YouTube Recommendation

vectorize videos information using Embedding
and then
recommendation implementation using Cosine similarity

Konkuk univ. Software dept.

201614150 김 지 현 201614152 엄 현 식 201614153 이 권 호
지도 교수 김 성 렬

Abstract.

유튜브, 아마존, 넷플릭스, 페이스북, 인스타그램 등 많은 기업들이 추천 시스템을 적용하여 유저의 사용 시간을 늘리기 위해 노력하고 있다. 그 결과, 늘어난 유저들의 사용 시간에 비례해 많은 수익을 창출되고 있다. 추천 시스템을 분석하고 구현해 보는 것은 기업이 실행하고 있는 현실적인 인공지능을 이용한 수익창출을 이해하기 좋은 예시이다.

이 문서에서는 구글에서 직접적으로 공개하고 있지 않은 추천 시스템에 대해 이야기할 예정이다. 유튜브 홈페이지에서 영상의 추천 목록을 크롤링하여 데이터셋을 정의하였으며 이 데이터셋의 데이터를 시각화하고 분석하는 작업을 진행하였다. 영상을 각 노드라고 보고 해당 노드의 추천이 데이터셋 내에 존재할 경우 엣지를 연결하는 식으로 그래프를 정의하였다. 이 그래프를 분석하여 어떤 특성이 비슷할 때, 연결(엣지 형성)이 이루어지는지 분석한 결과, 영상의 카테고리, 채널, 등록일, 태그 리스트 등이 비슷하거나 같을 때, 많은 연결을 발견할 수 있었다.

이렇게 발견한 정보를 바탕으로 [2 - section]에서와 같이 분류 모델을 이용하여 추천 목록을 얻고자 모델을 정의하고 학습을 진행하였다. 결과적으로 학습이 잘 진행되지 않았으며 원하는 추천 목록도 얻을 수 없었다. 자세한 내용은 해당 section에서 설명하겠다.

분류 모델을 이용해 풀지 못했던 추천이 어떤 식으로 이루어지는지에 대한 세부 설명이 필요하다. 이는 [3 - section]에서 설명하겠다.

앞서 분류 모델을 학습시키기 위해 영상 정보의 텍스트를 정수로 변환하여 사용하였는데, 이 한줄, 한줄이 마치 n차원의 벡터처럼 보여 텍스트 분석에 사용되는 벡터화 방식을 추천에 활용해 보기 위해서 총 4가지의 방식을 시도하였다. 이는 [4 - section]에서 설명하겠다.

이렇게 얻은 벡터를 이용해 코사인 유사도를 이용해 상위 n개의 영상을 추천해주는 시스템을 완성했으며, 소유하고 있던 데이터 셋을 이용해 테스트를 진행했을 때, 약 70%의 추천목록을 재현하는 것을 볼 수 있었다.

1. Dataset.

이번 프로젝트에서 사용한 데이터셋은 기본적으로 kaggle(www.kaggle.com)에서 얻은 2017 - 2018 년도의 일부 국가 인기 동영상 리스트이다. 이 데이터셋에 포함되어 있는 video_id를 이용하여 각 영상의 추천 목록(20개)을 크롤링하였다.
youtube api 사용시 일일 트래픽 할당량의 문제로 사용하지않고 beautifulsoup4를 사용하였다.

	country_code	video_id	trending_date	title	channel_title	category_id	publish_time	tags	views	likes
0	0	0dBIkQ4Mz1M	17.14.11	PLUSH - Bad Unboxing Fan Mail	iDubbbzTV	23	2017-11-13T17:00:00.000Z	plush|"bad unboxing"|"unboxing"|"fan mail"|"id...	1014651	127794
1	0	5qpjK5DgCt4	17.14.11	Racist Superman | Rudy Mancuso, King Bach & Le...	Rudy Mancuso	23	2017-11-12T19:05:24.000Z	superman|"rudy"|"mancuso"|"king"|"bach"...	3191434	146035

