목차

1. 과제 목표 및 기존 과제에서의 변경사항

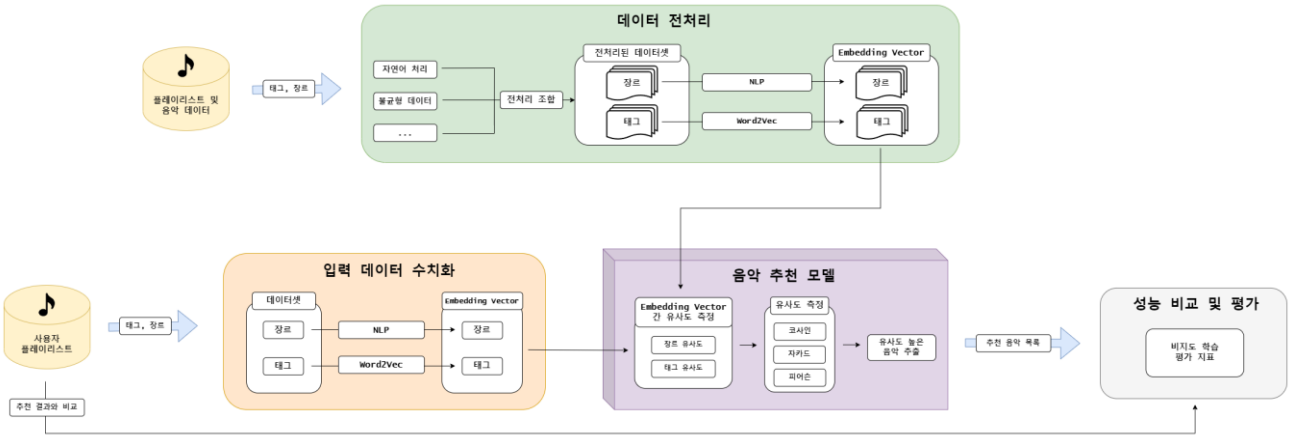

Figure 1. 과제 전체 구성도

1.1 과제 목표

본 졸업과제는 음악 추천에 사용되는 플레이리스트 및 음악 데이터에 대해 가장 효과적인 전처리 방법에 관해 연구한다. 플레이리스트 및 음악 데이터의 태그 및 장르는 자연어 처리, 불균형 데이터 처리 등 다양한 전처리 기법을 통해 각 전처리 조합 별 데이터셋을 생성한 후 이를 Embedding Vector 로 변환한다. 사용자 플레이리스트의 태그 및 장르 또한 데이터 전처리 과정에서 생성된 Embedding Vector 와의 유사도 계산을 위해 Embedding Vector 로 수치화한다. 음악 추천 모델은 앞서 생성한 두 Embedding Vector 의 유사도를 계산하여 가장 유사한 10 곡의 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 추천 모형의 성능을 보편적으로 향상시킬 수 있는 방법을 찾아내는 것이 본 과제의 최종 목표이다.

1.2 기존 과제에서의 변경사항

이슈	변경사항
추천 시스템의 경우 비지도 학습 방식을 사용하기 때문에, 기존에 사용하려던 지도 학습 방식의 분류 모델을 적용하기에는 적합하지 않음.	추천 시스템의 기본 유형인 Content Based Filtering(CBF), Collaborative Filtering(CF), 그리고 Hybrid Filtering(HF) 방식을 이용해 다양한 모델을 생성함.
CF 모델의 경우 고차원 데이터로 인한 Memory Error 를 유발해 모델 학습이 불가능하고, 일부 데이터를 샘플링해 사용할 경우 정확도가 매우 낮아짐.	이용할 수 있는 Feature 가 가장 많고, 학습 속도를 높일 수 있는 CBF 로 모델 개발 기법을 통일함.
추천 시스템의 각 유형마다 사용하는 Feature 가 다르고, 이에 따라 생성된 모든 모델에 동일한 데이터 전처리를 적용하기 어려움.	
주요 Feature 로 태그, 장르만을 사용하여 차원 축소의 필요성이 낮고, 각 Feature 에서 결측치는 존재하지 않음.	기존에 사용하려던 전처리 과정에서 차원 축소 및 결측치 관련 전처리는 배제함.

Table 1. 기존 과제에서의 변경사항

2. 설계 상세화 및 변경 내역

2.1 추천 알고리즘 선정

추천 시스템이란 특정 사용자가 선호할 것이라고 예상되는 아이템을 추천하는 시스템이다. 이러한 시스템에서 대표적으로 사용되는 알고리즘은 다음과 같다.

첫 번째로 Content Based Filtering 이다. CBF 는 사용자가 선호했던 아이템들을 기반으로 비슷한 유형의 아이템을 추천하는 방식으로, 텍스트·이미지 데이터 내에서 아이템을 표현할 수 있는 Feature 를 추출하여 사용자가 과거 소비했던 아이템과 Feature 가 비슷한 아이템을 추천한다.

두 번째로 Collaborative Filtering 이다. CF 는 '특정 아이템에 대한 선호도가 유사한 사용자는 다른 아이템에 대해서도 선호도가 비슷할 것이다' 라는 가정하에, 사용자가 평가한 아이템과 비슷한 선호도를 가진, 다른 사용자가 선택한 아이템을 추천하는 방식이다.

마지막으로 Hybrid Filtering 이다. HF 는 CBF 와 CF 를 결합한 모델로, CF 를 이용해 유사한 사용자 간의 아이템 기록을 비교 후, CBF 를 적용시켜 해당 사용자가 높게 평가한 아이템의 특징을 포함하는 아이템을 추천하는 방식이 대표적이다.

