 음악 이름을 입력한 화면이다.

텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명

*그림 8. 검색 화면 1*

사용자는 음악 이름 칸에 검색하고 싶은 음악을 입력할 수 있다. 그림 9는 음악 검색 결과가 나타난 화면이다.

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

자동 생성된 설명

*그림 9. 검색 화면 2*

검색 결과에는 음악의 이름, 아티스트, 발매일과 ID의 내용이 출력되며 선택 칸을 통해 음악을 선택할 수 있다. 사용자는 음악을 선택한 뒤 ‘선택한 음악 추가’ 버튼을 눌러 음악 추천 영역의 ‘추가된 음악 목록’에 추가할 수 있다. 그림 10은 음악이 목록에 추가된 화면이다.

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

자동 생성된 설명

*그림 10. 음악 추가 후 화면*

추가된 음악은 이름, 태그와 장르를 출력하며 사용자는 ‘음악 목록 초기화’ 버튼을 눌러 음악 목록을 초기화 할 수 있다. 그림 11은 추가할 태그를 입력한 화면이다.

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

*그림 11. 태그 추가 화면*

사용자는 태그 이름 칸에 추가하고 싶은 단어를 입력할 수 있다. 그림 12는 태그가 목록에 추가된 화면이다.

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

자동 생성된 설명

*그림 12. 태그 추가 후 화면*

추가된 태그는 목록에 출력되며 ‘추가된 음악 목록’의 태그에도 추가된다. 사용자는 ‘태그 목록 초기화’ 버튼을 눌러 태그 목록을 초기화 할 수 있다. 그림 13은 전처리 방식을 선택하는 화면이다.

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

*그림 13. 전처리 선택 화면*

사용자는 전처리 방식을 복수 선택할 수 있으나 한국어 전처리와 영어 전처리는 중복 선택할 수 없다. ‘음악 추천’ 버튼을 누르면 웹 페이지는 사용자가 설정한 음악과 태그 목록, 전처리 방식을 토대로 음악 추천 목록을 출력한다. 그림 14는 음악 추천 결과가 나타난 화면이다.

텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명

*그림 14. 음악 추천 결과 화면*

추천 결과에는 음악의 이름, ID, 태그와 장르 유사도의 내용이 출력되며 추가로 적용된 전처리 방식과 추천 결과에 대한 정확도가 출력된다. 추천 결과에 대한 정확도는 비지도 학습 모델의 평가 지표 중 한 플레이리스트를 입력으로 고려한 추천 정확성 평가 방법 인 Precision@K로 표시했다.

# 실험 과정 및 결과

## 실험 설계 및 과정

본 과제의 실험은 데이터에 적용된 전처리에 따라 발생하는 성능 변화 비교 및 그에 따른 가장 최적화된 전처리 조합을 찾는 것을 목표로 설정한다. 실험 과정은 크게 데이터 샘플링, 데이터 전처리, 음악 추천 및 평가로 구성된다.

그림 15는 본 과제의 전체 실험 과정을 나타낸 흐름도이다.

텍스트, 도표, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

*그림 15. 전체 실험 과정 흐름도*

데이터 샘플링 단계에서는 전체 플레이리스트 중 무작위로 300개가 포함된 샘플 데이터를 생성한다. 각 샘플 데이터는 1개의 입력 플레이리스트와 299개의 학습용 플레이리스트로 설정되어 음악 추천 모델을 거쳐 추천 결과를 도출하고, 이후 입력 플레이리스트를 바꿔가며 모든 플레이리스트가 각각 1번씩 입력으로 사용될 때까지 동일한 과정을 총 300번 반복 수행한다. 또한 실험 재현성 검증 및 결과 일관성 확인을 위해 샘플 데이터를 총 10개 생성하여 동일한 실험을 진행한다.

