# 데이터분석캡스톤디자인

# 10주차 수행보고

## Khupid 조

산업경영공학과 김동혁 관광학과 류연주 산업경영공학과 유정수

## Cluster fine-tuning 작업 및 User matrix 생성

+ 한국어 임베딩은 성능이 좋지 않아서, 영어로 진행하는 것으로 최종 결정

group_1	group_2	group_3	group_4	group_5	group_6	group_7	group_8	group_9	group_10	group_11	group_12	group_13
other	rude	low	good	shade	democratic	sour	simple	frozen	environmental	chaotic	sofa	outdoor
different	wrong	high	nice	black	social	sullen	easy	decomposed	aquatic	crowded	unheated	nearby
same	unfriendly	flat	comfortable	silver	religious	shabby	basic	restored	marine	devastated	twostory	neighborhood
similar	ignorant	discounted	bad	blue	children's	gloomy	clear	recoverable	ecological	unorganized	spacious	downtown
actual	unreasonable	steep	mixed	dark	Buddhist	miserable	complicated	purchased	ocean	isolated	living	shouting
various	malicious	usd	hard	bright	cultural	dreary	logic	preserved	freshwater	congested	terraced	underground
such	suspicious	uptick	rusty	red	spiritual	dim	complex	collected	ivaiv	disorderly	cozy	residential
separate	guilty	upper	sloppy	yellow	orthodox	smoggy	pure	removed	NaN	tense	attic	guard
echo	illegal	cut	disappointing	green	nonreligious	depressing	sophisticated	distributed	NaN	unpopular	hallway	citywide
origir al	unscrupulous	discount	strong	white	communal	D. arom	n 5 group 0	group 1 gro	un 2 group 3	group 4 grou	ın 6. aroun i	7 aroup 8 ar

### Cluster fine-tuning 예시 설명:

group\_2: 친절도에 관한 형용사 group\_10: 환경에 관한 형용사

group\_11: 혼갑도에 관한 형용사

group\_13: 장소의 물리적 특성에 관한 형용사

inflated

#### User matrix 설명:

유저의 리뷰 중 각 클러스터에 해당하는 단어를 몇번 사용하였는지 나타내는 행렬, 이후 현업필터에 장소 매트릭스와 함께 사용될 예정

U.		group_5	group_0	group_1	group_2	group_3	group_4	group_6	group_7	group_8	group_9	group_10	group_11
	0	0	1	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	0	1	0	0	0	0	0	0	2	0	2
	3	2	4	15	1	0	0	5	13	0	21	0	3
/	4	0	0	1	0	0	0	0	0	3	1	0	0
	5	1	0	3	0	0	0	0	0	0	0	0	0
	6	0	0	0	0	0	0	0	0	0	0	0	0
	7	0	1	1	1	0	0	1	3	0	1	0	0
	8	0	0	0	0	0	0	0	0	0	0	0	0
	9	0	0	1	0	0	0	0	0	0	0	0	0
	10	0	0	0	0	0	0	0	0	0	0	0	0

## NCF 모델 적용 방안 연구

저번주 논문을 읽고 모델 공부를 한 내용을 바탕으로 프로젝트에 어떻게 적용시킬 것인가에 대한 논의 및 연구 진행

#### 1.논의 사항

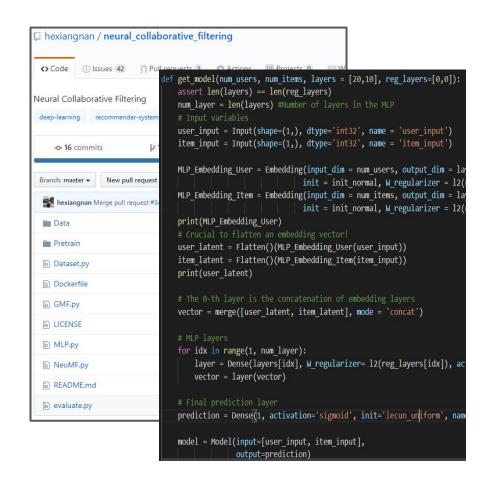
NCF 모델은 0과 1로 표현하여 **유저가 관심을 가질만한 아이템인가 아닌가** 에 집중, 우리가 현재 만든 아이템, 유저 matrix는 형용사 사용 빈도수 기반, 이를 어떻게 모델에 적용시킬 것인가?