추천 알고리즘	설명
Content Based Filtering (CBF)	● 사용자가 소비한 아이템과 비슷한 아이템을 추천하는 방식 ● 아이템 유사도를 비교하기 위해, Feature 를 수치화하는 전처리 작업이 필요
Collaborative Filtering (CF)	● 유사한 성향을 가진 사용자는 같은 아이템을 선호할 것이라는 가정하에, 다른 사용자가 소비한 아이템을 추천 ● 아이템에 대한 평점 데이터가 추가적으로 필요
Hybrid Filtering (HF)	● Collaborative Filtering 과 Content Based Filtering 을 조합한 방법

Table 2. 추천 알고리즘의 종류

위와 같은 알고리즘을 기반으로 3 가지의 서로 다른 추천 모델을 생성하고 각 성능을 비교하려 하였으나, Filtering 방식에 따라 사용할 수 있는 Feature 가 달랐고, 이에 따라 전처리 방식을 통일시킬 수 없었다. 또한 CF 와 HF 은 데이터의 Feature 를 고려하지 않고 단지 플레이리스트 내에 존재하는 음악의 종류와 평점만으로 아이템을 추천을 하는 방법으로, 이는 다양한 전처리 방법을 Feature 에 적용해 음악 데이터에 가장 효과적인 전처리 방법에 대한 정보를 찾는 본 과제의 목표와는 맞지 않았다. 따라서 데이터 Feature 의 전처리가 비교적 자유로운 CBF 로 모델을 설계하기로 했다.

2.2 음악 추천 과정

본 과제에서 사용할 CBF 방식 음악 추천 모델의 형태는 아래와 같다.

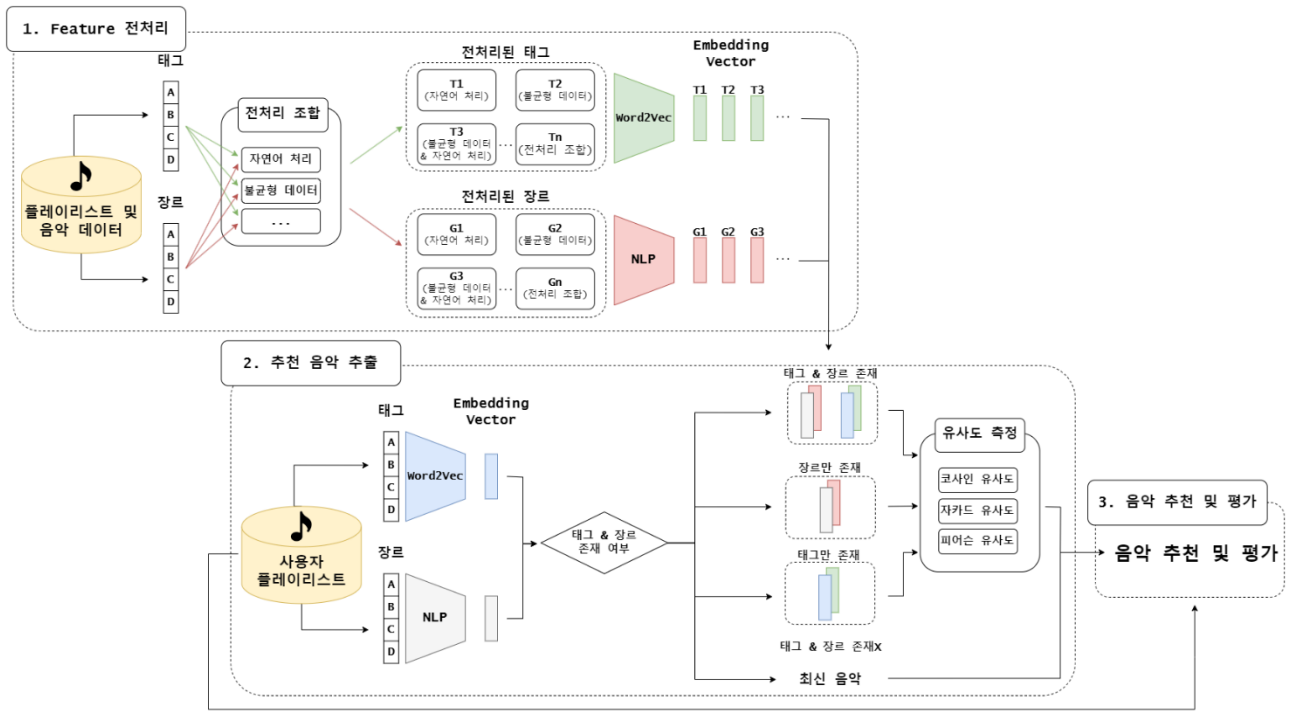

Figure 2. 음악 추천 모델 흐름도

2.2.1 Feature 전처리

모델 학습에 사용되는 Feature 는 음악의 특성을 나타낼 수 있고, 다른 음악과 유사도를 계산하기 용이한 정보인 태그와 장르로 설정하였다. 태그·장르 데이터는 자연어 형태로 존재하기 때문에, 단어들을 Embedding Vector 로 변환한다. Embedding Vector 는 자연어를 기계가 이해할 수 있는 벡터값으로 수치화한 것으로, 여기서는 음악들 간의 유사도를 계산하기 위해 사용한다.

현재 음악 추천 모델은 태그·장르를 단순히 Embedding Vector 로 변환 후 사용자 플레이리스트의 음악들과 유사도가 높은 음악을 추천한다. 본 과제에서는 다양한 전처리 방법을 적용한 음악 데이터들에 따라 추천 모형의 성능을 비교하는 것이 목표이므로, 이를 확장시키기 위한 추가적인 전처리 방법으로는 아래와 같이 고려하고 있다.

