뉴럴 네트워크(딥러닝)를 활용한 이커머스 추천 서비스 개발 프로젝트

DS 1기 김현재 오예은

목차

1. 프로젝트 진행 배경

- **이커머스 서비스**

- 프로모션 / 인기제품 중심의 추천
- 사용자가 특정 정보를 직접 입력한 상태에서의 개인화 추천
- 전통적인 추천모델(협업필터링 등)에서 나아가 더 개인화된 추천 모델 개발

- **이커머스 vs. 유튜브**

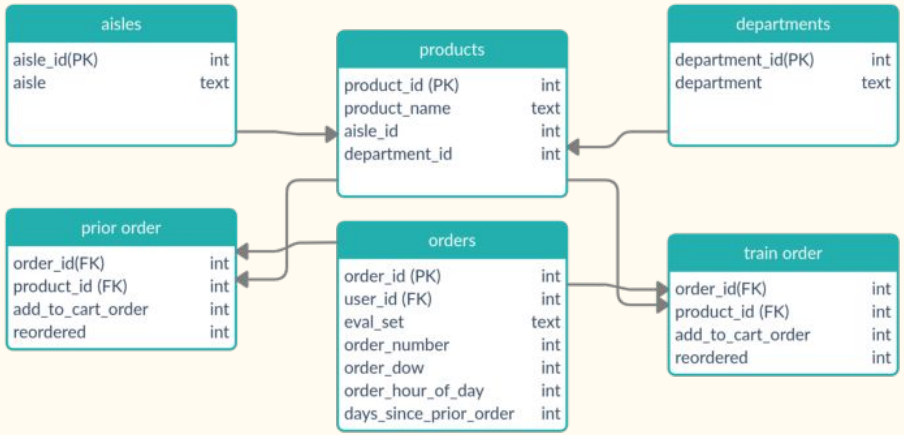
- 실시간으로 많은 데이터를 양산 (온라인 활동 중심)
- 딥러닝 콘텐츠 추천 모델을 커머스(쇼핑) 데이터에 적용
- 유저의 라이프스타일 또는 취향에 따른 구매이력(과거 데이터)을 활용해 딥러닝모델로 앞으로의 구매를 예측하여 추천하는 이커머스 플랫폼 제안

2. 이커머스 데이터셋 및 가설 소개

- **데이터셋 설명**

  Instacart Market Basket Analysis dataset (Kaggle)

  https://www.kaggle.com/c/instacart-market-basket-analysis/data



이미지 https://ichi.pro/assets/images/max/724/1*0c0vK-sbU8OqObWFs7imHw.png

- **주요 특성:** 주문번호, 제품번호, 주문요일, 주문시간대, 이전구매로부터소요기간

- **전처리 사항:**
  1) [orders]의 'days_since_prior_order (이전구매로부터소요기간)' 결측치(첫 구매 데이터) 제외
  2) 총 3,421,083개의 주문건수 중 'test'로 표기된 이용자 데이터를 제외
  3) 1만 명의 샘플링된 유저 구매정보만 사용 (총 1,566,110건)

2. 이커머스 데이터셋 및 가설 소개

- **가설**

> "유튜브 추천 방식인 ranking 모델에 Instacart 이용자의 **'재주문여부(reodered)'**를 상품에 대한 평점(rating)으로 활용하여 추출한 추천목록(1안)이 **'주문회차(order_number)'**를 평점(rating)으로 활용한 추천목록(2안)보다 **nDCG score**가 높을 것이다."