데이터 전처리 단계에서는 본 과제에서 제안한 전처리 방식들을 토대로 만들 수 있는 모든 전처리 조합을 모두 적용한다. 전처리 조합에는 CountVectorizer 또는 TF-IDF를 적용하는 2가지의 장르 임베딩 벡터 생성 방식, 전처리를 적용하지 않거나 최대치 기반 가중치 또는 전체 개수 기반 가중치를 적용하는 3가지의 태그 불균형 처리 방식, 전처리를 적용하지 않거나 한국어 전처리 또는 영어 번역 후 전처리를 적용하는 3가지의 태그 자연어 처리 방식, 그리고 전처리를 적용하지 않거나 최대치 기반 가중치 또는 전체 개수 기반 가중치를 적용하는 3가지의 장르 불균형 처리 방식이 활용된다. 전처리가 완료된 데이터는 모델의 추천 알고리즘에 따라 추천 결과를 출력한다.

음악 추천 및 평가 단계에서는 추천 결과에 대해 평가 지표를 적용해 모델의 성능을 평가 및 비교한다. 추천 모델의 경우 라벨링 되어 있지 않은 데이터 입력으로부터 패턴을 찾아 적절한 출력을 만들어내기 때문에, F1-Score, ROC-Curve 등 기존의 분류 모델 평가를 위한 지표를 사용할 수 없다. 따라서 Precision@K, MAP, nDCG 등의 비지도 학습 모델 평가 지표를 사용해 추천 모델의 성능을 검증한다. 표 7은 대표적인 비지도 학습 모델 평가 지표를 설명한다[13].

|  |  |
| --- | --- |
| 평가 지표 | 설명 |
| Precision@K | * 모델에서 추천한 음악 K개 중, 실제 플레이리스트에 속하는 음악의 개수 * 한 플레이리스트를 입력으로 고려한 추천 정확성 평가 방식 |
| Mean Average Precision@K | * 전체 플레이리스트를 입력으로 고려한 Precision@K 기반의 평균 추천 정확성 평가 방식 * Relevance() : 플레이리스트와 음악 사이의 관련성으로, 해당 음악이 실제로 플레이리스트에 속하면 1, 그렇지 않으면 0 값을 가짐 * Average Precision@K : 모델에서 추천한 음악 K 개 중, 특정 인덱스의 Precision 값 ( ) * Mean Average Precision@K : 모든 플레이리스트에 대한 AP@K 의 평균 ( ) * 결과값은 0~1 사이로 나타나며, 결과값이 클수록 추천 모델의 성능이 좋음을 의미함 |
| Normalized Discounted Cumulative Gain@K | * 전체 플레이리스트를 입력으로 고려한 Relevance기반의 추천 정확성 평가 방식 * Relevance와 추천된 음악의 순서를 함께 고려하여 추천시스템을 평가하는 방식 * 모든 Relevance가 동일하다고 가정할 때, 먼저 등장한 음악에 대해서는 높은 점수를, 나중에 등장한 음악에 대해서는 낮은 점수를 부여 * Cumulative Gain(CG) : 추천한 음악의 Relevance 합 * Discounted Cumulative Gain(DCG) : 추천 목록에서 후순위에 위치할수록 높은 패널티를 부여한 Relevance 합 ( ) * Ideal DCG(IDCG) : Relevance 값이 내림차순으로 정렬되어 있는 상태에서의 DCG 값 * Normalized DCG : DCG에 정규화를 적용한 것으로, 이상적인 추천 조합 대비 현재 모델의 추천 리스트가 얼마나 좋은지를 나타내는 지표 ( ) * 결과값은 0~1 사이로 나타나며, 결과값이 클수록 추천 모델의 성능이 좋음을 의미함 |

*표 7. 추천 시스템 평가 지표*

모든 실험이 완료되면 각 모델 및 전처리 별 MAP와 nDCG의 평균값을 구한 후 어떤 전처리에서 보편적인 성능 향상이 발생하는지 분석을 진행한다.