주요 Feature	전처리 방안	내용
태그	불균형 데이터 처리	특정 태그와 관련한 곡 개수의 불균형 정도에 따라, 유사도에 가중치를 부여
	텍스트 전처리	영어로 된 태그의 대소문자 통일
		특수 기호 제거

		Stopword 제거	불필요하게 자주 발생하는 단어 제거
		Stemming 제거	동일한 의미이지만, 표현 방식이 다양한 경우를 제거
장르	불균형 데이터 처리	특정 장르와 관련한 곡 개수의 불균형 정도에 따라, 유사도에 가중치를 부여	
	Embedding	CountVectorizer	단어가 나타나는 횟수를 각 단어에 부여된 인덱스에 표현
		TF-IDF	모든 데이터에서 전반적으로 자주 등장하는 단어에 대해 페널티를 부여
		벡터 평균	Embedding 된 플레이리스트 곡들의 벡터 평균
공통	가중치	플레이리스트 평점에 따른 가중치 부여	

Table 3. Feature 전처리 방안

태그 데이터를 Embedding Vector 로 변환하는 작업에는 Word2Vec 방법을 사용했다. Word2Vec 는 단어를 벡터값으로 변환하는 Word Embedding 중 가장 대표적인 방식으로, 음악 추천 시 입력된 태그와 유사한 다른 태그를 유추하는 작업에 활용했다.

장르 데이터를 Embedding Vector 로 변환하는 작업에는 CountVectorizer 를 사용했다. CountVectorizer 는 한 문장 안에서의 단어의 등장 빈도수를 One-hot Encoding 방식으로 나타내는데, One-hot Encoding 은 범주의 개수만큼 크기를 가지는 벡터를 만들어 표현하고 싶은 인덱스에는 1, 다른 인덱스에는 0 을 부여하는 벡터 표현 방식이다. 장르 데이터의 경우 자연어로 표현되어 있지만, 그 종류가 제한적인 항목의 형태로 나타나 범주형 데이터로 활용할 수 있어, CountVectorizer 를 활용해 장르 데이터를 One-hot Encoding 방식으로 변환했다.

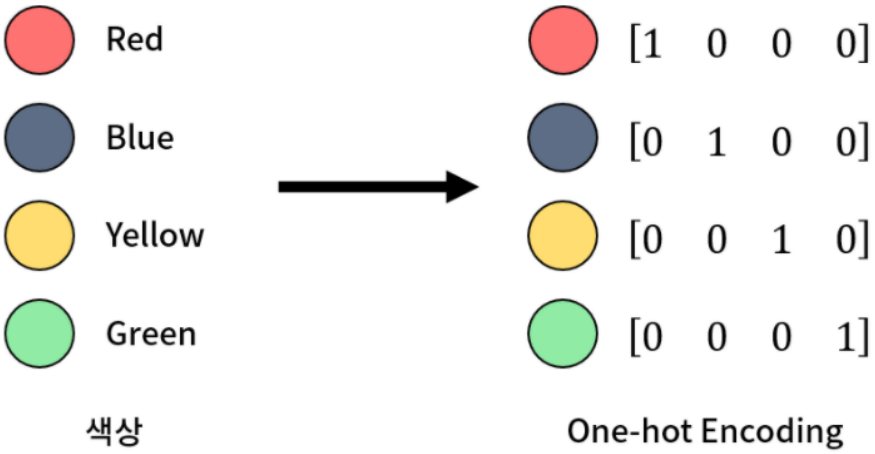

Figure 3. One-hot Encoding

2.2.2 추천 음악 추출

CBF 은 사용자의 소비 이력을 토대로 새로운 아이템을 추천하는 방법이다. 이러한 소비 이력 데이터를 확보하기 위해, 입력 플레이리스트를 사용자가 소비한 아이템으로 가정하여 사용했다. 이 때, 입력 플레이리스트 내 음악의 태그·장르 유무에 따라 모델의 출력을 다르게 하였다.

존재 유무		**장르**	
		있음	**없음**
태그	**있음**	태그·장르를 기준으로 유사도가 가장 높은 음악 추천	태그를 기준으로 유사도가 가장 높은 음악 추천
	없음	장르를 기준으로 유사도가 가장 높은 음악 추천	최신 음악(2018~2023 년)를 추천

Table 4. Feature 존재 유무에 따른 모델의 출력

또한 주요 Feature 로 선정한 태그·장르 그룹에 따라 소비한 아이템과 유사한 음악을 추출했다.

장르 그룹의 경우, 각 음악의 장르 Embedding Vector 에 대해 유사도를 계산해 유사도 행렬을 생성했다. 벡터 간 유사도를 계산하기 위한 방법은 대표적으로 코사인, 자카드 그리고 피어슨 유사도가 있으며, 실제 동작하는 방식은 다음과 같다.

유사도 측정 방식	**설명**
코사인 유사도	벡터 간 각도를 계산하여 유사성을 측정

자카드 유사도	합집합과 교집합 사이의 비율
피어슨 유사도	각 벡터의 표본평균으로 정규화를 진행한 후, 코사인 유사도를 측정

Table 5. 유사도 측정 방식

특히 코사인 유사도는 두 벡터 사이의 각도를 측정해 유사도를 계산하는 방법으로, 1 에 가까울수록 두 아이템이 유사하고, -1 에 가까울수록 두 아이템이 유사하지 않다. 또한 코사인 유사도는 벡터의 방향만으로 유사도를 측정하기 때문에 장르 데이터의 개수가 상이해도 유사도를 공정하게 측정할 수 있다는 장점이 있어, 샘플링된 데이터를 활용하는 현재 단계에서는 해당 유사도 측정 방식을 채택하여 사용했다. 이렇게 계산한 유사도 행렬에서 유사도가 높은 음악 순으로 음악을 추출한다. 이때, 입력 플레이리스트에서 가장 오래된 음악보다 이전에 출시된 음악은 포함하지 않는다.

