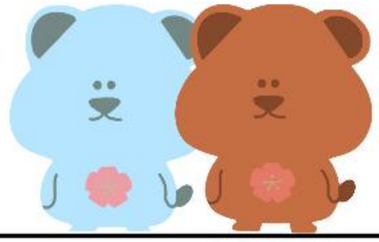
Deep Neural Networks for YouTube Recommendations(2016, Paul Covington)

손소현

EWHA EURON

목차



EWHA EURON

#00 추천시스템 기본 개념

 # 콘텐츠 기반 필터링 (Content-Based Filtering)
: 아이템 자체의 속성과 사용자의 선호도를 기반으로 추천을 제공하는 방식
사용자의 개인적인 취향과 아이템의 특성을 고려할 수 있으며, 새로운 아이템을 추천하는 데에도 유용.
but 아이템 속성을 수집하고 정리하는데 비용과 시간이 듦

 # 협업 필터링 (Collaborative Filtering)
: 사용자들의 행동이나 선호도를 기반으로 추천을 제공
사용자 행동에 대한 정보만을 기반으로 추천이 이루어지기 때문에 아이템이나 사용자의 특성에 대한 사전 지식이 필요하지 않음

Matrix Factorization(MF)

User-Item Matrix

	영화1	영화2	영화3
유저1	1	1	0
유저2	0	1	1
유저3	0	1	0

N x M

≈

User Matrix

	재미	감동	티켓값
유저1	?	?	?
유저2	?	?	?
유저3	?	?	?

N x K

×

Item Matrix

	영화1	영화2	영화3
재미	?	?	?
감동	?	?	?
티켓값	?	?	?

K x M

#00 추천시스템 기본 개념

 # 콘텐츠 기반 필터링 (Content-Based Filtering)
: 아이템 자체의 속성과 사용자의 선호도를 기반으로 추천을 제공하는 방식
사용자의 개인적인 취향과 아이템의 특성을 고려할 수 있으며, 새로운 아이템을 추천하는 데에도 유용.
but 아이템 속성을 수집하고 정리하는데 비용과 시간이 듦

 # 협업 필터링 (Collaborative Filtering)
: 사용자들의 행동이나 선호도를 기반으로 추천을 제공
사용자 행동에 대한 정보만을 기반으로 추천이 이루어지기 때문에 아이템이나 사용자의 특성에 대한 사전 지식이 필요하지 않음 **-> 얘를 딥러닝과 결합하는 방식을 설명하는 논문**

Matrix Factorization(MF)

User-Item Matrix

	영화1	영화2	영화3
유저1	1	1	0
유저2	0	1	1
유저3	0	1	0

N x M

≈

User Matrix

	재미	감동	티켓값
유저1	?	?	?
유저2	?	?	?
유저3	?	?	?

N x K

×

Item Matrix

	영화1	영화2	영화3
재미	?	?	?
감동	?	?	?
티켓값	?	?	?

K x M

#01 Introduction

EWHA EURON

#01 Introduction

엄청나게 큰 비디오 코퍼스부터 개인화 추천을 제공함

다음 세가지 특징때문에 extremely challenging

- scale : 많은 이미 존재하는 추천알고리즘은 작은 문제에는 잘 작동하지만, 구글의 큰 규모단위의 문제에는 실패. 유튜브의 아주 방대한 유저베이스와 비디오 코퍼스를 다루기 위한 효과적인 알고리즘이 필요
- freshness : 유튜브에는 초당 시간단위의 비디오가 업로드되는 아주 다이나믹한 코퍼스임. 그래서 추천시스템은 새롭게 업로드된 비디오도 반영해야 함.
- noise : 희소성과 관찰할 수 없는 외부요소들 때문에, 유튜브 사용자의 행동은 예측하기 어려움. 사용자는 피드백을 거의 남기지 않고, 영상에 대한 메타데이터도 온톨로지도 없고 형편없음. 그래서 추천시스템은 훈련 데이터의 이러한 특징에도 잘 작용해야 한다. (need to be robust)

#01 Introduction

(2016년 당시) 거의 모든 문제들에 대해 딥러닝을 일반적인 해결방법으로 사용하는 방향으로 변하고 있었음.
이 모델은 거의 10억개의 파라미터와 수천억의 데이터를 가지고 훈련되었음

(2016년 당시) 행렬분해 논문과 대조적으로 **추천시스템에 딥러닝을 사용한 논문은 상대적으로 거의 없었음**
(neural network, 협업필터링, 크로스 도메인 사용자 모델링, 콘텐츠 기반 필터링 등을 제안한 논문 언급하고 지나감)

#02 System Overview

EWHA EURON

#02 System Overview

크게, 후보군 생성과 랭킹으로 나누어짐

후보군 생성 (Candidate Generation)

- 사용자의 유튜브 활동 기록을 인풋으로 받아서 큰 비디오 코퍼스에서 작은 (수백개) 비디오 코퍼스를 아웃풋으로
- 사용자와 높은 정확도로 연관되어 있음
- 협업필터링을 통해 광범위한 personalization
- 사용자간 유사도는 비디오의 id, 검색기록, 인구통계학과 같은 피처로 계산됨

랭킹

- 높은 수준의 정확도를 요구하기 때문에
- 그림에서 나온 피처정보를 이용하여 desired objective function에 따라 비디오에 점수를 매김