그림 1-1. dataFrame.head()에서 각 영상의 정보 부분

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	
	Unsavory Group of Bad Hombres - Bad Unboxi...	憲□???ь 깵紐⑸畚 - PLUSH - Bad Unboxing Fan Mail	Bad Unboxing - MAN PROPOSES TO CRAZY HOUSE...	Bad Unboxing (STAR WARS) - Count Deku aven...	Bad Unboxing - Talking about my wrist inju...	Bad Unboxing	Amazon FAN Mail - Bad Unboxing	WOAH Crazy Toy! - Bad Unboxing Fan Mail	POKEMON KINDER EGGS - Samurai Buyer Unboxi...	Bad Unboxing - CANS!!! (Beans, Chili, Spag...	Puppet Show - Bad Unboxing	Bad Unboxing - Christmas *GIFT OPENING* Pr...	Bad Unboxing - A MAN APPLIES MAKEUP	REAL Orange Troll Man - Bad Unboxing Fan M...	F U
	Racist Superman | Rudy Mancuso, Alesso & K...	憲□???ь 깵紐⑸畚 - Superman vs Superman | Rudy Mancu...	Step Up: High Water | OFFICIAL TRAILER	Terrible Tennis Players | Rudy Mancuso & J...	Blood and Bone - Breaking The Mold Behind ...	Funny Dan Nampaikid Vines (W/Titles) Dan N...	Superhero Vacation | Rudy Mancuso, Anwar J...	360 | THE JOKER	Who the f*** is Wayne? | Wayne Official Te...	Worst Waiter Ever | Rudy Mancuso	Funny Instagram Videos November 2017 #2 | ...	Superhero Therapy | Rudy Mancuso, Lele Pon...	Top King Bach Vine Compilation | All King ...	Superhero Roommates | Rudy Mancuso, Anwar ...	A M

그림 1-1. dataFrame.head()에서 각 영상의 추천 목록

여기서 얻은 데이터를 바탕으로 분석을 진행하였는데, 우선적으로 어떤 영상이 서로 추천되는지 알아보기 위해 커다란 그래프로 그려보았다. 이 그래프를 그려 구글의 어떠한 알고리즘에 대한 정보를 찾는 것을 목표로 진행하였으나 뜻대로 이루어지지 않았다.

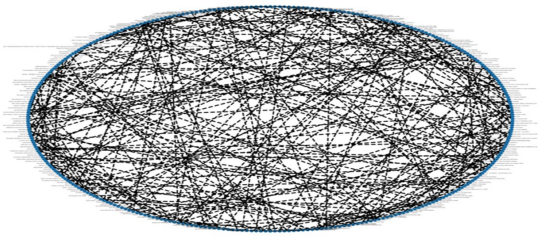

그림 1-3. connection graph

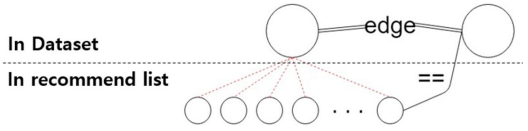

그림 1-4. edge 연결 방식

각 영상을 하나의 노드로 정의하고 그 노드의 추천 목록에 데이터셋 내의 노드(영상)가 존재할 경우, 두 노드를 엣지로 연결하는 그래프이다.

서로 연결된 노드의 수가 어느 정도 있다는 것을 확인하고 데이터 분석에 들어갔다. 어느 특성을 서로 공통적으로 가지거나 유사할 때, 연결이 발생하는지 알아보기 위해 각 노드를 방문하면서 추천 목록의 특성과 비교해 가중치를 계산하는 프로그래밍을 진행하였다. 이 때, 오차가 발생할 여지가 없는 국가 코드, 채널 타이틀, 카테고리, tag 등에 대해서는 서로 같은지 체크하였으며 좋아요 수, 싫어요 수, 댓글 수 등 수적으로 표현된 특성의 경우 오차를 +/- 10%로 주어 비교하였다. (추가적으로 20%, 5%에 대해서도 확인하였다.)