Figure 4. 코사인 유사도

앞서 언급한 3 가지 유사도 측정 방식을 모두 활용하여 각기 다른 추천 모델을 구축하고, 이후 동일한 전처리가 적용된 음악 데이터를 입력으로 사용하여 모델들의 성능을 비교할 계획이다. 이를 통해 다양한 추천 모델에서 전처리 효과의 보편성을 입증할 계획이다.

태그 그룹의 경우, 음악 내 태그 개수가 3 개 미만일 때 Word2Vec 를 통해 만들어진 Embedding Vector 에서 유사도가 높은 다른 태그를 추가하고, 해당 태그를 가진 음악의 등장 빈도가 높은 순서대로 음악을 추출한다.

2.2.3 음악 추천 및 평가

2.2.2 에서 추출한 태그·장르 그룹 모두에 속하는 음악을 10 개 선택하고, 선택한 음악이 10 개 미만인 경우 각 그룹에서 유사도 순위에 따라 음악을 추가한다. 이렇게 구한 10 개의 음악을 실제 사용자가

선호하는지 판단하기 위한 평가 지표가 필요하다. 추천 모델의 경우 라벨링 되어 있지 않은 데이터 입력으로부터 패턴을 찾아 적절한 출력을 만들어내기 때문에, 본래 사용하려던 F1-Score, ROC-Curve 등 분류 모델 평가를 위한 지표를 사용할 수 없다. 따라서 비지도 학습 모델의 평가 지표 중에서도 추천 모델의 성능을 검증하는데 대표적인 방법을 적용하기로 했다. 본 과제에서 사용할 모델 평가 방법들은 다음과 같다.

평가 방법	설명
Precision@K	• 모델에서 추천한 음악 K 개 중, 실제 플레이리스트에 속하는 음악의 개수
Recall@K	• 실제 플레이리스트에 속하는 음악 중, 모델에서 추천한 음악이 포함되는 개수
Mean Average Precision@K	• Average Precision@K : 모델에서 추천한 음악 K 개 중, 특정 인덱스의 Precision 값 ($AP@K = \frac{1}{m}\sum_{i=1}^{K} Precision@i \cdot rel(i)$) • Mean Average Precision@K : 모든 플레이리스트에 대한 AP@K 의 평균 ($MAP@K = \frac{1}{\lvert Playlist \rvert}\sum_{p=1}^{\lvert Playlist \rvert}(AP@K)_p$) • 결과값은 0~1 사이로 나타나며, 결과값이 클수록 추천 모델의 성능이 좋음을 의미함
Normalized Discounted Cumulative Gain@K	• Relevance : 플레이리스트와 특정 음악이 얼마나 관련이 있는지 나타내는 값으로, 임의로 설정이 필요 • Cumulative Gain : 추천한 음악의 Relevance 합 • Discounted Cumulative Gain : CG 에서 추천한 음악 순서에 따라 가중치를 부여 ($DCG_K \sum_{i=1}^{K} \frac{rel_i}{\log_2(i+1)}$) • Ideal DCG : 최선의 추천을 했을 때의 DCG 값 • Normalized DCG : DCG 에 정규화를 적용한 것으로, 이상적인 추천 조합 대비 현재 모델의 추천 리스트가 얼마나 좋은지를 나타내는 지표 ($NDCG_K = \frac{DCG}{IDCG}$) • 결과값은 0~1 사이로 나타나며, 결과값이 클수록 추천 모델의 성능이 좋음을 의미함

Table 6. 모델 평가 지표

MAP@K 에서 m 은 플레이리스트 내 음악의 개수이고, $rel(i)$ 은 relevence 로 해당 음악이 실제로 플레이리스트에 속하면 1, 그렇지 않으면 0 값을 가진다.

2.3 음악 추천 모델 구현

2.3.1 Feature 전처리

Table 7 과 9~10 은 플레이리스트 데이터에서 태그와 장르 데이터를 Embedding Vector 로 변환하는 코드이다.

```
from gensim.models.word2vec import Word2Vec

# 태그를 Word2Vec 에 학습시킨다
w2v = Word2Vec(sentences = train_data_sample2['tags'], vector_size = 100,
               window = 5, min_count = 5, workers = 4, sg = 1)
```

Table 7. Tag to Embedding Vector

Table 7 에서 태그 데이터는 Word2Vec 를 통해 태그 간 유사도를 학습시켰다. 이때 설정한 매개변수는 Table 8 과 같다.

