Deep Neural Networks for YouTube Recommendations

Paul Covington, Jay Adams, Emre Sargin

KAIST/BTM & ISysE/20190552

장산하

INDEX

1. Introduction

1. Introduction

Youtube의 목표

- 사용자가 보고 싶어하는 개인화된 동영상을 추천

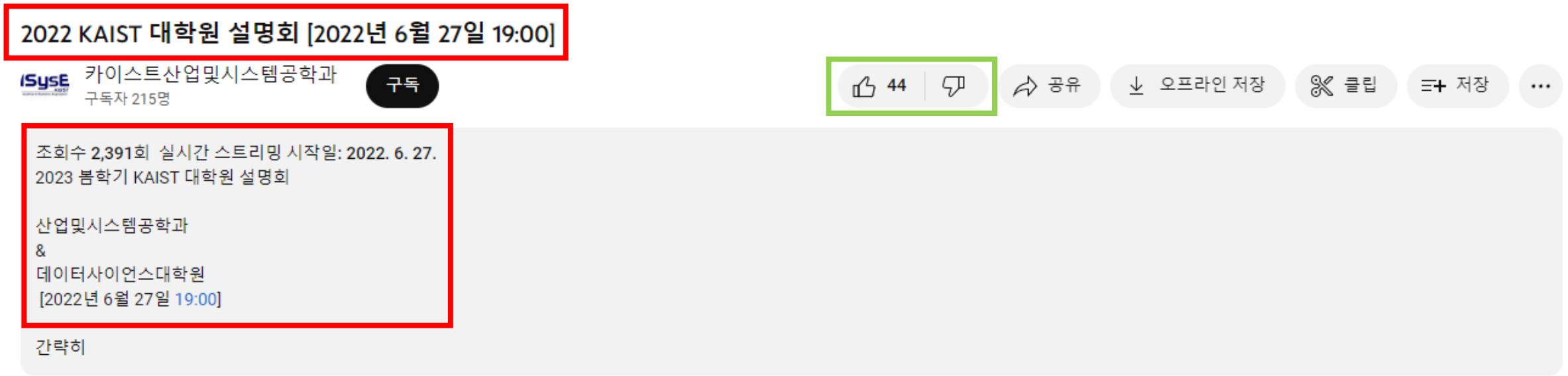
Youtube의 Challenge

- Scale
  - 엄청난 양의 데이터 및 적용하기 힘든 기존의 추천 알고리즘
- Freshness
  - 새롭게 생성되는 Contents와 User의 Actions을 즉각 반영해야 함
- Noise
  - 낮은 Meta data 퀄리티, Implicit Feedback 위주 데이터