결과적으로 좋아요 수, 싫어요 수 그리고 댓글 수 등은 거의 영향을 끼치지 못하였다. 그 이유로는 이번에 진행한 프로젝트가 유저 데이터 기반의 추천 목록을 크롤링하여 실험한 것이 아닌 실제로 그 영상 자체만을 놓고 추천이 이루어진 목록을 활용했기 때문에 앞선 결과를 보인 것으로 생각된다.

유저 데이터의 경우, 유튜브 측에서 사생활 문제, 악용 가능성 등의 이유로 공개하지 않고 있기 때문에 얻을 수 있는 영상 정보 데이터만을 가지고 실험 및 프로젝트를 진행하게 되었다.

2. Fail Try (classify).

처음 문제를 해결하기 위해 시도한 방법은 분류 모델을 추천 시스템에 적용하는 것이었다. 그 당시 알고 있는 머신 러닝 관련 지식은 NN(Neural Network), KNN, SVM 등을 이용한 기초적인 분류 모델밖에는 없었다. [2 - section]에서 설명했다시피 우리의 프로젝트에서 사용한 데이터셋은 각 영상마다 정보를 가지고 있으며 크롤링을 이용해 정답 레이블을 얻었다고 판단했기 때문에 충분히 분류를 활용할 수 있을 것이라 판단했다.
분류 모델의 구현에는 Keras를 사용하였다. Keras는 모델을 정의하고 쉽게 layer들을 쌓을 수 있으며 각 layer에서 사용할 수 있는 다양한 종류의 활성화 함수(Relu, Sigmoid 등)와 손실 함수 등을 제공하고 있기 때문에 빠른 시도를 해볼 수 있다고 판단해 진행하게 되었다.
해당 실험에서는 앞서 설명한 추천 목록에 직접적으로 영향을 미치는 속성들만을 이용하였다. 속성들이 숫자 형태로 존재하는 일부도 있지만 대다수가 문자열로 존재했기 때문에 계산 과정을 위해 이를 인덱싱하여 정수 형태로 바꿔 사용하였다.

```
[12]: model.compile(optimizer=optimizers.RMSprop(lr=0.001),
                    loss='mse',
                    metrics=['accuracy'])

[14]: model.fit(x_train, y_train, epochs=100, batch_size=64)
Epoch 7/100
15000/15000 [==============================] - 0s 16us/step - loss: nan - acc: 0.6245
Epoch 8/100
15000/15000 [==============================] - 0s 15us/step - loss: nan - acc: 0.6245
Epoch 9/100
15000/15000 [==============================] - 0s 14us/step - loss: nan - acc: 0.6245
Epoch 10/100
15000/15000 [==============================] - 0s 15us/step - loss: nan - acc: 0.6245
Epoch 11/100
15000/15000 [==============================] - 0s 19us/step - loss: nan - acc: 0.6245
Epoch 12/100
15000/15000 [==============================] - 0s 15us/step - loss: nan - acc: 0.6245
Epoch 13/100
15000/15000 [==============================] - 0s 15us/step - loss: nan - acc: 0.6245
Epoch 14/100
15000/15000 [==============================] - 0s 16us/step - loss: nan - acc: 0.6245
Epoch 15/100
15000/15000 [==============================] - 0s 16us/step - loss: nan - acc: 0.6245
Epoch 16/100
```

그림 2-1. 학습이 제대로 이루어지지 않는 분류 모델

multi -class 문제로 접근을 했기 때문에 cost function을 softmax로 적용시켰다.
하지만 unique한 정답 데이터의 개수가 10만개 정도를 one - hot encoding형식으로 정답레이블을 구성하였더니 손실함수가 줄지 못하고 학습이 진행되지 않은 것을 확인 할 수 있었다.
따라, input 과 정답 label간의 연관성을 발견하지 못한것이다.

만약 labelpowerset으로 multi-class가 아닌 multi -label 접근 방법으로 진행하였다면 2^10만개의 조합 중 Dataset 안에 우리가 갖고있는 조합으로 줄일 수 있었겠지만 역시 너무많은 정답 label의 개수 때문에 학습이 진행되지 않을것이라고 판단하였다.