→ Input rating feature에 따른 1,2안 모델의 추천 결과 비교

- **분석 방법** - YouTube 딥러닝모델 활용

1) Candidate generation model에서 후보 목록 추출
2) Ranking model을 활용하여 위에서 추출된 목록에서 추천 순위 목록 추출
   2-1) 추천순위 목록에 투입되는 rating feature(like vs. dislike)를 2가지(아래 1, 2안)로 나누어 진행

1안	2안
• **'재주문여부(reodered)'**를 평점(rating: like/dislike)으로 활용 • 해당 상품을 재주문한 경우(1) like, 아닌 경우(2) dislike	• **'주문회차(order_number)'**를 평점(rating: like/dislike)으로 활용 • 해당 주문이 평균 주문회차(건수)인 18번째 이상이면 like, 미만이면 dislike

3) 1, 2안의 추천 순위 목록의 평가지표인 nDCG score 비교

3. 딥러닝 기반 유튜브 추천모델 개념

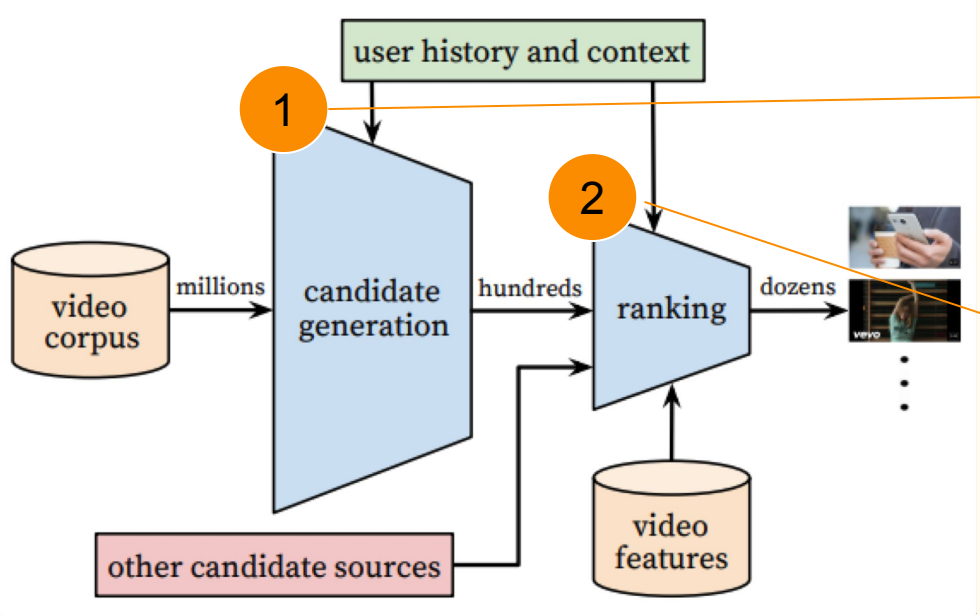


Figure 2: Recommendation system architecture demonstrating the "funnel" where candidate videos are retrieved and ranked before presenting only a few to the user.

[Step1] Candidate generation모델

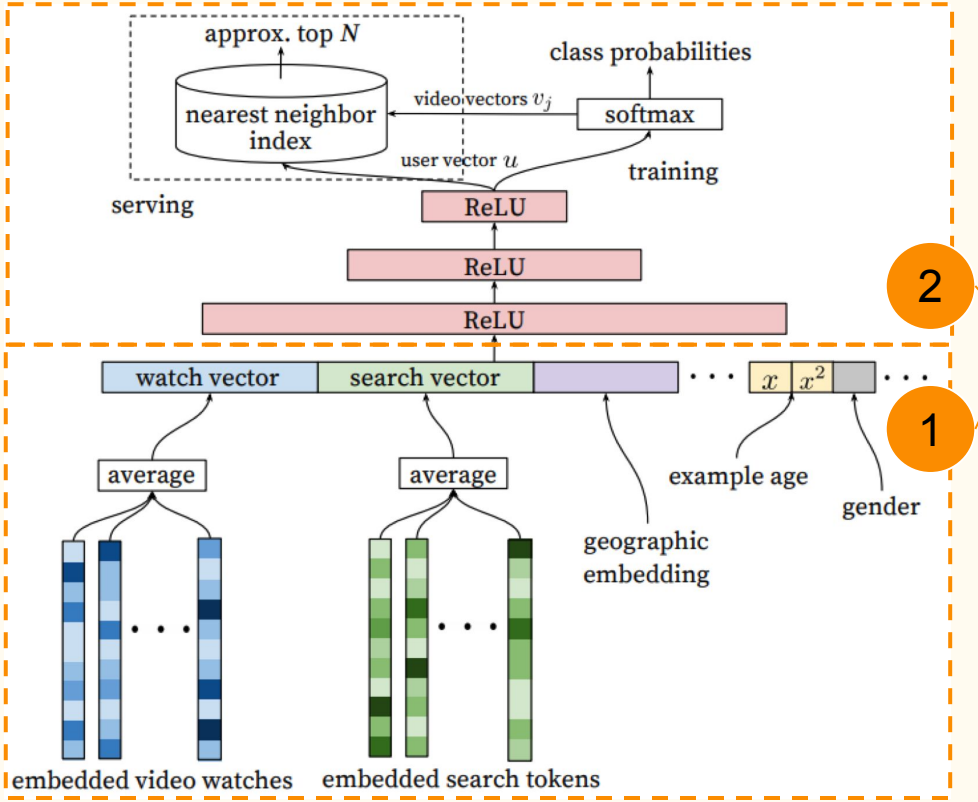
모델에서 사용자가 시청했던 영화, 검색어 등을 기반으로 '후보 추천목록'을 추출한다.