1. Introduction

유튜브의 Meta Data란

제목, 설명, 태그, 자막, 카테고리 등의 동영상에 대한 모든 정보



1. Introduction

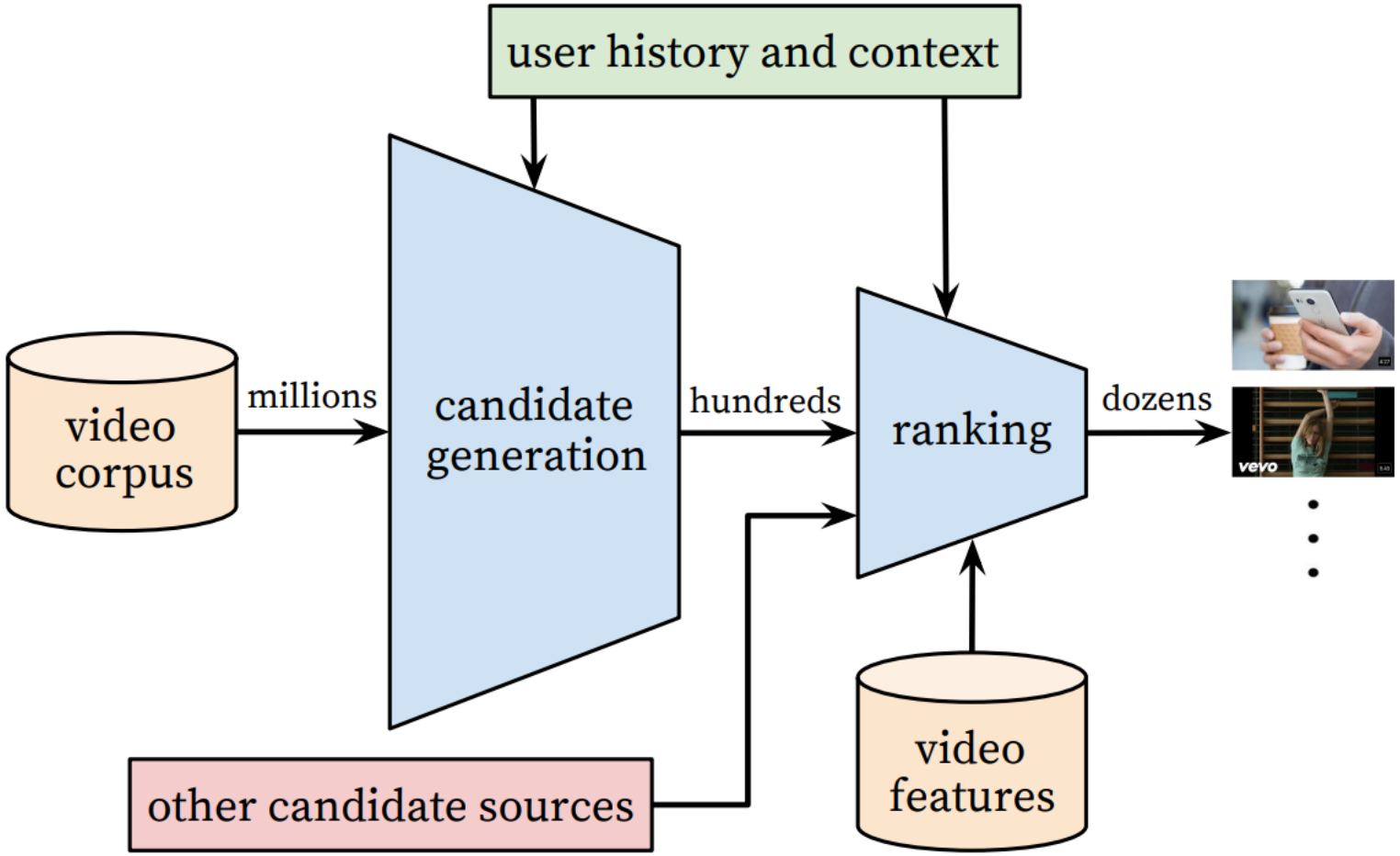
The Youtube Video Recommendation System(RecSys 2010)

- 특별한 의도가 없는 즉, 킬링 타임을 위한 유저에게 컨텐츠를 추천
- Candidate Generation & Ranking 방식
- NN을 사용하지 않고 Relatedness Score, Video Quality, User's Activation Log 등 사용
- Batch 방식으로 유저별 추천 리스트를 미리 계산 후 유저별로 저장해둔 리스트를 사용 시 불러와 서빙
- 하루에도 여러 번 잦은 업데이트를 함