## 실험 결과

실험 결과는 태그 자연어 처리, 태그 불균형 처리, 장르 임베딩 벡터 생성 방식, 장르 불균형 처리 각각에 대해 전처리 방식에 따른 성능을 비교하고, 최종적으로 가장 좋은 효과를 보였던 전처리 조합을 설명한다.

### 태그 자연어 처리에 따른 성능 비교

태그 자연어 처리에 따른 4가지 모델에 대한 MAP, nDCG 성능을 비교한 결과이다. 표 8, 9는 태그 데이터에 대해 자연어 처리를 하지 않은 경우와 한국어 전처리, 영어 번역 후 전처리를 적용한 경우에 따라 각 추천 모델의 성능을 MAP와 nDCG 지표로 나타낸다. 그림 16은 각 방식 별 전처리 이후 평균 성능 변화율을 시각적으로 나타낸다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 태그 우선  코사인 측정  모델 | 태그 우선  피어슨 측정  모델 | 장르 우선  코사인 측정  모델 | 장르 우선  피어슨 측정  모델 |
| 전처리 미적용 | 0.3023 | 0.0519 | 0.5054 | 0.0488 |
| 한국어 전처리 | 0.3056 | 0.0521 | 0.5077 | 0.0493 |
| 영어 전처리 | 0.3043 | 0.0523 | 0.5076 | 0.0493 |

*표 8. 자연어 전처리에 따른 MAP 비교 표*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 태그 우선  코사인 측정  모델 | 태그 우선  피어슨 측정  모델 | 장르 우선  코사인 측정  모델 | 장르 우선  피어슨 측정  모델 |
| 전처리 미적용 | | 0.5091 | 0.1328 | 0.6808 | 0.1068 |
| 한국어 전처리 | | 0.5134 | 0.1319 | 0.6848 | 0.1078 |
| 영어 전처리 | | 0.5118 | 0.1320 | 0.6840 | 0.1078 |

*표 9. 태그 자연어 처리 방식에 따른 nDCG 비교 표*

*그림 16. 태그 자연어 처리 방식에 따른 평균 성능 변화율 그래프*

자연어 처리를 진행하지 않은 경우와 비교했을 때, 한국어 전처리를 수행하면 평균 MAP는 약 0.738%, 평균 nDCG는 약 0.396%의 상승을 보였고, 영어 번역 후 전처리를 수행하면 평균 MAP는 약 0.741%, 평균 nDCG는 약 0.329% 상승을 보였다.

### 태그 불균형 처리에 따른 성능 비교

태그 불균형 처리에 따른 4가지 모델에 대한 MAP, nDCG 성능을 비교한 결과이다. 표 10, 11은 태그 데이터에 대해 불균형 처리를 하지 않은 경우와 최대치 기반, 전체 개수 기반 불균형 처리를 적용한 경우에 따라 각 추천 모델의 성능을 MAP와 nDCG 지표로 나타낸다. 그림 17은 각 방식 별 전처리 이후 평균 성능 변화율을 시각적으로 나타낸다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 태그 우선  코사인 측정  모델 | 태그 우선  피어슨 측정  모델 | 장르 우선  코사인 측정  모델 | 장르 우선  피어슨 측정  모델 |
| 전처리  미적용 | | 0.3101 | 0.0571 | 0.5096 | 0.0526 |
| 최대치 기반  가중치 적용 | | 0.2985 | 0.0471 | 0.5036 | 0.0457 |
| 전체 개수 기반  가중치 적용 | | 0.3035 | 0.0521 | 0.5076 | 0.0492 |

*표 10. 태그 불균형 전처리에 따른 MAP 비교 표*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 태그 우선  코사인 측정  모델 | 태그 우선  피어슨 측정  모델 | 장르 우선  코사인 측정  모델 | 장르 우선  피어슨 측정  모델 |
| 전처리  미적용 | 0.5184 | 0.1417 | 0.6869 | 0.1139 |
| 최대치 기반  가중치 적용 | 0.5045 | 0.1226 | 0.6787 | 0.1010 |
| 전체 개수 기반  가중치 적용 | 0.5113 | 0.1325 | 0.6840 | 0.1076 |