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 : 5)
Workers	● 모델을 만들 때의 스레드 개수 (설정 값 : 4)
sg	● 예측 방법 설정 (설정 값 : 1)

Table 8. Word2Vec 에 사용되는 매개변수

Vector_size 는 생성할 Embedding Vector 의 크기를 설정한다. Window 는 한 단어를 학습할 때 앞뒤로 고려하는 단어의 개수를 설정한다. 값이 커지면 단어의 의미적 정확도가 올라가지만 그만큼 계산량이 증가한다. Min_count 는 단어가 학습에 사용되기 위한 최소 빈도수로 어떤 단어가 이 값보다

적게 등장하였을 경우 그 단어는 학습에서 배제한다. 값이 작을수록 더 많은 단어들이 계산에 포함되어 계산량이 증가한다. Workers 는 Word2Vec 모델을 만들 때 생성되는 스레드의 개수를 설정하는데 스레드의 개수가 많아질수록 모델이 만들어지는 속도가 증가하지만 그만큼 CPU 의 사용량도 증가한다. Sg 는 만들어진 모델이 단어를 예측하는 방법을 설정한다. Sg 값이 0 일 경우 주변 단어를 통해 해당 단어를 예측하는 CBOW, 1 일 경우 해당 단어를 통해 주변 단어를 예측하는 skip-gram 방법으로 적용된다. Skip-gram 의 경우가 CBOW 의 경우보다 보편적 성능이 뛰어난 것이 검증되어 있어 Sg 값을 1 로 설정했다.

```
from sklearn.feature_extraction.text import CountVectorizer

# 리스트의 형태로 저장된 장르 데이터를 하나의 문장으로 만든다
song_tag_appended['gnr_literal'] = song_tag_appended['gnr'].apply(lambda x : (' ').join(x))

# CounterVectorizer 를 통해 장르 데이터를 벡터화한다
count_vect = CountVectorizer()
gnr_mat = count_vect.fit_transform(song_tag_appended['gnr_literal'])
```

Table 9. Genre to Embedding Vector

Table 9 에서 각 음악의 장르들을 하나의 문장으로 합쳐 CountVectorizer 를 통해 모든 곡을 벡터화한 후 이들을 모아 희소 행렬로 변환했다.

```
from sklearn.metrics.pairwise import cosine_similarity

# 벡터들의 유사도를 서로 측정하여 행렬로 저장한다
gnr_sim = cosine_similarity(gnr_mat, gnr_mat)
```

Table 10. 코사인 유사도 행렬 생성

Table 10 에서 Table 9 으로부터 얻은 희소 행렬을 사용해 코사인 유사도를 측정하고 그 결과를 유사도 행렬로 저장하였다. 이 때 매개변수로 동일한 행렬을 넣었는데 이는 두 벡터 간의 유사도를 측정한 것이 아닌 모든 벡터 간의 유사도를 측정해 행렬로 저장한 것이다.

2.3.2 그룹 별 추천 음악 추출

Table 11~12 는 사용자의 태그와 장르 데이터를 사용해 음악을 추출하는 코드이다.

```
ts = tags

# 태그가 존재할 경우 + 태그의 개수가 3 개 미만인 경우 w2v 로 태그를 3 개까지 늘린다
all_tags = []
if len(ts) != 0 and len(ts) < 3:
    for tag in ts:
        sim_tags = w2v.wv.most_similar(tag)
        for t in sim_tags:
            all_tags.append(t)
    all_tags = sorted(all_tags, key = lambda x : -x[1])
    i = 0
    while len(ts) != 0 and len(ts) < 3:
        tag = all_tags[i][0]
        if tag not in ts:
            ts.append(tag)
            i += 1

# 해당 태그가 존재하는 플레이리스트의 음악을 추출하고 등장 빈도수로 정렬한다
tag_songs = dict()

for tag in ts:
    for i in range(len(tag_df['song_id'])):
        if tag in tag_df['tags'][i]:

            for ss in tag_df['song_id'][i]:
                if not ss in songs:
```

```
            if ss in tag_songs:
                tag_songs[ss] += 1

            else:
                tag_songs[ss] = 1

tag_songs = sorted(tag_songs.items(), key=lambda x: x[1], reverse=True)
```

Table 11. 태그 데이터를 활용한 음악 그룹 생성

Table 11 에서 사용자의 태그 데이터를 이용해 음악을 추출한다. 태그의 개수가 부족하면 학습된 Word2Vec 를 사용하여 태그의 개수를 늘리고, 해당 태그를 가진 플레이리스트의 음악을 추출하여 등장 빈도 순으로 정렬해 저장한다. 만일 태그 데이터가 존재하지 않는 경우 음악을 추출하지 않는다.

```
def find_sim_song(df, sim_matrix, songs, top_n=10):
    simi = np.zeros(len(df['song_id']))
    minyear = 3000

    # 입력 플레이리스트의 음악들 중 가장 오래된 음악의 발행 년도를 저장한다
    for song in songs:
        title_song = df[df['song_id'] == song]
        minyear = min(minyear, title_song['issue_date'].values[0]//10000)

    # 유사도 행렬을 이용해 모든 음악에 플레이리스트의 음악 간 유사도를 저장한다
    for song in songs:
        title_song = df[df['song_id'] == song]
        title_index = title_song.index.values

        simi = simi + sim_matrix[title_index, :]

    simi /= len(songs)

    df['similarity'] = simi.reshape(-1, 1)
    temp = df.sort_values(by="similarity", ascending=False)
```

```
    # 입력 플레이리스트 안의 음악과 저장한 발행 년도 이전의 음악을 제외하고
    # 유사도가 높은 순으로 정렬하여 상위 n 개를 추출한다
    for song in songs:
        title_song = df[df['song_id'] == song]
        title_index = title_song.index.values

        temp = temp[temp.index.values != title_index]

    temp = temp[temp['issue_date'] > minyear*10000]

    final_index = temp.index.values[ : top_n]

    return df.iloc[final_index]

# 기존 음악이 있는 경우 장르 유사도를 계산해
#상위 100 개의 음악을 찾아낸다
if len(songs) > 0:
    simi_songs = find_sim_song(song_df, sim_mat, songs, 100)

# 기존 음악이 없는 경우 최신 음악(2018~2023 년도)를 찾아낸다
else:
    simi_songs = song_df
    simi_songs = simi_songs[simi_songs['issue_date'] > 20180000]
    simi_songs = simi_songs[simi_songs['issue_date'] < 20240000]
```

Table 12. 플레이리스트 내 장르 데이터를 활용한 음악 그룹 생성

Table 12 에서 사용자의 장르 데이터를 이용해 음악을 추출한다. 장르 데이터가 존재하는 경우 유사도 행렬에서 유사도가 가장 높은 음악을 저장한다. 이 때, 플레이리스트에서 가장 오래된 음악보다 이전에 출시된 음악은 포함하지 않는다. 만일 장르 데이터가 없는 경우 최신 음악을 저장한다.