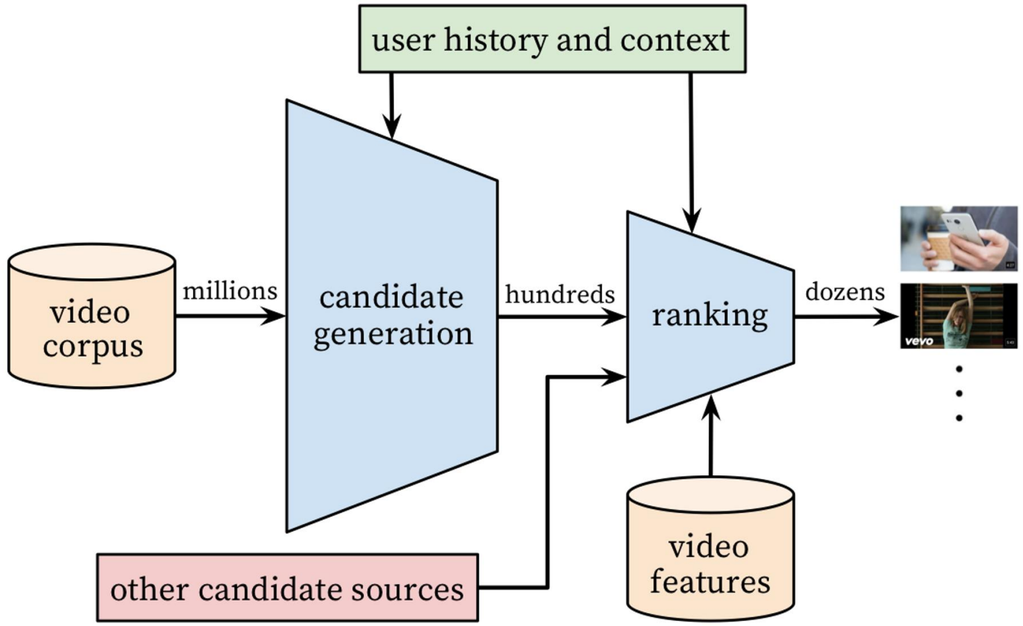


Figure 2: Recommendation system architecture demonstrating the "funnel" where candidate videos are retrieved and ranked before presenting only a few to the user.

#02 System Overview

이렇게 2단계로 나누어진 추천은 큰 비디오 코퍼스에서 기기(앱화면)에 표시될 아주 작고, 특정한 비디오를 뽑게 해 줌

- 이런 구조는 다른 소스(2010년에 나온 첫번째 논문 참고)를 가지고 후보군들을 섞게(blending)도 할 수 있음

offline 행렬(precision, recall, ranking loss) 쓰기도 했지만, 결국에 효과적인 모델을 만들기에 사용한 것은, 실시간 피드백을 통한 a/b테스트

실시간 피드백에서는 클릭률, 시청시간 등과 같은 미묘한 변화들을 측정할 수 있었음

실시간 피드백 결과가 offline 행렬과 항상 일치 하진 않음

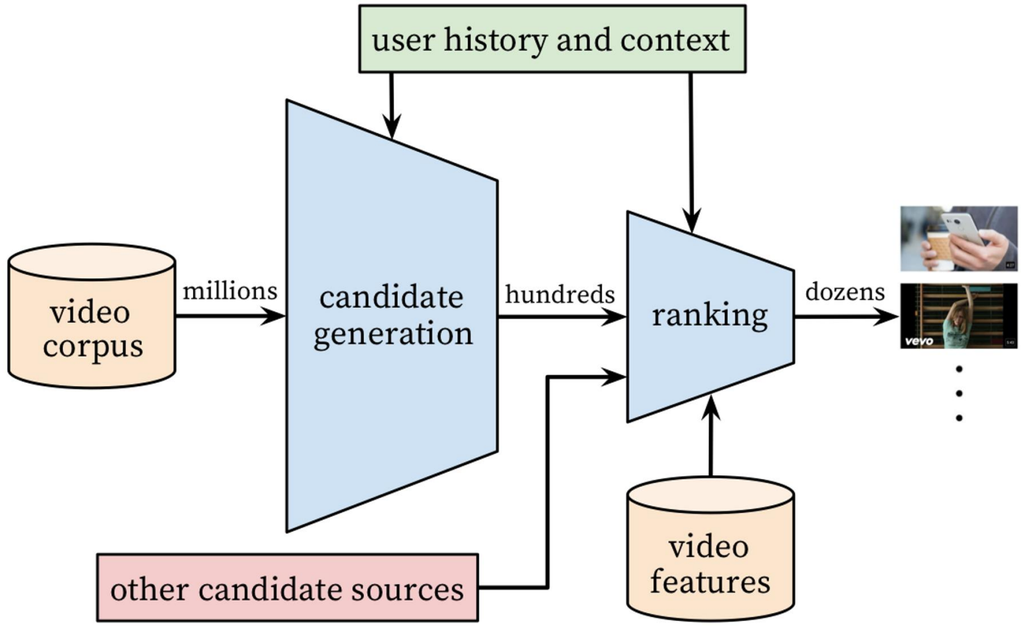
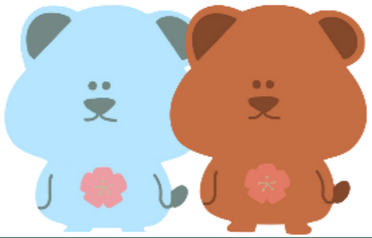


Figure 2: Recommendation system architecture demonstrating the "funnel" where candidate videos are retrieved and ranked before presenting only a few to the user.