[Step2] Ranking 모델

후보모델을 더 정교화한 것으로, 사용자에게 노출되었지만 보지 않은 영상들, 좋아요 누른 영상, 영상의 장르 등 다른 특성들을 추가하여 '추천목록'들에 점수를 매겨 '순위 추천목록'을 추출한다.

※출처: 'Deep Neural Networks for YouTube Recommendations'-Paul Covington, Jay Adams, Emre Sargin Google

3. 딥러닝 기반 유튜브 추천모델 개념-(1)후보모델(Candidate)

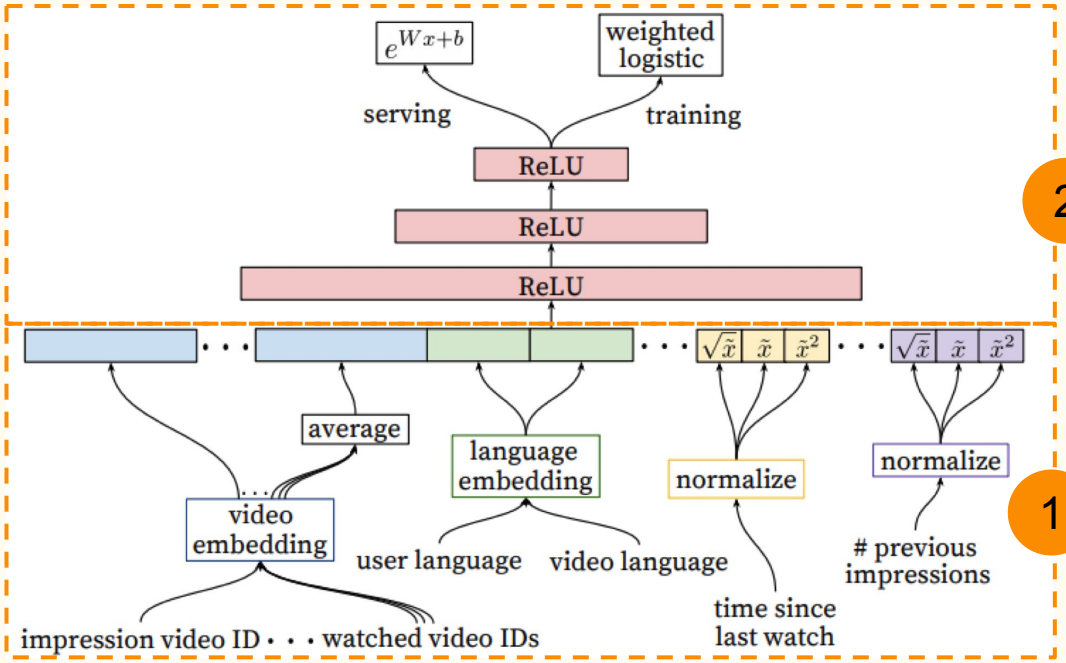


[1. Input data]
유저별로 과거 시청했던 영상정보, 검색어 정보, 성별, 직업 등 유저특성데이터를 학습데이터로 사용

[2. Model structure]
뉴럴네트워크(딥러닝) 학습구조 사용.
-은닉층 계산활성화함수 ReLU
-출력층 다중분류문제로 softmax 활성화함수 사용.
-출력결과물: 과거 시청영상과 가장 근접한 영상정보 인덱스 N개 후보목록을 추출.

※출처: 'Deep Neural Networks for YouTube Recommendations'-Paul Covington, Jay Adams, Emre Sargin Google

3. 딥러닝 기반 유튜브 추천모델 개념-(2)순위모델(Ranking)



[1. Input data]
유저별로 노출된 영상, 시청한 영상, 시청한 시간 등 학습데이터로 사용

[2. Model structure]
뉴럴네트워크(딥러닝) 학습구조 사용.
-은닉층 계산활성화함수 ReLU
-출력층 softmax 활성화함수 사용.
-출력결과물: Top N 순위의 추천 목록 인덱스 추출

※출처: 'Deep Neural Networks for YouTube Recommendations'-Paul Covington, Jay Adams, Emre Sargin Google

4. 이커머스 데이터를 적용한 추천모델 - (1)후보추천모델

〈Train_data〉
Instacart 유저별 주문데이터에서
랜덤으로 10명을 선정.

〈Feature〉
-user: user_id를 순서대로 배열한 인덱스값
-user_id: user_id고유값
-product_id: 각 유저별 주문상품번호 리스트 (ex. [45008, 23877, 33900,..])
-order_hour_of_day: 유저별 주문 시간 리스트 (ex. [2, 5, 10, …])
-days_since_prior_order: 각 유저별 이전 주문에서 다음 주문 사이 날짜 (ex. [3, 5, 9, ..])