3. Recommendation.

3-1. 아이템 데이터 기반 추천

아이템 데이터 기반 추천이란 사용자가 어떤 아이템을 구매하면 이 아이템과 비슷한 아이템을 추천해주는 시스템이다. 따라서 다수의 사용자에게로 쉽게 확장 가능하며, 사용자의 특정 관심사를 특징할 수 있다. 하지만, 이 모델은 사용자가 특정 관심사 외의 다른 아이템에도 관심이 있을 경우를 예측할 수 없다.

3-2. 협업 필터링

협업 필터링이란 어떤 아이템 A와 B를 사용자 a와 b가 모두 구매했고, 사용자 a가 아이템 C를 구매했다면, 사용자 b에게 아이템 C를 추천해 주는 시스템이다.

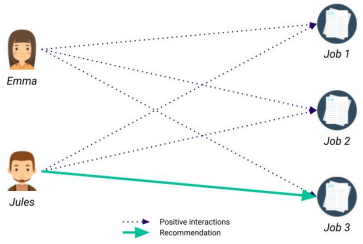

그림 3-1.협업필터링이란?

협업 필터링에는 유저-유저 협업 필터링과 아이템-아이템 협업 필터링의 두 가지 종류가 있다. 유저-유저 협업 필터링은 해당 사용자와 취향이 비슷한 사람에게 가중치를 주는 방식을 말한다. 해당 사용자와의 유사성을 곱해서 가중치를 더 주고 반대의 가중치에는 마이너스 값을 곱해 가중치를 줄여 준다. 개인의 취향이 고려되기 때문에 각 사용자에 맞춘 개인화된 추천 방식을 가지고 있다. 아이템-아이템 협업 필터링은 사용자가 이전에 선호한 아이템과 가장 유사한 아이템을 추천해준다. 따라서 아이템 간의 유사도가 높고, 사용자에게 유사도가 높은 새로운 아이템을 추천해줄수 있다는 장점이 있다.

3-3. 하이브리드

하이브리드 추천 시스템이란 [3-1].과 [3-2].의 추천 방식을 섞은 것인데 협업 필터링의 방식에서 두 사용자가 구매한 아이템의 특성이 비슷하다면 추천하는 시스템이다. 예를 들어 넷플릭스는 사용자가 높게 평가했던 영화와 비슷한 특성을 띄는 영화를 추천하고, 비슷한 사용자들의 검색 습관과 시청 여부를 비교함으로써 추천한다. 여기에서 사용자가 높게 평가했던 영화는 아이템 데이터 기반 추천의 아이디어이고, 비슷한 사용자들의 검색 습관, 시청 여부는 협업 필터링의 아이디어이다.

4. Vectorizer.

2번에서 분류모델을 이용한 추천 구현에는 실패했지만 데이터를 정수화해보니 벡터와 생김새가 유사해 여기서 아이디어를 얻어 영상의 정보를 여러가지 방식으로 (Text-mining) 벡터화하였다.
사용한 벡터화 방식은 아래에 설명하겠다.

4-1. countVectorizer

하나의 문서에서 각 단어의 빈도수를 활용하여 벡터화하는 방식이다. 그렇기 때문에 벡터화 방식은 단순하다. 그러나 단순히 빈도수만을 활용하고 있기 때문에 특정 문서 안에만 들어있는 unique한 단어의 가중치를 높이지 않기때문에 문서간의 상관관계를 얻기 어렵다는 단점이 있다.

4-2. Tf - idf

Tf-idf란 문서의 집단이 있을 때, 어떤 단어 a가 특정 문서에서 얼마나 중요한지 수치화하는 모델이다. Tf-idf에서 TF(term frequency)란 어떤 단어가 특정 문서에서 얼마나 자주 등장하는지의 값이고, IDF(inverse document frequency)란 어떤 단어 a가 전체 문서에서 얼마나 많이 등장하는지의 값의 역수다. 따라서 문서의 집단에서 IDF값이 높을 수록 해당 문서를 특정할 수 있는 단어일 확률이 크다. 이 TF와 IDF를 곱한 것이 바로 Tf-idf이다. 즉 Tf-idf는 특정 검색어가 많이 포함된 문서에 점수를 높이 매기는 방식이다.