#03 Candidate Generation

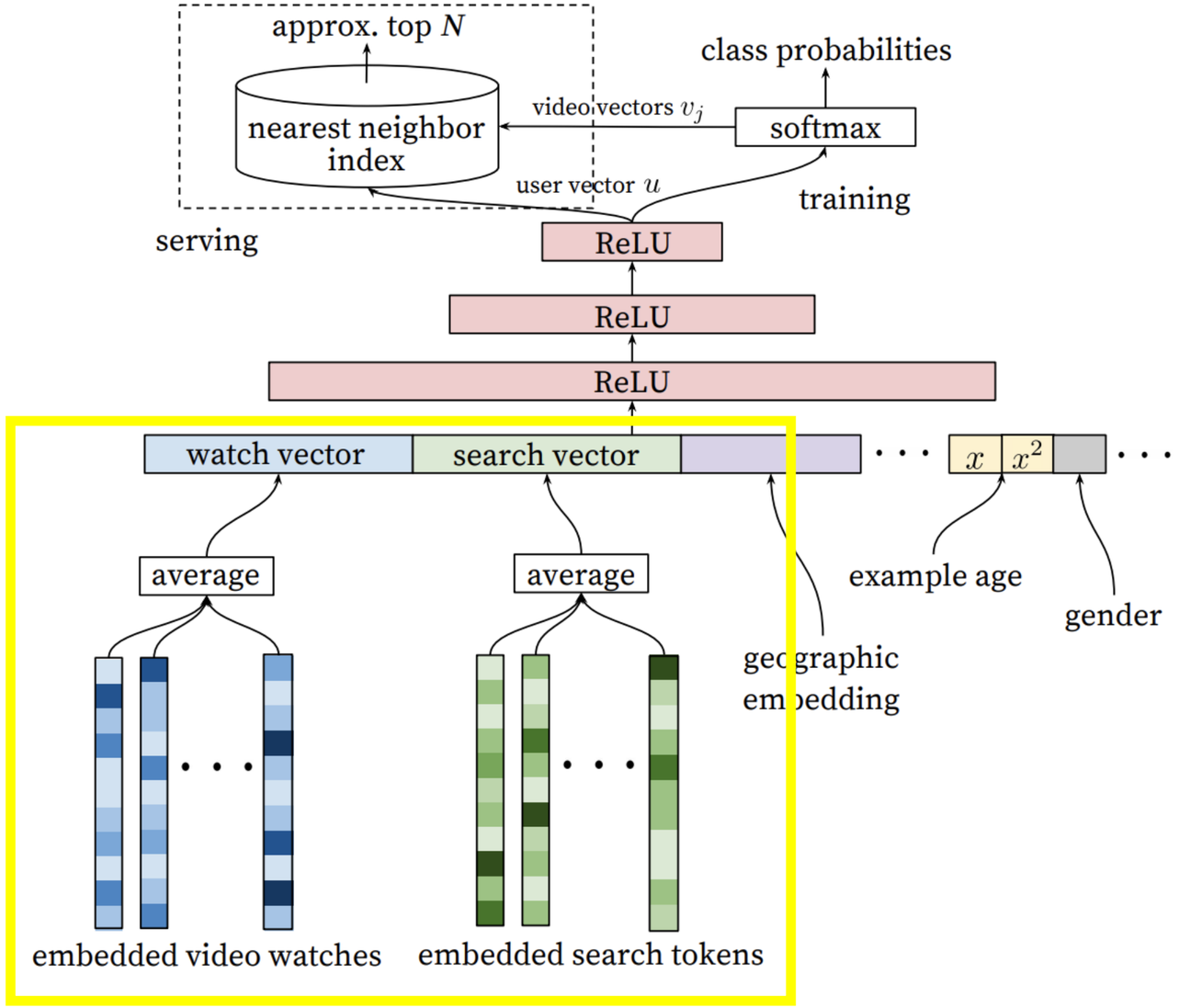


EWHA EURON

#03 Candidate Generation

이 단계는 방대한 비디오 코퍼스에서 몇백개의 영상으로 후보군을 생성하는 단계임

이전의 추천 시스템은 rank loss를 바탕으로 만든 행렬분해 관점이고, 간단한 신경망을 사용한 적 있지만 이때는 사용자가 과거에 본 비디오 내역만 이용했었음



#03 Candidate Generation

3.1 recommendation as classification
사용자(U)와 Context(C)를 기반으로 특정 시간(t)에서 수백만개의 아이템(V) 중 각 아이템(i)의 시청 class를 예측하는 extreme multiclass classification으로 추천 문제를 정의

$$P(w_t = i|U, C) = \frac{e^{v_i u}}{\sum_{j \in V} e^{v_j u}}$$

여기서 dnn의 임무는 사용자임베딩을 사용자 히스토리 함수로 학습하고, softmax classifier로 비디오를 분류하는데 도움되는 context를 학습하는 것

여기서 wt는 특정시간 t에 본 비디오 시청
i는 클래스
U는 사용자 임베딩
v 비디오 코퍼스 임베딩
i는 클래스
C context

유튜브에 명시적 피드백(좋아요, 싫어요 등)이 있지만, 엄청 sparse 해서 시청시간, 완료 등을 활용한 암시적 피드백을 사용

#03 Candidate Generation

extremely multiclass classification

•offline

수백만개의 클래스를 훈련시키기 위해 negative sampling을 시도

중요도 가중치를 이용해서 샘플링을 수정

hierachical softmax는 비교할만한 정확도를 얻을 수 없어서 (+ 다른 이유도 있고) 사용하지 않음

•at serving time (실시간)