*표 11. 태그 불균형 전처리에 따른 nDCG 비교 표*

*그림 17. 태그 불균형 처리에 따른 평균 성능 변화율 그래프*

태그 불균형 처리를 진행하지 않은 경우와 비교했을 때, 최대치 기반 가중치로 태그 불균형을 처리한 경우 평균 MAP는 약 8.867%, 평균 nDCG는 약 7.157%의 하락을 보였다. 전체 개수 기반 가중치로 태그 불균형을 처리한 경우 평균 MAP은 약 4.392%, 평균 nDCG는 약 3.448%의 하락을 보였다.

불균형 처리 결과 오히려 성능 하락이 발생했다. 이에 대한 원인을 분석하고자 먼저 데이터의 불균형 정도를 확인하기 위해 실험에 사용한 10개의 샘플링 데이터에서 각 태그의 등장 횟수를 측정했다. 그림 18~20은 이 중 첫 번째 샘플링 데이터에서 가장 많이 등장한 3개와 가장 적게 등장한 3개의 태그 및 빈도를 그래프로 나타낸 것이다.

*그림 18. 첫 번째 테스트(전처리 미적용)에 사용된 모든 플레이리스트의 태그 카운트 결과*

*그림 19. 첫 번째 테스트(한국어 전처리)에 사용된 모든 플레이리스트의 태그 카운트 결과*

*그림 20. 첫 번째 테스트(영어 전처리)에 사용된 모든 플레이리스트의 태그 카운트 결과*

태그 데이터의 불균형 정도는 집합의 정보를 나타내는 지표로 활용되는 섀넌 엔트로피[14]를 이용해 판단했다. 섀넌 엔트로피는 결과가 0에 가까울수록 매우 균형적으로 분포된 데이터임을 의미하며, 임의의 임계값부터 사이에 존재할 경우 데이터가 무질서하게 혼합되어 있음을 의미한다. 이 때 임계값은 으로 설정했으며, 은 아이템의 가짓수이다. 섀넌 엔트로피의 계산식은 아래와 같다.

*수식 6. 섀넌 엔트로피 계산 공식*

측정된 태그 분포를 토대로 한 섀넌 엔트로피 계산 결과는 약 4.105로 임계값인 7.308과 최대 엔트로피인 9.134 사이에 속하지 않는다. 이는 데이터에 불균형이 충분하지 않음을 나타낸다. 나머지 9개 샘플의 태그 데이터 역시 섀넌 엔트로피 값이 임계값 미만으로 나타났다.

본 과제에서 제안한 태그 불균형 처리를 위한 가중치는 유사도 계산 결과에 직접 곱하는 연산을 수행하므로 데이터의 불균형 정도가 부족하다면 작은 가중치에도 매우 큰 결과 변화가 발생할 수 있다.

따라서 태그 불균형 전처리의 경우, 실험에 사용한 태그 데이터의 불균형이 충분하지 않아 불균형 처리를 적용한 결과 오히려 성능이 떨어진 것으로 추정할 수 있다.

### 장르 임베딩 벡터 생성 방식에 따른 성능 비교

장르 임베딩 벡터 생성 방식에 따른 4가지 모델에 대한 MAP, nDCG 성능을 비교한 결과이다. 표 12, 13은 장르 데이터의 임베딩 벡터 생성 방식에 CountVectorizer, TF-IDF를 적용한 경우에 따라 각 추천 모델의 성능을 MAP와 nDCG 지표로 나타낸다. 그림 21은 각 방식 별 전처리 이후 평균 성능 변화율을 시각적으로 나타낸다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 태그 우선  코사인 측정  모델 | 태그 우선  피어슨 측정  모델 | 장르 우선  코사인 측정  모델 | 장르 우선  피어슨 측정  모델 |
| CountVectorizer | 0.3056 | 0.0555 | 0.5051 | 0.0510 |
| TF-IDF | 0.3024 | 0.0487 | 0.5088 | 0.0472 |