-predict_labels: 유저별 label. (product_id의 범주 내에서 유저별로 라벨을 붙임.)

〈Candidate 모델 성능 결과〉
Accuracy: **0.8** / loss : 0.7327

〈하이퍼파라미터〉
EMBEDDING_DIMS = 16
DENSE_UNITS = 64
DROPOUT_PCT = 0.1
ALPHA = 0.1
NUM_CLASSES = data["product_id"].max() + 2
LEARNING_RATE = 0.1

〈 Test_Data〉
Instacart 유저별 주문데이터에서
랜덤으로 10명 선정

〈Candidate 후보추천목록 결과〉

```
# candidate generation:
###### 각 user당 7개의 추천 후보 데이터(predict_label)를 뽑아낸다.
import numpy as np
N = 7
k = np.sort((-pred).argsort()[:,:N])
print(k)
k = k.flatten()
k[k>data["product"].max()]=0
k = np.unique(k)
```

```
k
```

```
array([    0,  2747, 12382, 25251, 28979, 29049, 30245, 30464, 31212])
```

5. 이커머스 데이터를 적용한 추천모델 - (2)순위추천모델 - 1안('reordered'사용)

〈Train_data〉
Instacart 유저별 주문데이터에서 랜덤으로 10명을 선정.

〈Feature〉
-user: user_id를 순서대로 배열한 인덱스값
-user_id: user_id고유값
-dislike: 재주문되지 않은 상품번호
-like: 재주문된 상품번호
-pd_name_d: **Candidate model에서 추천된 후보 상품들**의 주문상품명 (encorded) 리스트 (ex. [1299, 11206, ...])
-order_hour_of_day: 유저별 주문 시간(nomarlized) 리스트 (ex. [0.56521,0.34782, …])
-predict_labels: 유저별 label

〈 랭킹1안 모델 성능 결과〉
Accuracy: **0.7** / loss : 0.4736

〈하이퍼파라미터〉
EMBEDDING_DIMS = 16
DENSE_UNITS = 64
DROPOUT_PCT = 0.1
ALPHA = 0.0
NUM_CLASSES=new_data["product_id"].max() + 3
LEARNING_RATE = 0.003

〈 Test_Data〉
Instacart 유저별 주문데이터에서 랜덤으로 10명을 선정.