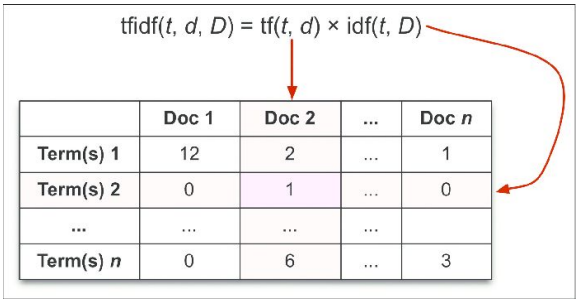

그림 4-1. tf - idf

4-3. BM 25

BM25는 문서 내에서의 근접성에 관계없이 각 문서에 나타나는 용어에 따라 문서 세트의 순위를 지정하는 단어 모음 검색 기능이다.
tf-idf와 같이 tf, idf, 문서의 길이를 고려하는데 tf-idf와 다른 점은 tf와 문서의 길이의 영향이 줄어들고 idf의 영향이 커진다는 점이다. 먼저 tf에서는 단어 빈도가 높아질수록 검색 점수도 지속적으로 높아지는 반면, BM25에서는 특정 값으로 수렴한다. 그리고 BM25에서는 DF가 높아지면 검색 점수가 0으로 급격히 수렴하므로 조사, 접미사 등이 검색 점수에 영향을 덜 미친다.
마지막으로 tf-idf와 달리 문서의 길이의 평균값을 적용하여 문서의 길이가 검색 점수에 영향을 덜 미친다. 즉 BM25는 다른 문서와 차별화되는 단어가 많이 포함된 문서에 점수를 높이 매기는 방식이다.

4-4. Embedding layer

Keras를 이용한 임베딩은 각 영상의 제목과 그 영상이 가지는 특성들의 1 대 1 pairs로 학습이 진행된다. video embedding과 attribute embedding 둘로 나눠 영상과 속성들의 인덱싱 길이에 맞는 차원을 가지게 된다. 이 때, 모델에 인풋되는 파라미터는 총 3200만개가 된다. 이렇게 두개로 나눠진 embedding 결과를 Dot()을 통해 하나의 벡터로 만드는 작업을 진행하게 된다. 이렇게 같은 영상에 대해서 벡터를 업데이트 하는 식으로 embedding된다. 결과로 나오는 embedding vector의 사이즈를 50부터 200까지 진행한 결과, 100까지는 유의미한 성능의 향상을 보였지만 100 이후의 사이즈에 대해서는 성능향상에 비해 높은 계산량을 보였다. 결국 최종 embedding 사이즈는 100으로 정하게 되었다.