*표 12. 장르 임베딩 방식에 따른 MAP 비교 표*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 태그 우선  코사인 측정  모델 | 태그 우선  피어슨 측정  모델 | 장르 우선  코사인 측정  모델 | 장르 우선  피어슨 측정  모델 |
| CountVectorizer | 0.5146 | 0.1394 | 0.6825 | 0.1109 |
| TF-IDF | 0.5083 | 0.1251 | 0.6839 | 0.1040 |

*표 13. 장르 임베딩 방식에 따른 nDCG 비교 표*

*그림 21. CountVectorizer 적용 시 TF-IDF 대비 평균 성능 변화율 그래프*

TF-IDF를 적용한 경우와 비교했을 때, CountVectorizer를 적용하면 평균 MAP는 약 5.574%, 평균 nDCG는 약 4.760%의 상승을 보였다.

전반적으로 CountVectorizer가 TF-IDF보다 높은 성능을 보였는데, 이는 본 과제에서 One-hot Encoding이 문장형 텍스트의 자연어 처리 목적보다는 장르 카테고리의 벡터화 목적으로 사용되었기 때문에 TF-IDF에 사용된 IDF 값이 성능 향상을 이끌어내기 어려웠던 것으로 짐작할 수 있다.

### 장르 불균형 처리에 따른 성능 비교

장르 불균형 처리에 따른 4가지 모델에 대한 MAP, nDCG 성능을 비교한 결과이다. 표 14, 15는 장르 데이터에 대해 불균형 처리를 하지 않은 경우와 최대치 기반, 전체 개수 기반 불균형 처리를 적용한 경우에 따라 각 추천 모델의 성능을 MAP와 nDCG 지표로 나타낸다. 그림 22는 각 방식 별 전처리 이후 평균 성능 변화율을 시각적으로 나타낸다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 태그 우선  코사인 측정  모델 | 태그 우선  피어슨 측정  모델 | 장르 우선  코사인 측정  모델 | 장르 우선  피어슨 측정  모델 |
| 전처리  미적용 | 0.2906 | 0.0450 | 0.4954 | 0.0444 |
| 최대치 기반  가중치 적용 | 0.3142 | 0.0581 | 0.5137 | 0.0532 |
| 빈도 기반  가중치 적용 | 0.3074 | 0.0533 | 0.5117 | 0.0498 |

*표 14. 장르 불균형 처리에 따른 MAP 비교 표*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 태그 우선  코사인 측정  모델 | 태그 우선  피어슨 측정  모델 | 장르 우선  코사인 측정  모델 | 장르 우선  피어슨 측정  모델 |
| 전처리  미적용 | 0.4960 | 0.1182 | 0.6733 | 0.1000 |
| 최대치 기반  가중치 적용 | 0.5235 | 0.1441 | 0.6897 | 0.1143 |
| 빈도 기반  가중치 적용 | 0.5148 | 0.1344 | 0.6866 | 0.1082 |

*표 15. 장르 불균형 처리에 따른 nDCG 비교 표*

*그림 22. 장르 불균형 처리에 따른 평균 성능 변화율 그래프*

장르 불균형 처리를 진행하지 않은 경우와 비교했을 때, 최대치 기반 가중치로 장르 불균형을 처리한 경우 평균 MAP는 약 15.147%, 평균 nDCG는 약 11.038%의 상승을 보였다. 전체 개수 기반 가중치로 장르 불균형을 처리한 경우 평균 MAP은 약 9.916%, 평균 nDCG는 약 6.910%의 상승을 보였다.