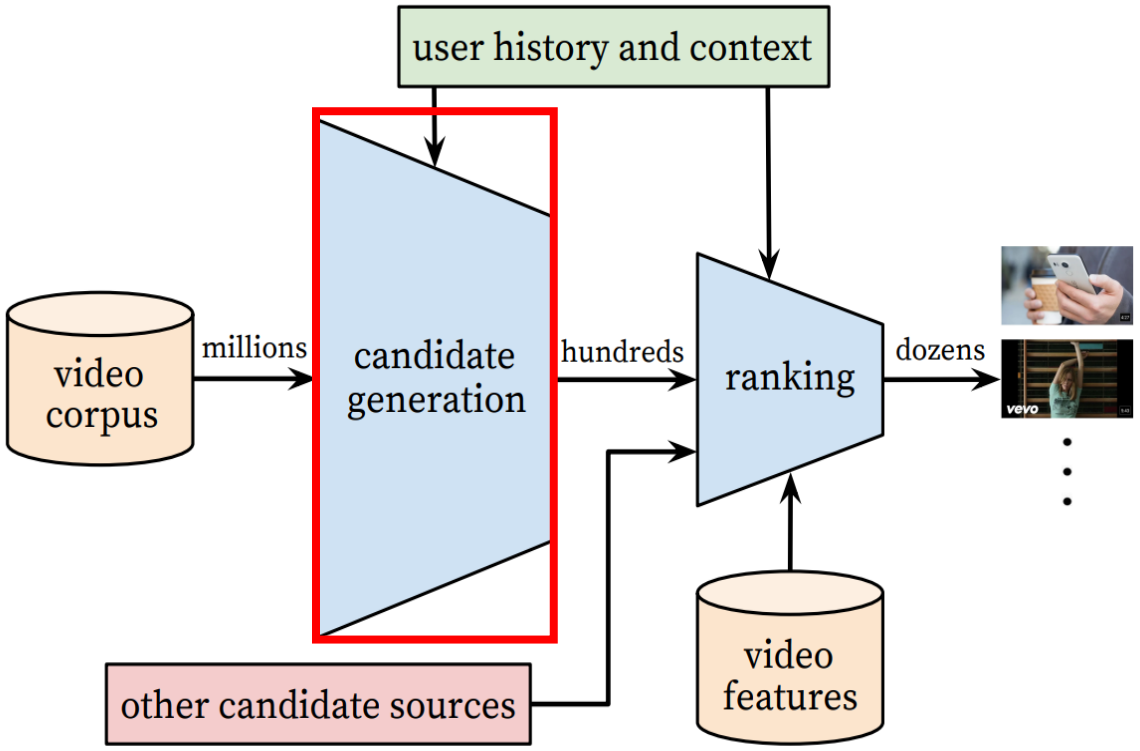
DNN을 활용한 추천 시스템 제작

2. Overview

2. Overview


user history and context
video corpus
millions
candidate generation
hundreds
ranking
dozens
vevo
other candidate sources
video features

2. Overview

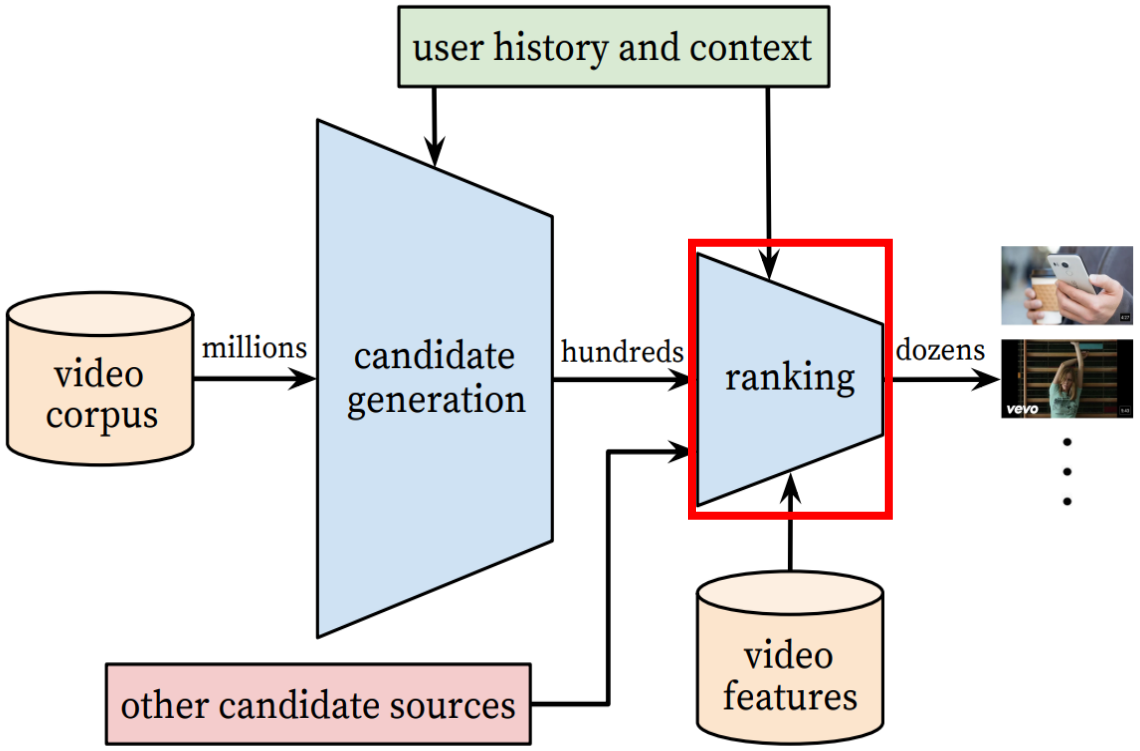


Candidate Generation

- User history와 Context 사용
- Millions → Hundreds
- Broad Personalization via Collaborative Filtering
- Video 및 검색 기록, Demographics