2.3.3 음악 추천

```
# 태그와 장르 유사도로 각각 만들어낸 음악 목록
# 둘 모두에 존재하는 음악을 우선적으로 10 개 추출한다
recommended = []
index = 0

while len(recommended) < 10 and index < len(tag_songs):
    tag_song = tag_songs[index][0]

    if tag_song in simi_songs:
        recommended.append(tag_song)

    index += 1

# 둘 모두에 존재하는 음악이 10 개 미만인 경우
# 각각에서 우선순위가 높은 음악들을 추출한다
if len(recommended) < 10:

    # 태그와 장르 두 그룹에서 동일한 개수만큼 음악 추출
    # sc = 장르 그룹에서 추출할 음악의 개수
    if len(recommended) % 2 == 0:
        sc = (10-len(recommended)) / 2
    else:
        sc = (10-len(recommended)) // 2

     # 태그는 있고 장르가 없는 경우 태그 그룹에서 10 곡을 추출
    if len(songs) == 0:
        sc = 0

    # 이미 추출한 음악을 제외하고 태그 그룹에서 정해진 개수만큼 추출한다
    # 태그가 없을 경우 동작하지 않음
    index = 0
    while len(tag_songs) != 0 and len(recommended) < (10 - sc):
        tag_song = tag_songs[index][0]

        if not tag_song in recommended:
            recommended.append(tag_song)

        index += 1
```

```
    # 이미 추출한 음악을 제외하고 추천 음악이 10 개가 될 때까지
    # 장르 그룹에서 추출한다
    index = 0
    while len(recommended) < 10:
        simi_song = simi_songs['song_id'].values[index]

        if not simi_song in recommended:
            recommended.append(simi_song)

        index += 1
```

Table 13. 음악 추천 함수

Table 13 에서 두 그룹으로부터 최종적으로 추천할 음악을 결정한다. 두 그룹 모두에 존재하는 음악을 10 곡까지 우선적으로 추출하고, 부족할 경우 각 그룹에서 유사도가 높은 순서대로 음악을 추출한다. 만약 태그나 장르 데이터 중 하나가 없는 경우 다른 한 그룹에서 음악을 추출하고, 둘 다 없는 경우 최신 음악을 저장한 simi_songs 그룹에서 음악을 추출한다.

2.4 음악 추천 결과

구축된 모델의 출력은 입력 플레이리스트 내 음악의 태그·장르 유무에 따라 다르며 이를 확인하기 위해 아래 Table 14 의 태그·장르 데이터를 이용한다.

```
my_tags = ['락']
my_songs = [525514, 129701, 229622]
```

Table 14. 입력 태그·장르 데이터

2.4.1 태그 및 장르 데이터가 존재할 경우

태그와 장르 데이터가 모두 존재하는 경우, 태그와 장르 유사도로 만들어낸 음악 목록에 함께 존재하는 음악을 추출한다. 이 때, 추천된 음악 개수가 10 개 미만인 경우 태그와 장르 각각에서 유사도가 높은 순서대로 음악을 추가적으로 추출한다.

	song_id	song_name	artist_name_basket	issue_date	tags
697	146989	YOUTH	[Troye Sivan]	20160123	[Pop, HipHop, 휴식, 그루브, 감성, 비오는날, 나만알고싶은, 비트, 팝...
698	430106	Everglow	[Coldplay]	20151204	[느껴져, Pop, HipHop, Maroon, 펑크락, 가을, 스트레스, 휴식, ...
706	15124	Something Just Like This	[The Chainsmokers, Coldplay]	20170407	[Pop, 휴식, 감성, 밤, 나만알고싶은, 팝송, 드라이브, 라디오, 느낌있는, ...
2112	258814	Surrender (Feat. 린)	[챈슬러 (Chancellor)]	20161130	[Pop, 사랑, HipHop, 이별, 인디, 추억, 새벽, 달달, 우울, 감성, ...
2114	86380	오늘도 그대만 (Feat. 정동원)	[T.P RETRO (타디스 프로젝트롭)]	20170228	[사랑, 인생곡들만, 이별, 인디, 추억, 새벽, 여름밤, 우울, 감성, 발라드, ...
11005	191430	Sing	[Ed Sheeran]	20140407	[힐링, 휴식, 잔잔한]
15445	463782	Dreamin` Slow	[Mac Demarco]	20170227	[휴가, 분위기, 휴식, 감성적인, 책읽을때, 혼자, 잔잔한, 추억, 주말]
9	205238	Dingue, Dingue, Dingue	[Christophe Mae]	20110124	[락]
10498	701978	Meant To Be	[Rachel Belman]	20101011	[힐링, 카페, 봄]
16038	513731	The Sun The Trees (Acoustic Ver.)	[Russian Red]	20111018	[팝]

Figure 5. 태그·장르 데이터 존재 시 결과

2.4.2 장르 데이터만 존재하는 경우

장르 데이터만 존재하는 경우, 장르 유사도가 높은 순서대로 음악을 추출한다.