장르 데이터는 불균형 처리 결과 성능 향상을 볼 수 있었다. 이는 태그 데이터와 달리 불균형이 충분히 있었기 때문으로 추정된다. 장르 데이터의 불균형 정도를 확인하기 위해 태그 데이터와 같은 방식으로 실험에 사용한 10개의 샘플링 데이터에서 각 장르의 등장 횟수를 측정했다. 그림 23은 이 중 첫 번째 샘플링 데이터에서 가장 많이 등장한 3개와 가장 적게 등장한 3개의 장르 및 빈도를 그래프로 나타낸 것이다.

*그림 23. 첫 번째 테스트에 사용된 모든 플레이리스트의 장르 카운트 결과*

장르 데이터의 불균형 정도 역시 섀넌 엔트로피를 이용해 판단했다. 측정된 장르 분포를 토대로 한 섀넌 엔트로피 계산 결과는 약 6.011로 임계값인 6.006과 최대 엔트로피인 7.508 사이에 속한다. 이는 데이터에 불균형이 충분히 존재한다는 것을 의미한다. 나머지 9개 샘플의 장르 데이터 역시 섀넌 엔트로피 값이 임계값 이상으로 나타났다.

따라서 장르 데이터는 태그 데이터와 달리 불균형이 충분히 존재하여 불균형 처리가 의도한 효과를 낼 수 있었던 것으로 짐작할 수 있다.

### 최적 전처리 적용 결과

위의 실험 결과를 토대로 추천 시스템의 성능에 가장 큰 향상이 있었던 전처리 방식들을 조합하여 전처리를 적용하지 않은 경우와 성능을 비교했다. 사용한 전처리 방식은 태그 데이터에 대한 한국어 전처리, 전체 개수 기반 장르 불균형 전처리, CountVectorizer를 이용한 장르 임베딩 벡터 생성이다.

표 16은 아무런 전처리를 하지 않았을 때와 최적의 전처리 조합을 적용했을 때 4가지 모델의 평균 성능을 MAP와 nDCG 지표로 나타낸다. 그림 24는 최적 전처리 적용 이후 평균 성능 변화율을 시각적으로 나타낸다.

|  |  |  |
| --- | --- | --- |
|  | MAP | nDCG |
| 적용 이전 | 0.2450 | 0.2809 |
| 적용 이후 | 0.3963 | 0.4528 |

*표 16. 최적 전처리 전후 MAP / nDCG 비교 표*

*그림 24. 최적 전처리 적용 시 평균 성능 변화율 그래프*

아무런 전처리도 하지 않은 경우와 비교했을 때 평균 MAP는 약 14.643%, 평균 nDCG는 약 14.270%의 상승을 보였다.

# 결론 및 향후 연구 방향

## 결론

본 과제는 다양한 전처리를 적용한 음악 데이터를 CBF 방식으로 구축한 음악 추천 시스템의 입력으로 활용하여 각각의 성능을 평가 및 비교하는 연구를 수행했다. 또한 모델의 성능을 크게 좌우할 수 있는 데이터의 Feature를 찾아내고 자연어 전처리나 불균형 데이터 처리 등 적합한 전처리 기법을 학습하고 추천 모델에 적용했다. 데이터 전처리 외에도 음악 추천 모델을 개발하며 추천 알고리즘 및 평가 지표 알고리즘 구현과 같은 비지도 학습 과정 전체를 경험했다.

본 과제의 실험 결과에 따르면, 태그 데이터에 자연어 처리를 진행하지 않은 경우와 비교했을 때, 한국어 전처리를 수행하면 평균 MAP는 약 0.738%, 평균 nDCG는 약 0.396%의 상승을 보였고, 영어 번역 후 전처리를 수행하면 평균 MAP는 약 0.741%, 평균 nDCG는 약 0.329% 상승을 보였다.