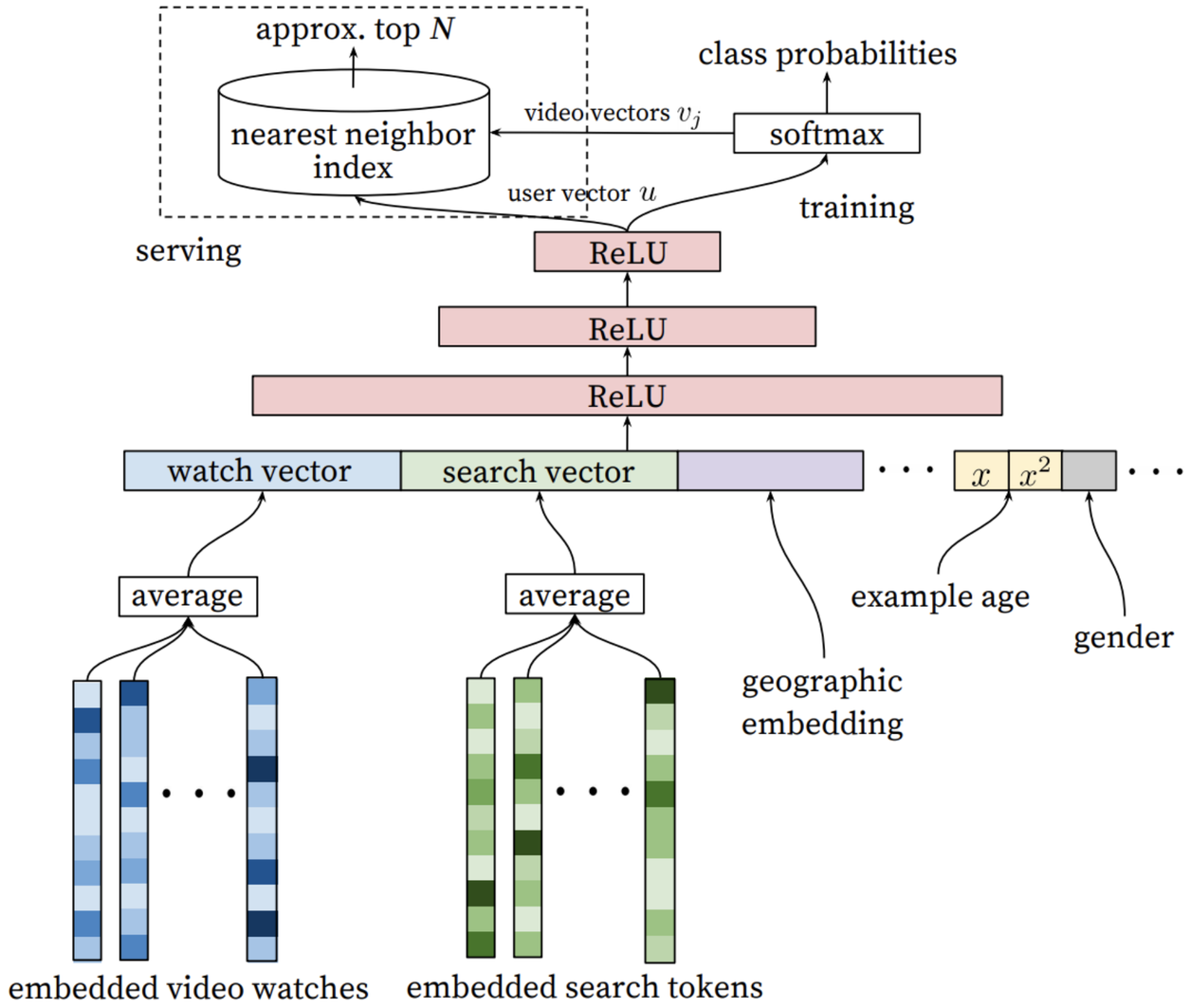
사용자가 볼 것 같은 비디오 N개 뽑기 : user vector u + 학습된 video vector v_j

이전 유튜브 시스템은 hashing 사용했었고, 이번에도 사용

결국 dot product space에서 nearest neighbor을 찾는 과정

A/B테스트 결과에는 큰 차이가 보이진 않아서 결과를 빨리 내는 게 중요

#03 Candidate Generation



3.2 model architecture

bag of words 언어모델에 영향을 받아 비디오를 고정된 단어로 임베딩하고 이 임베딩을 신경망에 넣음

고정된 차원이 요구되기 때문에 단순하게 시청아이템 벡터의 '평균' 을 적용
(sum, component-wise max등 사용했으나 평균이 가장 좋음)