	song_id	song_name	artist_name_basket	issue_date	tags
11005	191430	Sing	[Ed Sheeran]	20140407	[힐링, 휴식, 잔잔한]
15445	463782	Dreamin` Slow	[Mac Demarco]	20170227	[휴가, 분위기, 휴식, 감성적인, 책읽을때, 혼자, 잔잔한, 추억, 주말]
9	205238	Dingue, Dingue, Dingue	[Christophe Mae]	20110124	[락]
10498	701978	Meant To Be	[Rachel Belman]	20101011	[힐링, 카페, 봄]
16038	513731	The Sun The Trees (Acoustic Ver.)	[Russian Red]	20111018	[팝]
16037	489449	Lego House (Acoustic)	[Ed Sheeran]	20111111	[팝]
9997	392798	Fuerteventura	[Russian Red]	20111018	[나만의Best3, 피쉬슈즈, 인디, indie]
9450	598147	Take Your Time	[Louis Yoelin]	20131105	[힐링, 휴식, 기분전환]
10908	661924	Familiar	[Agnes Obel]	20161021	[기분좋은, 아침, 출근, 상쾌, 팝송]
695	310974	9 Crimes	[Damien Rice]	20061107	[Pop, 뉴에이지, 재즈, 매장음악, 휴식, 까페, 감성, 잔잔한, 겨울, 피아노...

Figure 6. 장르 데이터만 존재 시 결과

2.4.3 태그 데이터만 존재하는 경우

태그 데이터만 존재하는 경우, 태그 유사도가 높은 순서대로 음악을 추출한다.

	song_id	song_name	artist_name_basket	issue_date	tags
697	146989	YOUTH	[Troye Sivan]	20160123	[Pop, HipHop, 휴식, 그루브, 감성, 비오는날, 나만알고싶은, 비트, 팝...
698	430106	Everglow	[Coldplay]	20151204	[느껴져, Pop, HipHop, Maroon, 펑크락, 가을, 스트레스, 휴식, ...
706	15124	Something Just Like This	[The Chainsmokers, Coldplay]	20170407	[Pop, 휴식, 감성, 밤, 나만알고싶은, 팝송, 드라이브, 라디오, 느낌있는, ...
2112	258814	Surrender (Feat. 린)	[챈슬러 (Chancellor)]	20161130	[Pop, 사랑, HipHop, 이별, 인디, 추억, 새벽, 달달, 우울, 감성, ...
2114	86380	오늘도 그대만 (Feat. 정동원)	[T.P RETRO (타디스 프로젝트 팀)]	20170228	[사랑, 인생곡들만, 이별, 인디, 추억, 새벽, 여름밤, 우울, 감성, 발라드, ...
89	394031	Into the Unknown (From "Frozen 2"/Soun...	[Idina Menzel, Aurora]	20191115	[크리스마스캐럴, 인디음악, 분위기, 눈오는날, 연말, 캐럴, 겨울노래, 스트레스,...
119	457519	꿀차	[우효]	20180102	[크리스마스캐럴, 분위기, 눈오는날, 연말, 캐럴, 겨울노래, 크리스마스송, 크리스...
120	453762	너 정말 예쁘다	[최낙타]	20180410	[크리스마스캐럴, 분위기, 눈오는날, 연말, 캐럴, 겨울노래, 크리스마스송, 크리스...
121	349398	LOVE YA!	[혁오 (HYUKOH)]	20180531	[크리스마스캐럴, 분위기, 눈오는날, 연말, 캐럴, 겨울노래, 크리스마스송, 크리스...
122	631142	편지	[장희원]	20180603	[크리스마스캐럴, 따뜻한, 분위기, 사랑, 눈오는날, 연말, 캐럴, 겨울노래, 크리...

Figure 7. 태그 데이터만 존재 시 결과

2.4.4 태그 및 장르 데이터 모두 존재하지 않는 경우

태그와 장르 데이터가 모두 존재하지 않는 경우 최신 음악 순서대로 음악을 추출한다.

	song_id	song_name	artist_name_basket	issue_date	tags
89	394031	Into the Unknown (From "Frozen 2"/Soun...	[Idina Menzel, Aurora]	20191115	[크리스마스캐럴, 인디음악, 분위기, 눈오는날, 연말, 캐럴, 겨울노래, 스트레스,...
119	457519	꿀차	[우효]	20180102	[크리스마스캐럴, 분위기, 눈오는날, 연말, 캐럴, 겨울노래, 크리스마스송, 크리스...
120	453762	너 정말 예쁘다	[최낙타]	20180410	[크리스마스캐럴, 분위기, 눈오는날, 연말, 캐럴, 겨울노래, 크리스마스송, 크리스...
121	349398	LOVE YA!	[혁오 (HYUKOH)]	20180531	[크리스마스캐럴, 분위기, 눈오는날, 연말, 캐럴, 겨울노래, 크리스마스송, 크리스...
122	631142	편지	[장희원]	20180603	[크리스마스캐럴, 따뜻한, 분위기, 사랑, 눈오는날, 연말, 캐럴, 겨울노래, 크리...
123	406082	하늘엔 별이 떠있고 너만큼은 빛나질 않아	[이민혁]	20180903	[크리스마스캐럴, 분위기, 눈오는날, 연말, 캐럴, 겨울노래, 크리스마스송, 크리스...
124	548389	사랑에 빠졌네	[정준일]	20181101	[크리스마스캐럴, 분위기, 눈오는날, 연말, 캐럴, 겨울노래, 크리스마스송, 크리스...
125	205179	꿀맛	[정미애]	20190815	[크리스마스캐럴, 생각, 다이어트, 눈오는날, 런닝, 휴식, 요가, 연말, 추천, ...
126	567076	숨겨진 세상 (Into the Unknown End Credit Version) (...	[태연 (TAEYEON)]	20191115	[크리스마스캐럴, 분위기, 눈오는날, 연말, 캐럴, 겨울노래, 크리스마스송, 크리스...
180	418694	Attention	[Charlie Puth]	20180511	[Pop, 휴식, 감성, 트로피컬하우스, 나만알고싶은, 드라이브, 트렌드, 운동, ...