- 방안 1) 기존 연구 모델에서는 activation, loss 함수에 각각 Sigmoid, binary cross entropy를 사용함. 이를 각각 ReLU, MSE로 변경하여 진행
- 방안 2) 아이템, 유저 matrix에서 빈도수를 binary형태로 변경 (만약, 한번이라도 리뷰에 사용된 형용사 그룹이라면 1로 구분)
- => 우선 NCF 모델을 변형없이 사용해보기 위해 2번 방안으로 진행하는 것으로 결정

#### 2.적용 사항

최종적으로 유저-장소별 관심 정도가 [0,1] 사이의 1차원 행렬의 형태로 output으로 나오게 된다. 이중 가장점수(확률)가 높은 장소들을 추려내고 이후에 최종 필터(위치, 데이트 예산 등) 적용.

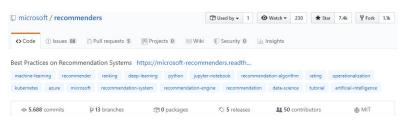
## NCF 모델 코드 연구



연구 중인 논문을 바탕으로 만들어진 코드를 테스트 (movie-lens, pinterest 데이터 이용)

- keras, tensorflow 버전 업그레이드에 따른 버전 수정 및 코드 수정
  - 에러가 많아 디버깅 중

## → NCF가 구현된 라이브러리를 사용하는 방안



## NCF 모델 테스트

- 1. 리뷰로부터 input으로 사용할 user matrix 생성
- 2. 모델을 구현해둔 코드 테스트



- 1. 클러스터링 모델 튜닝
- 2. NCF 모델 논문 구현 코드
- 3. NCF 모델 라이브러리
- (+) 각자 수집한 데이터 번역

#### NCF 라이브러리 사용한 결과

	userID	itemID	rating	timestamp
0	196	242	3.0	881250949
1	186	302	3.0	891717742
2	22	377	1.0	878887116
3	244	51	2.0	880606923
4	166	346	1.0	886397596



	userID	itemID	prediction		
0	1.0	149.0	0.150561		
1	1.0	88.0	0.307006		
2	1.0	101.0	0.346554		
3	1.0	110.0	0.185647		
4	1.0	103.0	0.002877		

```
def get_model(num_users, num_items, layers = [20,10], reg_layers=[0,0]):
                                             assert len(layers) == len(reg_layers)
                                             num_layer = len(layers) #Number of layers in the MLP
                                             user_input = Input(shape=(1,), dtype='int32', name = 'user_input')
def get_model(num_users, num_items, late
                                             item_input = Input(shape=(1,), dtype='int32', name = 'item_input')
   # Input variables
                                             MLP_Embedding_User = Embedding(input_dim = num_users, output_dim = layers[0]/2, name = 'user_embedding',
    user_input = Input(shape=(1,), dtype
                                                                           init = init_normal, W_regularizer = 12(reg_layers[0]), input_length=1)
    item_input = Input(shape=(1,), dtype
                                             MLP_Embedding_Item = Embedding(input_dim = num_items, output_dim = layers[0]/2, name = 'item_embedding',
                                                                           init = init_normal, W_regularizer = 12(reg_layers[0]), input_length=1)
    MF_Embedding_User = Embedding(input_
                                             # Crucial to flatten an embedding vector!
    MF_Embedding_Item - Embedding(input
                                              user_latent = Flatten()(MLP_Embedding_User(user_input))
                                             item_latent = Flatten()(MLP_Embedding_Item(item_input))
    # Crucial to flatten an embedding ve
                                             \# The \theta-th layer is the concatenation of embedding layers
    user_latent = Flatten()(MF_Embedding
                                             vector = merge([user_latent, item_latent], mode = 'concat')
    item_latent = Flatten()(MF_Embedding
    # Element-wise product of user and i
                                             for idx in xrange(1, num_layer):
    predict_vector = merge([user_latent,
                                                 layer = Dense(layers[idx], W_regularizer= 12(reg_layers[idx]), activation='relu', name = 'layer%d' %idx)
                                                 vector = layer(vector)
    # Final prediction layer
    #prediction = Lambda(lambda x: K.sig
                                             # Final prediction layer
    prediction - Dense(1, activation-'si
                                             prediction = Dense(1, activation='sigmoid', init='lecun_uniform', name = 'prediction')(vector)
    model = Model(input=[user_input, ite
                                             model = Model(input=[user_input, item_input],
                output-prediction)
                                                          output=prediction)
```

return model

return model