임베딩은 파라미터들과 조인됨. 그림을 보면 하나의 넓은 레이어가 ReLU와 연결되는 것을 볼 수 있음

#03 Candidate Generation

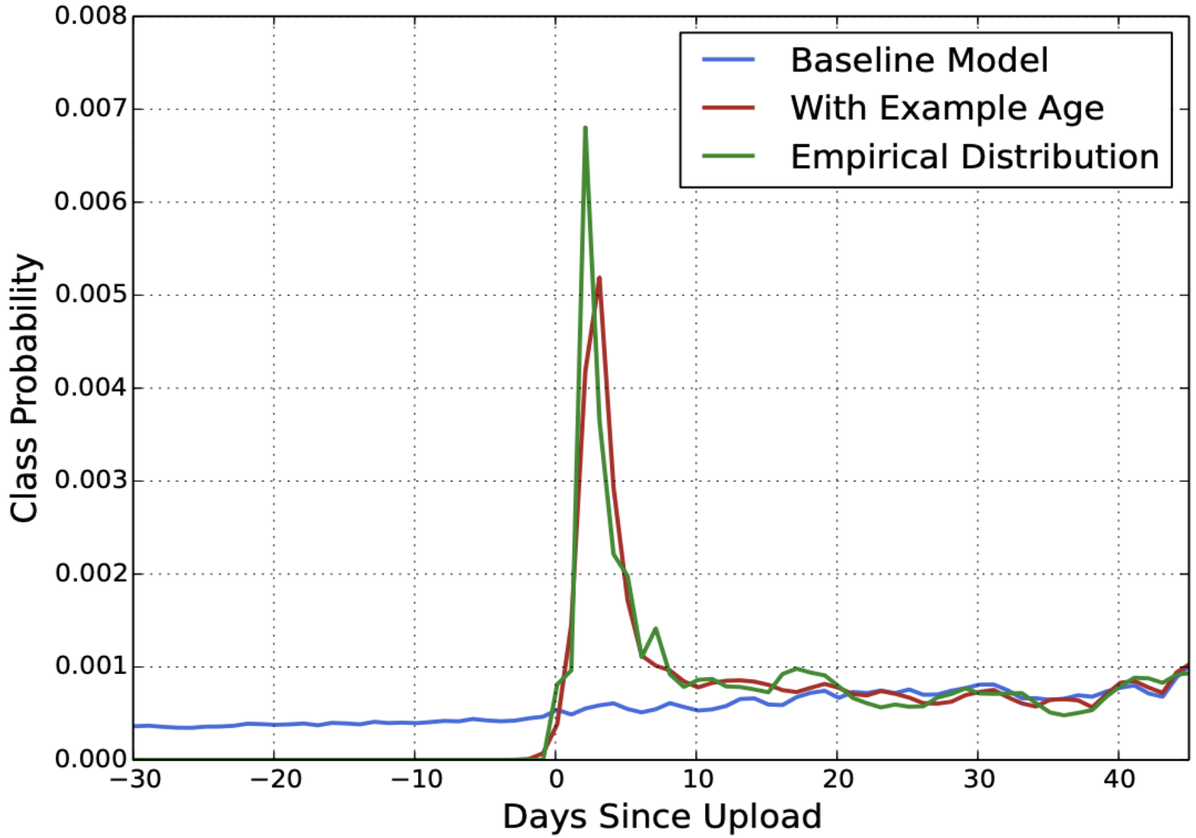
3.3 heterogeneous signals

- dnn을 행렬분해의 일반화로 사용하는 것의 주요 장점은 연속형 변수와 범주형 변수를 모델에 쉽게 넣을 수 있다는 것임
- 검색 기록은 시청 기록과 비슷하게 취급
- 일단 평균내게 되면 임베딩된 검색 질문(queries)들은 '검색기록(search history)' 으로써 요약됨
- 인구통계학적 정보(지역, 나이, 성별 등)는 새로운 사용자가 왔을 때 추천하는데에 유용함

'Example Age' Feature

- 유튜브 영상 추천에서는 fresh한 비디오가 extremely important
- 유튜브 사용자는 크게 관련이 없더라도, fresh한 비디오를 선호한다는 것을 관찰
- 단순히 사용자가 보고싶은 새로운 영상을 추천해주는 1차 효과 뿐만 아니라, 바이럴 콘텐츠를 전파하는 2차 현상도 있음
- 그렇지만 인기도 분포를 보면, 비디오 코퍼스의 분포는 몇주 평균을 반영, (= 시간적 sequence가 없음)
- 이것을 고치기 위해 example age (비디오 나이)를 고려

#03 Candidate Generation



이 그래프를 보면 베이스보다는, 비디오 나이를 고려한 게 성능이 훨씬 좋고,
경험적 분포(Empirical Distribution)를 넣은 게 성능이 좀 더 좋다는 것을 알 수 있음.

#03 Candidate Generation

3.4 label and context selection
추천은 surrogate problem. 즉 다른 맥락의 문제도 해결가능
예를 들어 영화평점 예측 알고리즘은 영화추천으로도 사용 가능
유튜브에서는 어떻게 활용할 수 있을지 살펴보자

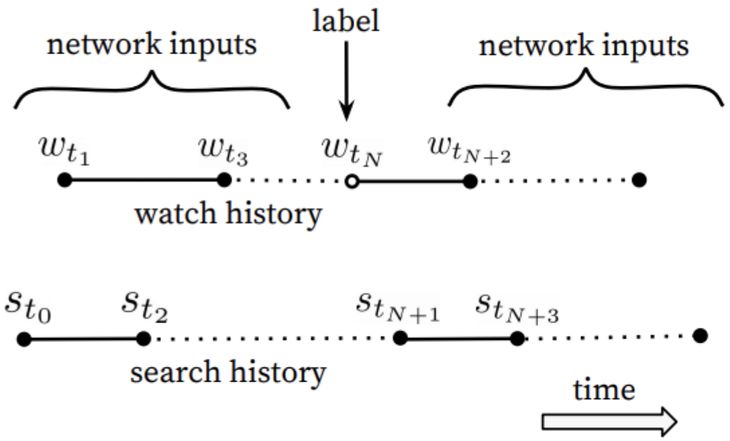
1) 학습 데이터 :
- 유튜브를 통해 본 **모든 기록(외부사이트 포함)**으로부터 만듦
  (유튜브에서 이미 추천해서 본 것들은 빼고 ← 새로운 콘텐츠가 나오기 어렵고, 추천시스템이 편향된 결과를 만들것이기 때문)
- 사용자가 추천시스템 이용하지 않고 비디오를 탐색하면, 이 탐색을 최대한 빠르게 다른 사용자에게
  '협업필터링' 을 통해 보급 (⇒ 딥러닝 말고 다른 방법도 적극 활용)
- 이용자별 감상 횟수를 제한. 엄청나게 많이 보는 사람들의 영향을 빼기 위해
- 추천 결과나 검색 결과를 즉시 활용하지 않음
- 검색키워드는 일부러 순서를 날린 bag-of-tokens사용.
  그렇지 않으면 검색한 내용이 계속 메인에 떠서 비효율적 (테일러를 예시로~)