균형 잡힌 데이터인 태그 데이터에 불균형 처리를 진행했을 때, 최대치 기반 가중치로 태그 불균형을 처리한 경우 평균 MAP는 약 8.867%, 평균 nDCG는 약 7.157%의 하락을 보였고, 전체 개수 기반 가중치로 태그 불균형을 처리한 경우 평균 MAP은 약 4.392%, 평균 nDCG는 약 3.448%의 하락을 보였다.

장르 데이터의 임베딩 벡터화 방식을 TF-IDF에서 CountVectorizer로 변경했을 때 평균 MAP는 약 5.574%, 평균 nDCG는 약 4.760%의 상승을 보였다.

균형 잡히지 않은 데이터인 장르 데이터에 불균형 처리를 진행했을 때, 최대치 기반 가중치로 장르 불균형을 처리한 경우 평균 MAP는 약 15.147%, 평균 nDCG는 약 11.038%의 상승을 보였고, 전체 개수 기반 가중치로 장르 불균형을 처리한 경우 평균 MAP은 약 9.916%, 평균 nDCG는 약 6.910%의 상승을 보였다.

최종적으로 음악 데이터에서 태그에 대한 한국어 전처리, 장르에 대한 전체 개수 기반 불균형 처리 및 임베딩 벡터 생성 방식을 CountVectorizer로 설정했을 때 추천 모델의 성능이 가장 높았음을 확인할 수 있었다. 위 전처리를 적용하면 아무런 전처리를 적용하지 않은 경우와 비교하여 평균 MAP는 약 14.643%, 평균 nDCG는 약 14.270%의 상승을 보였고, 그 외의 전처리 조합은 이와 비교해 성능이 떨어짐을 확인할 수 있었다.

이러한 연구 결과는 음악 데이터의 태그, 장르 Feature 를 사용하는 다른 모델 또는 서비스에서 태그에 대한 자연어 전처리나 장르에 대한 불균형 데이터 처리를 통해 서비스의 품질을 향상시킬 수 있는 가능성을 보여준다.

## 향후 연구 방향

본 과제에서는 시간과 자원의 문제로 총 10,000 개의 플레이리스트 데이터 중 300개를 랜덤 샘플링하여 평균 11,620개의 음악을 모델에서 학습했고, 이 과정을 10번 반복 수행했다. 본래 1000개의 플레이리스트 데이터를 매 반복마다 랜덤 샘플링하려 했지만, 각 단계에서 432개의 모델을 생성 및 평가하는데 약 4주의 시간이 걸려 샘플 수를 300개로 축소했다. 또한 음악 데이터의 태그, 장르만으로 추천 알고리즘을 구현해 음악의 작곡가나 발행연도 등 다른 Feature를 고려하지 못한 점도 존재한다.

따라서 기존 알고리즘에서 사용하던 태그, 장르 Feature 외에 모델의 성능에 좋은 영향을 끼칠 수 있는 추가적인 Feature 를 찾아 그에 적합한 전처리를 탐구해보고, 샘플링된 플레이리스트 데이터가 아닌 전체 플레이리스트 데이터를 활용해 추가 학습을 시도할 계획이다.

# 구성원 별 역할 및 개발 일정

## 구성원 별 역할

|  |  |
| --- | --- |
| 이름 | 역할 |
| 진현 | 음악 추천 페이지 추천 함수 모듈 구현  nDCG를 사용한 추천 모델 평가 구현  평가 결과 분석 및 시각화 |
| 임우영 | 음악 추천 페이지 로직 구현  태그 자연어 전처리, 불균형 처리  태그 벡터 평균을 통한 유사도 계산 구현  MAP@K 를 사용한 추천 모델 평가 구현 |
| 김영수 | 음악 추천 페이지 디자인  장르 불균형 처리, 임베딩 방식 처리  장르 유사도 계산 방식 변환 기능 구현 |
| 공통 | 추천 모델 성능 테스트  보고서 작성 및 시연 준비 |