〈 랭킹1안 추천순위 결과〉

```
# ranking1안
###### 각 user당 top-7개의 추천 데이터를 뽑아낸다.
N = 7
ranking_1 = (-pred_1).argsort()[:, :N]

ranking_1[ranking_1>new_data['product_id'].max()]=0
print(ranking_1)

ranking_1_probability = np.sort(pred_1[:, :N])
print(ranking_1_probability)
```

```
[[ 5130 14200  9851 21906 12264 24391 19367]
 [12264 19367 21906 29486 10138   447 23355]
 [ 5130 14200  9851 21906 12264 24391 19367]
 [19367 24391  9805 24352 21906  5130 12264]
 [15210 24391  9851  5130 14200  9805 24352]]
```

5. 이커머스 데이터를 적용한 추천모델 - (2)순위추천모델 - 2안('order_number' 사용)

〈Train_data〉
Instacart 유저별 주문데이터에서 랜덤으로 10명을 선정.

〈Feature〉
-user: user_id를 순서대로 배열한 인덱스값
-user_id: user_id고유값
-dislike: 평균 주문회차(18) 미만 상품번호
-like: 평균 주문회차(18) 이상 상품번호
-pd_name_d: **Candidate model에서 추천된 후보 상품들**의 각 유저별 주문상품명(encorded) 리스트 (ex. [1299, 11206, ...])
-order_hour_of_day: 유저별 주문 시간 (nomarlized) 리스트 (ex. [0.56521,0.34782, …])
-predict_labels: 유저별 label

〈 랭킹2안 모델 성능 결과〉
Accuracy: **0.8** / loss : 0.2804

〈하이퍼파라미터〉
EMBEDDING_DIMS = 16
DENSE_UNITS = 64
DROPOUT_PCT = 0.1
ALPHA = 0.0
NUM_CLASSES=new_data["product_id"].max() + 3
LEARNING_RATE = 0.003

〈 Test_Data〉
Instacart 유저별 주문데이터에서 랜덤으로 10명을 선정

〈 랭킹2안 추천순위 결과〉

```
# ranking2안
###### 각 user당 top-7개의 추천 데이터를 뽑아낸다.
N = 7
ranking_2 = (-pred_2).argsort()[:, :N]
ranking_2[ranking_2>new_data_2['product_id'].max()]=0
print(ranking_2)

ranking_2_probability = np.sort(pred_2[:, :N])
print(ranking_2_probability)

# 추출된 인덱스 값들은 user별 구매목록라벨(여러개 상품들이 있는).
# 유사도 높은 다른 유저들의 구매목록이 추천된 셈.
```

```
[[28889  2482   895 10521 27969  8783 24049]
 [20909  4382 27647 10521 27969   685   895]
 [  895 28889  2482 10521 27969  8783  4220]
 [20909 27647  4382 27969 16488  6559 26788]
 [ 2482 28889   895  8783   685 27969 10521]]
```

6. 1,2안 순위 평가 : nDCG 계산원리 및 결과 해석

$$Cumulative\ Gain(CG) = \sum_{i=1}^{n} relevance_i$$

$$DCG = \sum_{i=1}^{n} \frac{relevance_i}{log_2(i+1)}$$

$$DCG_A = \frac{2}{log_2(1+1)} + \frac{3}{log_2(2+1)} + \frac{3}{log_2(3+1)} + \frac{1}{log_2(4+1)} + \frac{2}{log_2(5+1)} \approx 6.64$$

$$DCG_B = \frac{3}{log_2(1+1)} + \frac{3}{log_2(2+1)} + \frac{2}{log_2(3+1)} + \frac{2}{log_2(4+1)} + \frac{1}{log_2(5+1)} \approx 7.14$$

- DCG: Discounted Cumulative Gain
- iDCG: ideal DCG
- nDCG: normalized DCG = DCG / iDCG

[Step1]
본 프로젝트에서는 CG를 모두 1로 가정후,

[Step2]
랭킹 1,2안의 각각의 추천목록 중 **실제 유저가 true_like한 경우의 순위를 DCG계산에 포함**하여 DCG를 산출.