#03 Candidate Generation

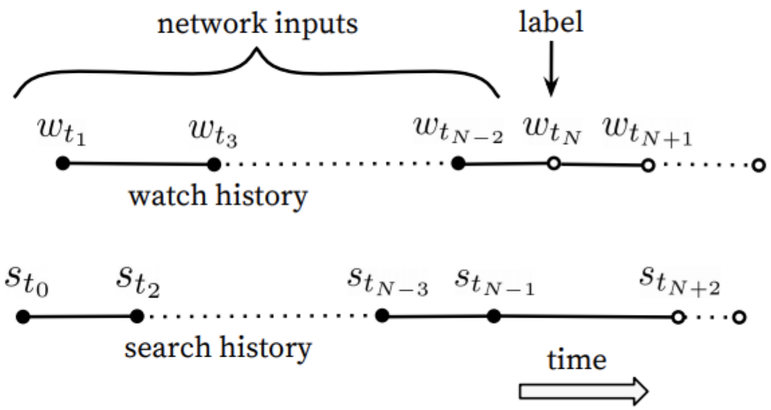
3.4 label and context selection

2) Episodic series

- 장르 Episodic series한 방식으로 비디오를 비대칭적으로 시청 ex. 장르4-3-2-1-3-4-3-2-1 이런식으로
- 또 넓게 인기있는 범위(broadly popular)에서 보다가 작은 범위(smaller niches)로 시청
- 그래서 무작위로 추천하는 것 대신, 사용자가 '**바로 다음**'에 볼 것 같은 걸 예측하는 것이 성능이 좋음



(a) Predicting held-out watch

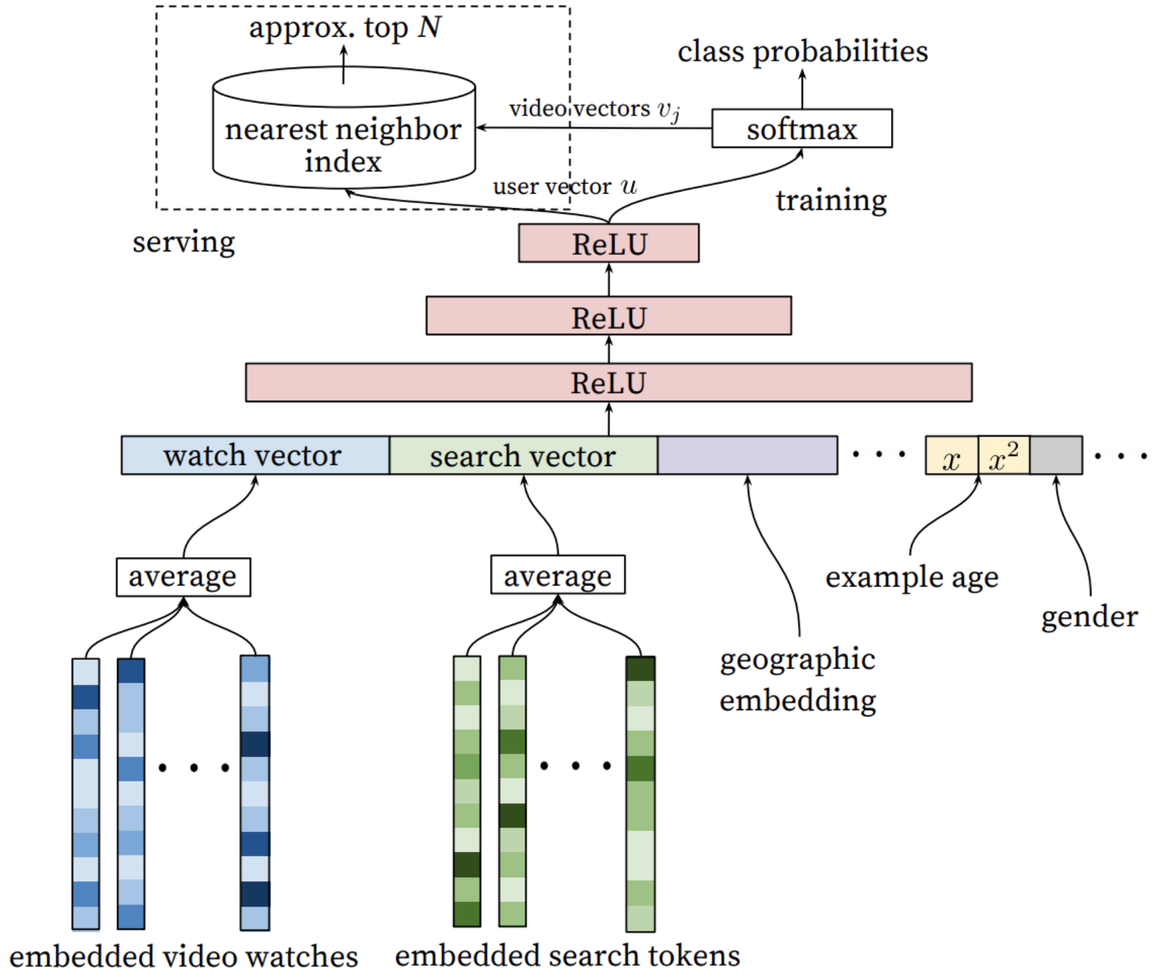


(b) Predicting future watch

#03 Candidate Generation

3.5 experiments with features and depth

점점 좁은 쪽으로 통과시키는 Fully connected "Tower"형태.