Figure 8. 태그·장르 데이터 모두 존재하지 않을 시 결과

3. 갱신된 과제 추진 계획

지금까지 만들어진 음악 추천 모델은 입력 데이터를 별다른 전처리 없이 사용한다는 점에서 한계가 존재한다. 본 과제의 목표인 '음악 데이터 전처리 효율성 연구' 를 위해 향후 과제 계획에서는 이를 해결하는 것을 최우선 순위로 높여 진행하고자 한다. 또한 음악 추천 모델의 평가 지표에 따라 현재 생성한 음악 추천 모델의 성능을 검증하고, 추가적으로 유사도 측정 방식을 다르게 하여 만들어진 서로 다른 모델 간의 성능 또한 비교해 볼 예정이다.

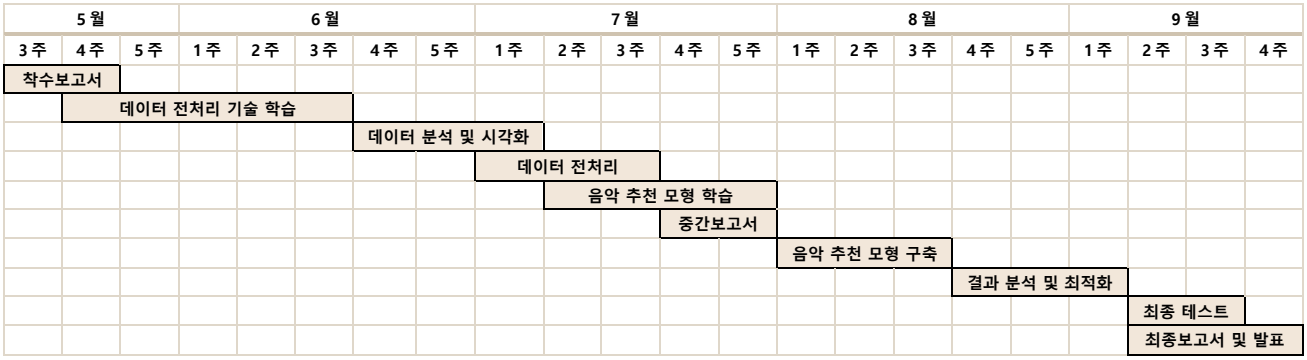

Table 15. 이전까지의 과제 계획

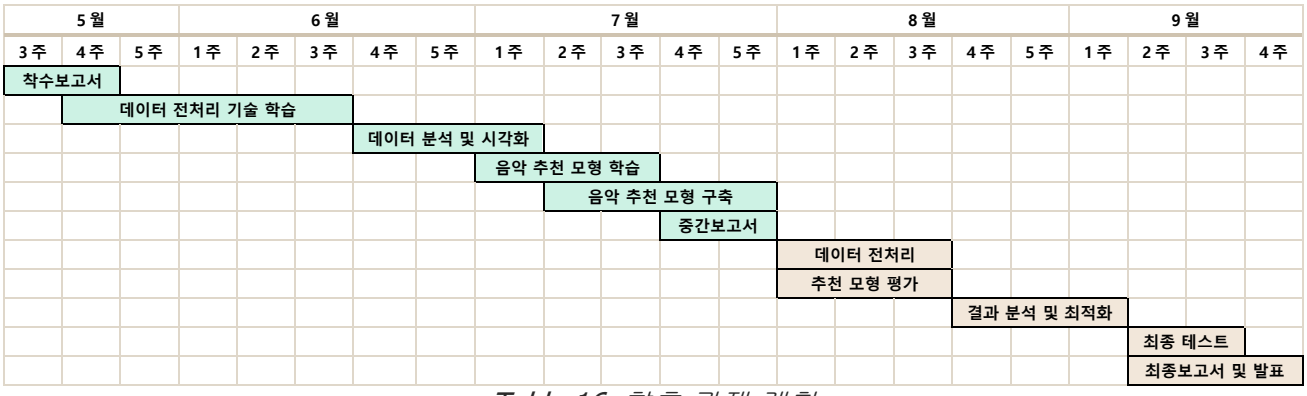

Table 16. 향후 과제 계획

4. 구성원 별 진척도

이름	역할
진현	Drop / Replace / Normalization 을 활용한 초기 데이터 전처리 기법 조합 mAP, nDCG 를 활용한 음악 추천 모델 성능 평가 지표 개발 진행 중
임우영	Autoencoder 를 이용한 모델 기반 CF 추천 모형 개발 Word2Vec, 벡터 평균을 이용한 Embedding Vector 변환 및 코사인 유사도 계산 추천 모델 평가 및 태그에 대해 자연어 전처리 진행 중
김영수	Word2Vec, 코사인 유사도를 이용한 CBF 추천 모형 개발 플레이리스트에 수록된 곡의 개수를 기준으로 불균형 데이터 전처리 진행 중
공통	추천 모델 스터디 및 코드 작성

Table 17. 구성원 별 역할