방대한 User수로 인해 Cold-Start 문제 해결

2. Overview



Ranking

- User history와 Context, Video features, Other Sources 사용
  - Candidate Generation보다 더 많은 feature 사용
- Hundreds → Dozens
- 후보군에 있는 비디오별 Score 계산 후 Best 추천 리스트 제공

2. Overview

- 개발 과정에서는 Precision, Recall, Ranking Loss 등 offline metrics를 구축하여 성능을 향상 $Precision = \frac{TP}{TP + FP}$ $Recall = \frac{TP}{TP + FN}$
- 하지만, A/B 테스트를 통해 실제 환경에서 효율성을 테스트하며 이 때, Click-through rate, 시청 시간 등과 지표를 사용
- Offline metrics 결과와 A/B 테스트 결과가 항상 일치하는 것은 아니기에 실제 환경에서 이뤄진 A/B 테스트 결과로 최종 알고리즘을 결정

3. Candidate Generation

3.1 Recommendation as extreme multiclass Classification

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

- U: User 정보
- C: Context 정보
- $u \in R^n$: User 정보와 Context 정보를 조합한 Embedding
- $v \in R^n$: 각 영상들의 Embedding
- $w_t = i$는 videos들 중 i를 time t에 본다는 것을 의미

DNN을 통해 u를 학습

3.1 Recommendation as Classification

u, v는 Dense Vector로 Embedding

- Dense Representation은 사용자가 설정한 값으로 데이터의 차원을 맞춤
- One-hot vector는 너무 Sparse space로 Embedding 됨