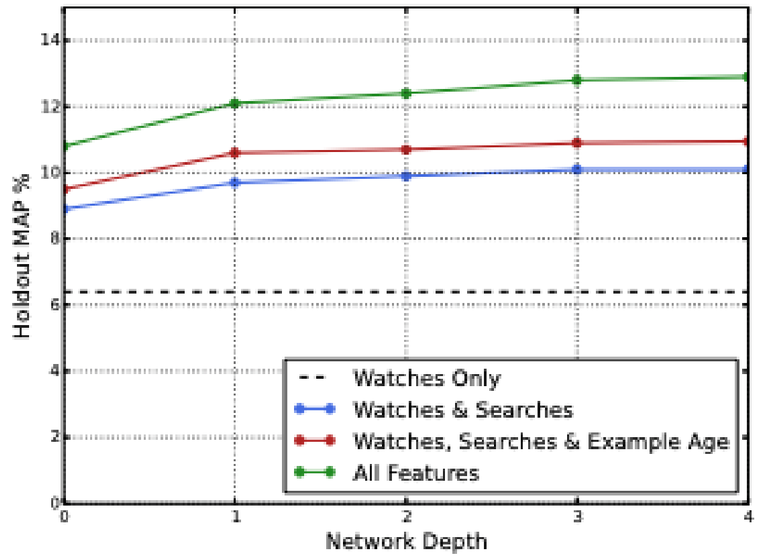


#03 Candidate Generation

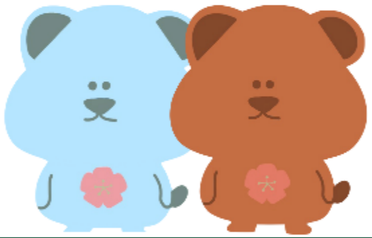
3.5 experiments with features and depth

여기서 실험적으로, **Layer는 더 넓게 깊이 쌓는 것이 더 좋은 성능**을 내는 걸 확인

- Depth 0: A linear layer simply transforms the concatenation layer to match the softmax dimension of 256
- Depth 1: 256 ReLU
- Depth 2: 512 ReLU → 256 ReLU
- Depth 3: 1024 ReLU → 512 ReLU → 256 ReLU
- Depth 4: 2048 ReLU → 1024 ReLU → 512 ReLU → 256 ReLU