[Step3]
iDCG는 모든 순위를 포함하여 계산.

[Step4] DCG / iDCG = nDCG 를 계산.
⇒ **랭킹 1,2안들 중 어느 쪽이 더 true_like에 가깝게 추천하였는지 평가.**

7. 결론

►(1) 랭킹 1안의 nDCG 스코어가 랭킹 2안보다 0.05 높으므로, 가설검증됨.

- **ranking_1안 ndcg 평균: 0.41**
- ranking_2안 ndcg 평균: 0.36

[ranking_1안의 dcg, idcg, ndcg평균]

dcg	1.628199
idcg	3.953465
ndcg	0.411841

[ranking_2안의 dcg, idcg, ndcg평균]

dcg	1.450625
idcg	3.953465
ndcg	0.366925

►(2) 유저별로 nDCG스코어를 보면, 랭킹 1안이 높은 경우가 있고, 랭킹 2안이 높은 경우가 있으므로, 유저별 맞춤 제안으로 활용하면 좋을 것.

- **user3: 0.53 (ranking1_ndcg) >** 0.34 (ranking2_ndcg)
- **user1**: 0.53 (ranking1_ndcg) **< 0.61 (ranking2_ndcg)**

[ranking_1안의 유저별 dcg, idcg, ndcg]

	dcg	idcg	ndcg
test-user1	2.130930	3.953465	0.539003
test-user2	0.630930	3.953465	0.159589
test-user3	2.130930	3.953465	0.539003
test-user4	1.930677	3.953465	0.488351
test-user5	1.317529	3.953465	0.333259

[ranking_2안의 유저별 dcg, idcg, ndcg]

	dcg	idcg	ndcg
test-user1	2.448459	3.953465	0.619320
test-user2	1.000000	3.953465	0.252943
test-user3	1.356207	3.953465	0.343043
test-user4	1.130930	3.953465	0.286060
test-user5	1.317529	3.953465	0.333259

8. 본 프로젝트의 향후 활용방안 제안

► 1. 마케팅 프로모션, A/B테스트 데이터로 활용
본 추천목록으로 나온 상품목록 중 미구매 상품이라면, 프로모션 쿠폰을 제공하는 등 구매 촉진을 유도할 수 있을 것.
1, 2안에서 나온 추천목록 상품들로 다양한 마케팅적 A/B 테스트가 가능할 것으로 보임.

► 2.추천목록의 다양성 지표 추가한 모델 업그레이드
'Entropy Diversity란?'
Entropy Diversity란 엔트로피의 개념을 추천 결과에 적용한 것으로, 모든 사용자들에게 비슷한 종류의 상품을 추천할 경우 해당 상품 추천은 자주 발생하므로 정보량이 낮다. 반면 개인에게 맞춤화 된 추천은 발생 횟수가 적으므로 정보량이 높아진다. 이들의 기대값을 구한 것이 바로 Entropy Diversity이다.
5명의 유저에게 A부터 다섯개의 아이템 가운데 한개를 추천해주는 model A와 model B가 있다고 가정한다. model A는 모든 유저에게 상품 A를 추천하였다고 할 때, A가 추천될 확률은 1, 정보량은 0이 된다. 모든 유저에 대한 추천의 정보량이 0이므로 Entropy Diversity 역시 0이 된다. 반면 model B의 경우 모든 유저에게 각기 다른 상품을 추천한다. 이 때 각각의 상품이 추천되는 확률은 0.2, 정보량은 0.32, 엔트로피는 1.6이 된다.
※ 참고자료(추천엔진의 성능지표 관련 블로그https://yeomko.tistory.com/32) 에서 발췌