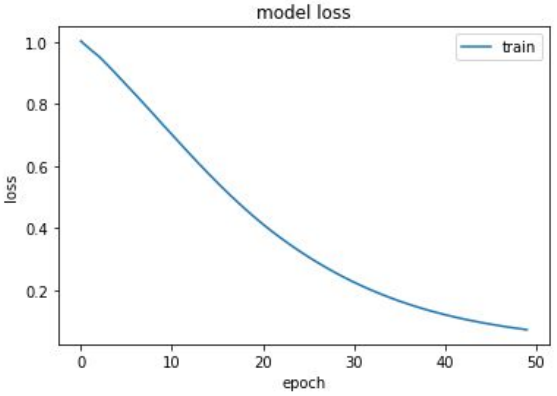

그림 4-2. 임베딩 모델 학습 그래프

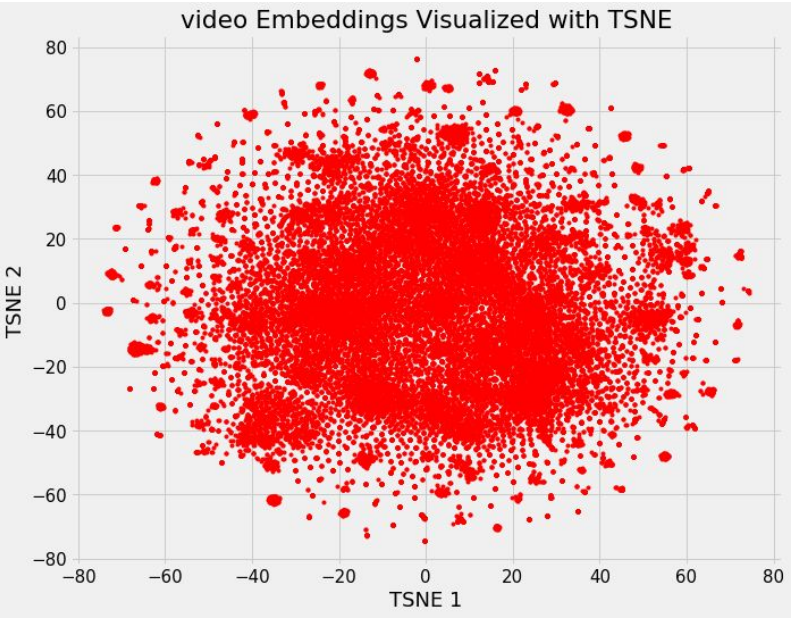

그림 4-3. 임베딩 모델 결과 표현
(100차원에서 2차원으로 차원 축소)

5. Experiments.

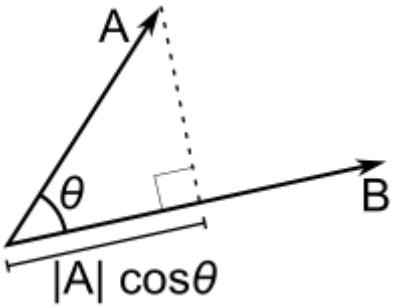

그림 5-1. 각도를 이용한 벡터간의 유사도

각도가 작을수록 같은 방향을 가르키는 유사한 벡터라는 개념을 이용해 실험을 진행했다.

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

그림 5-2. 코사인 유사도 공식

[4-4]에서 설명한 임베딩 레이어에서 n차원 공간에 벡터를 뿌린 후 코사인 유사도를 이용하여 벡터간의 유사도를 측정한다.
코사인 유사도란 내적공간의 두 벡터간 각도의 코사인값을 이용하여 측정된 벡터간의 유사도를 말한다. 각도가 0°일 때의 코사인값은 1이며, 180°일 때는 -1이다.
따라서 이 값은 벡터의 크기가 아닌 방향의 유사도를 판단하는 목적으로 사용된다.
코사인 유사도 외에 다른 방식을 사용하지 않은 이유는 벡터를 만들 때 다른 attribute들과는 달리 tags는 각 영상마다 개수가 달라서 그대로 벡터를 만든다면 크기가 일정하지 않은 벡터가 된다.
이 방식에서 벡터의 크기를 고려하는 다른 유사도 측정 방식을 사용한다면 결과값이 상당히 다르게 나올 수 있기 때문에 벡터의 크기를 고려하지 않는 코사인 유사도를 사용하였다.
예를 들어 유클리드 거리를 사용했을 경우, 같은 문서를 두번 복사 한다면 두 벡터 간의 유클리드 차이 값이 발생하지만 코사인 유사도로 한다면 방향이 같기 때문에 더 정확한 방법으로 측정 할 수 있다.
따라서 벡터의 각도와 방향이 일치할수록 높은 점수를 가지게 된다. 이런 식으로 뽑은 상위 n개가 우리가 예측한 추천 리스트를 나타낸다. 각 텍스트 벡터화 방식마다 상위 k개를 뽑아 각각 재현율을 측정한 표는 아래에서 설명하겠다.