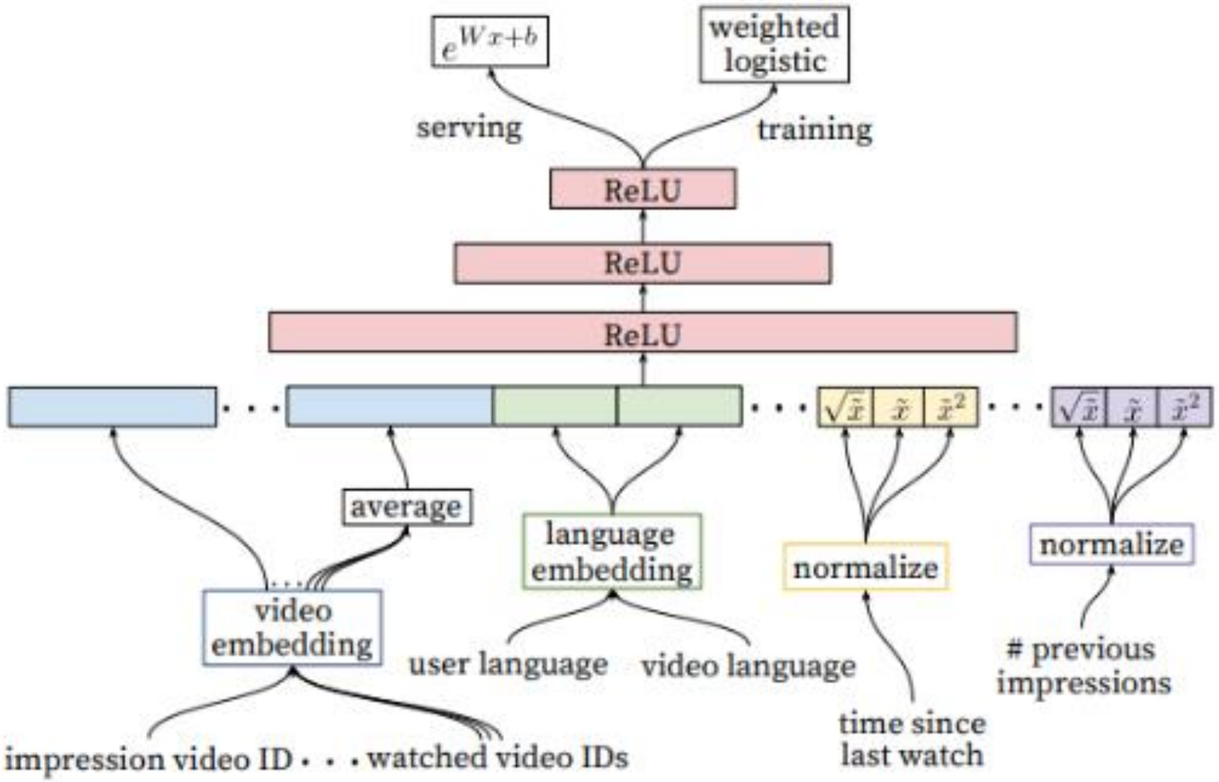
#04 Ranking



EWHA EURON

#04 Ranking

- ranking 모델의 주 역할은 각 사용자의 feature를 사용해서 후보 아이템을 특성화하고 조정하는 것임
- 후보군 생성 모델과 구조 자체는 유사하며, 각 아이템에 score(점수)를 줘서 정렬함으로써 사용자에게 제공
- 후보군 생성 모델과 비슷한 구조



#04 Ranking

4.1 feature representation

feature engineering

- 수백개의 feauture가 사용
- 딥러닝을 통해 학습한 feature 뿐만 아니라 hand written 전처리가 필수
- Input으로 들어가는 영상의 개수가 줄어들었기 때문에 추가적으로 유저 & 영상간 관계 파악하기 위해 피처들을 추가 (이런 피처는 particularly powerful)
- 해당 비디오의 채널과 사용자의 과거시청내역을 고려 - 이채널에서 몇개를 봤는지, 언제 마지막으로 봤는지
- 또 어떤 요소가 해당 비디오 후보군에 넣었는지도 고려하는 것이 crucial함
- 추천했는데 사용자가 안 볼 경우, 격하시킴

Embedding Categorical Features

- categorical feature의 경우에는 binary 데이터, 사용자의 마지막 검색 기록, 점수를 부여할 아이템 ID, 최근 본 N개의 아이템 ID이 있음
- categorical 데이터가 지나치게 많을 경우, click의 빈도수를 기반으로 top N을 선정
- 반대로 데이터가 부족한 경우에는 zero 임베딩을 함
- candidate generation 과정처럼 multivalent feature(최근 본 N개의 아이템)의 경우에는 평균을 적용함

Normalizing Continuous Feature

- 값 x를 [0,1) 에 들어오도록 스케일링
- x뿐만 아니라 x**2, sqrt(x)도 다 넣음

#04 Ranking

4.2 modeling expected watch time

- positive(비디오 클릭o) negative(비디오 클릭x)
- positive면 그 비디오를 얼마나 봤는지 기록
- weighted logistic regression를 이용해 시청 시간 예측

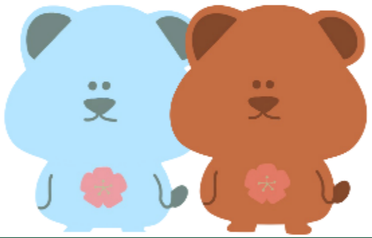
4.3 experiments with hidden layers

- 실제로 보지 않은 비디오가 높은 점수를 받았었다면, 예측 시간이 틀렸다고 간주
- 깊고 넓을수록 성능이 잘 나옴

Hidden layers	weighted, per-user loss
None	41.6%
256 ReLU	36.9%
512 ReLU	36.7%
1024 ReLU	35.8%
512 ReLU → 256 ReLU	35.2%
1024 ReLU → 512 ReLU	34.7%
1024 ReLU → 512 ReLU → 256 ReLU	34.6%

Table 1: Effects of wider and deeper hidden ReLU layers on watch time-weighted pairwise loss computed on next-day holdout data.

#05 Conclusions



EWHA EURON

#05 Conclusion

- deep collaborative filtering 모델은 기존의 방법(matrix factorization)보다 성능을 많이 향상시킴
- "영상의 나이(Example age)"가 성능을 크게 개선
- 딥러닝은 비선형성을 통해 효율적으로 표현하는데 효과적이다. 그러나 랭킹 단에서는 전통적인 ML에 가까움
- positive이면 시청시간 예측에 가중치를 두고(=시청시간으로 대체),
  Negative이면 unity(통일성)에 가중치를 주는 것은(=0으로 통일하는 것)
  CTR(Click Through Rate 클릭율)보다 효과적이다.

#05 Conclusion

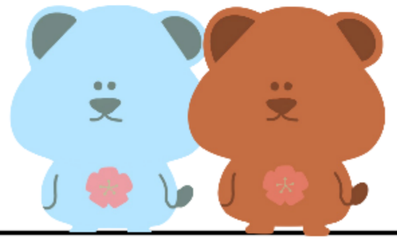
최근 업데이트

- 2019년 발표
- multi-gate mixture-of-expert(MMoE)를 통해 multitask learning을 구현하고자
- 다음에 뭘 시청할지 에 집중하면 광고나 낚시형 영상에 끌릴 확률 높아짐
- 이걸 MMoE를 통해 해결

소감

- 유튜브 코퍼스의 방대함이라는 특징때문에 2단계로 나누는 것 발상이 인상적
- Ranking단계에서 x뿐만 아니라 x**2, sqrt(x)도 다 넣는 게 신기
  → 아마 ml이면 다중공선성 때문에 그렇게 하면 안 될 것 같은데, 아마 딥러닝이라서 가능한듯

THANK YOU



EWHA EURON