## 개발 일정

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5월 | | | 6월 | | | | | 7월 | | | | | 8월 | | | | | 9월 | | | |
| 3주 | **4주** | **5주** | **1주** | **2주** | **3주** | **4주** | **5주** | **1주** | **2주** | **3주** | **4주** | **5주** | **1주** | **2주** | **3주** | **4주** | **5주** | **1주** | **2주** | **3주** | **4주** |
| 착수보고서 | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **데이터 전처리 기술 학습** | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | **데이터 분석 및 시각화** | | |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | **음악 추천 모델 학습** | | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  | **음악 추천 모델 구축** | | | |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  | **중간보고서** | |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  | **데이터 전처리** | | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  | **추천 모델 평가** | | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **결과 분석 및 최적화** | | |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **최종 테스트** | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **최종보고서 및 발표** | | |

# 참고 문헌

|  |
| --- |
| [1] Choonsik\_mom. (2022, July 01). 다양한 NLP Embedding 기법 [Online]. Available: <https://velog.io/@choonsik_mom/%EB%8B%A4%EC%96%91%ED%95%9C-NLP-Embedding-%EA%B8%B0%EB%B2%95.araboja>  [2] Wikidocs. (2022, November 14). Cosine Similarity [Online]. Available: <https://wikidocs.net/24603>  [3] Umbum. (2019, October 20). Pearson Correlation Coefficient [Online]. Available: <https://umbum.dev/1006/>  [4] Wikipedia. (2019, April). Recommender System [Online]. Available: <https://en.wikipedia.org/wiki/Recommender_system>  [5] Daniel.py. (2022, May 27). 카카오 AI추천 : 카카오 음악 추천을 경험해보고 싶다면? [Online]. Available: <https://tech.kakao.com/2022/05/27/melon-playlist-dataset/>  [6] Bell.park. (2021, December 27). 카카오 AI 추천 : 카카오의 콘텐츠 기반 필터링 [Online]. Available: <https://tech.kakao.com/2021/12/27/content-based-filtering-in-kakao/>  [7] MelonRec. (2020, July 28). Kakao arena 3rd Competition – Melon Playlist Continuation [Online]. Available: <https://github.com/jjun0127/MelonRec>  [8] Gensim. (2022, December 21). Word2Vec Documentation [Online]. Available: <https://radimrehurek.com/gensim/models/word2vec.html>  [9] KoNLPy. 파이썬 한국어 NLP [Online]. Available: <https://konlpy.org/ko/latest/index.html>  [10] Datascienceschool. Natural Language Toolkit Package [Online]. Available: <https://datascienceschool.net/03%20machine%20learning/03.01.01%20NLTK%20%EC%9E%90%EC%97%B0%EC%96%B4%20%EC%B2%98%EB%A6%AC%20%ED%8C%A8%ED%82%A4%EC%A7%80.html>  [11] Stackoverflow. (2021, March). Minimum number of words in the vocabulary for Word2Vec [Online]. Available: <https://stackoverflow.com/questions/66267818/minimum-number-of-words-in-the-vocabulary-for-word2vec-models>  [12] Cjkangme. (2023, February 25). 언더샘플링과 오버샘플링 [Online]. Available: <https://velog.io/@cjkangme/%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D-%EC%96%B8%EB%8D%94%EC%83%98%ED%94%8C%EB%A7%81%EA%B3%BC-%EC%98%A4%EB%B2%84%EC%83%98%ED%94%8C%EB%A7%81>  [13] Sungkee. (2021, September 01). 추천시스템 성능 평가 방법 [Online]. Available: <https://sungkee-book.tistory.com/11>  [14] Ethan. (2023, February 19). Entropy [Online]. Available:  <https://velog.io/@nochesita/%EC%A0%95%EB%B3%B4%EC%9D%B4%EB%A1%A0-%EC%97%94%ED%8A%B8%EB%A1%9C%ED%94%BC-Entropy> |