(A) Countvectorizer recall

t0p_k (개)	Max (%)	Min (%)	Avg (%)
03	100.0	0.0	14.6
05	100.0	0.0	19.9
07	100.0	0.0	23.2
10	100.0	0.0	25.9
15	100.0	0.0	32.2
20	100.0	0.0	34.5

(B) TF- idf recall

t0p_k (개)	Max (%)	Min (%)	Avg (%)
03	100.0	0.0	17.3
05	100.0	0.0	21.6
07	100.0	0.0	25.3
10	100.0	0.0	30.4
15	100.0	0.0	33.7
20	100.0	0.0	37.9

(C) Bm 25 chart

t0p_k (개)	Max (%)	Min (%)	Avg (%)
03	100.0	0.0	23.3
05	100.0	0.0	30.2
07	100.0	0.0	32.9
10	100.0	0.0	36.2
15	100.0	0.0	40.2
20	100.0	0.0	46.3

(D) Word embedding

t0p_k (개)	Max (%)	Min (%)	Avg (%)
03	100.0	0.0	39.3
05	100.0	0.0	47.2
07	100.0	0.0	54.9
10	100.0	0.0	59.2
15	100.0	0.0	64.2
20	100.0	0.0	69.3

그림 5-3. 영상의 정보를 가진 텍스트를 벡터로 만든 4가지 방법인
CountVectorizer, TF-idf, Bm 25, Embedding model의
유사도 상위 n개로 구성된 추천 목록이 각 영상의 정답 목록을 얼마나 재현했는지 기록한 표

countvectorizer, Tf-idf, Bm 25, Embedding model의 4가지 방법을 이용해 각자 (top K) 03 ~ 20 개의 추천 리스트를 뽑아 정답 데이터를 얼마나 재현해냈는지 비교해 보았다. 재현율이란 쉽게 말해 정답이라고 예측한 것 중 실제로 정답인 것들의 비율이다.

Accuracy, Precision, Recall, F1-score 같은 여러가지 데이터를 판단하는 기준이 있지만 재현율을 선택한것은 추천시스템을 얼마나 유사하게 표현했냐는것에 의미를 두었기때문에 Recall을 사용하였다.

최종적으로 사용한 모델은 임베딩 모델이며, 그 재현율은 약 70% 정도이다. 이 재현율은 우리가 수집하고 정제한 데이터셋 안에서 측정한 비율이기 때문에 우리가 직접 크롤링해 얻은 추천리스트를 순서까지 정확히 맞추는 것이 아닌 그 중에서 몇 개를 재현해냈는지 계산했다.

6. Conclusions.

우리 생활에 밀접하게 연관되어 있는 추천 시스템에 대해 공부하고 이를 구현해보는 프로젝트를 진행하였다. 구글에서 별도로 공개하고 있지 않고 있는 추천 시스템에 대한 정보를 최대한 수집하였으나 얻을 수 없는것들도 존재했다. 그렇기에 유튜브에서 직접 데이터를 크롤링하고 데이터셋으로 만들어 이를 활용해 우리 팀만의 추천 시스템을 구현한 뒤에 유튜브의 추천 결과와 비교하는 방식으로 프로젝트를 진행하게 되었다.
그 결과 대략 70% 정도의 재현율을 보이는 시스템 구현에 성공할 수 있었으며, 유저 데이터를 얻을 수 있다면 협업필터링 방식을 이용한 더 높은 성능의 추천 시스템을 완성시킬 수 있을 것이다.

References.

- www.youtube.com [크롤링]
- https://www.kaggle.com/coffeeinspace/youtube-videos-cleaned-set
- 유소엽, 정옥란. (2014). 소셜 카테고리 기반 유튜브 추천 시스템. 한국정보과학회 학술발표논문집, (), 138-140.
- SINGHAL, Ayush; SINHA, Pradeep; PANT, Rakesh. Use of deep learning in modern recommendation system: A summary of recent works. *arXiv preprint arXiv:1712.07525*, 2017.
- DAVIDSON, James, et al. The YouTube video recommendation system. In: *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010. p. 293-296.