유튜브에는 Explicit Feedback과 Implicit Feedback 모두 존재

- Explicit Feedback: 좋아요/싫어요, 설문조사, 공유 등

But, 모델의 Bias와 Cold-start 문제를 방지하기 위해 Implicit Feedback 사용

또한, 영상을 끝까지 본 경우를 positive example로 지정

3.1 Efficient Extreme Multiclass

Softmax Classification에서 클래스의 수에 따라 계산량이 기하급수적으로 증가

↓

Negative Sampling을 통한 효율적 학습

Cross-entropy Loss Minimize with True and Neg-class

사용자에게 Top N개의 아이템을 계산 후 추천하기에 Serving Time이 문제

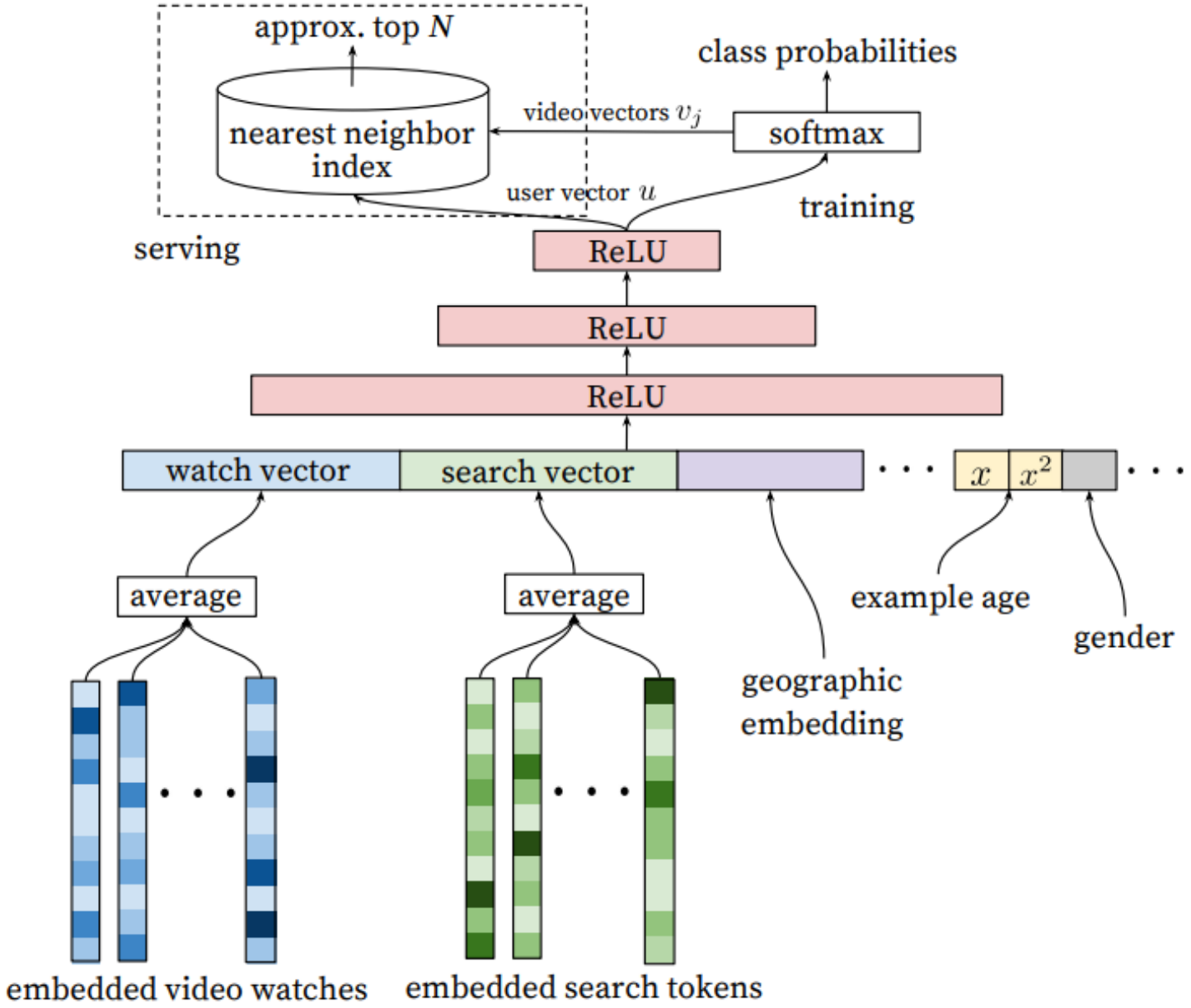
↓

Nearest Neighbor Search 알고리즘 사용

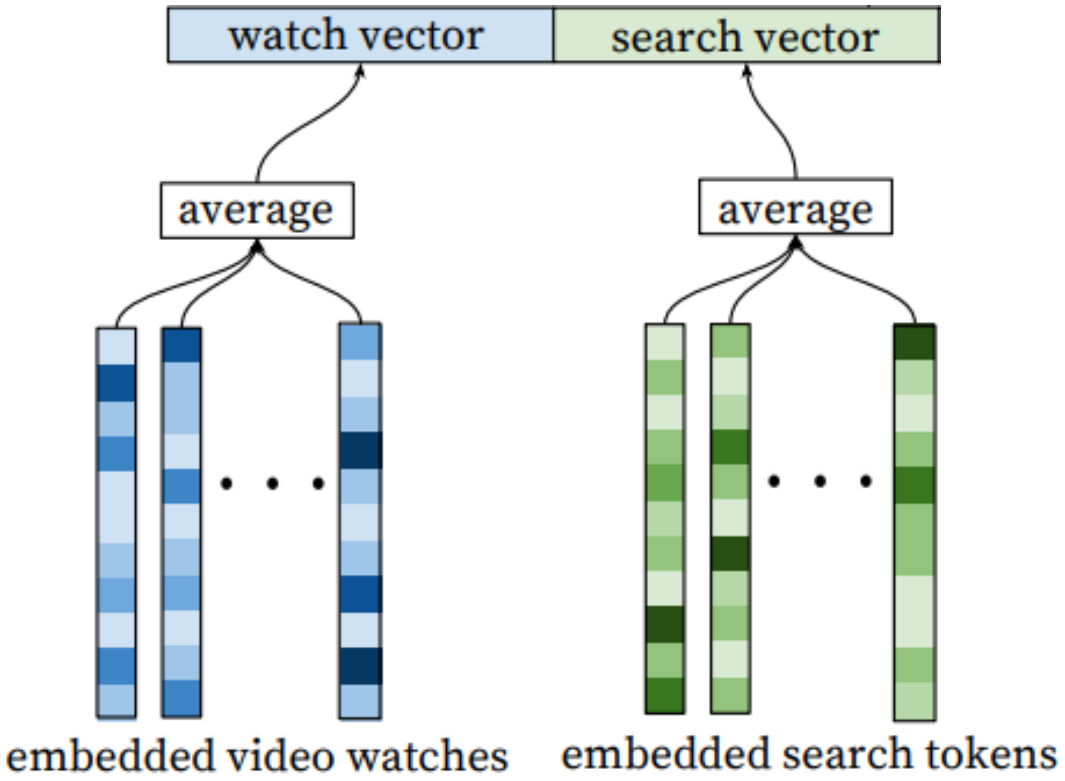
Serving Time을 매우 줄이고, 정확도는 큰 차이가 없음

(A/B 테스트 결과)

3.2 Model Architecture

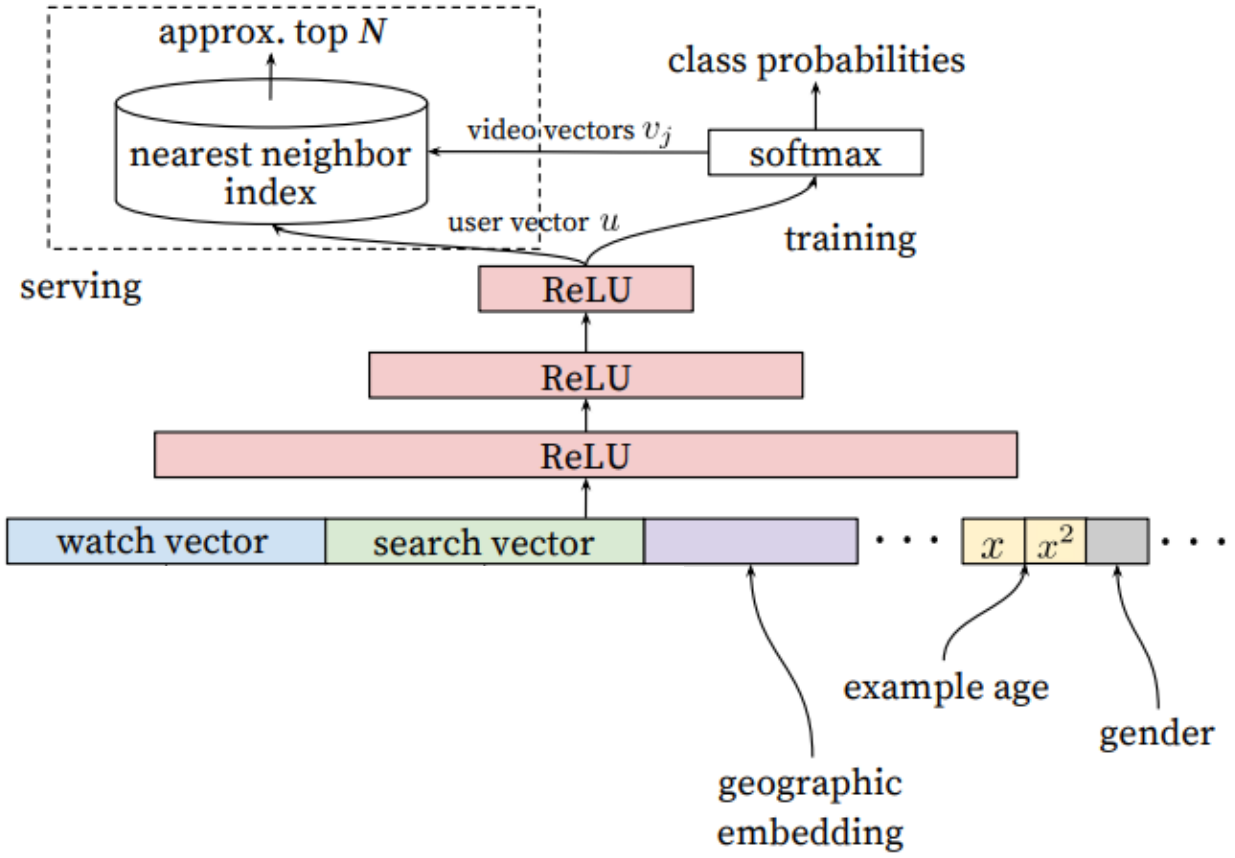


3.2 Model Architecture



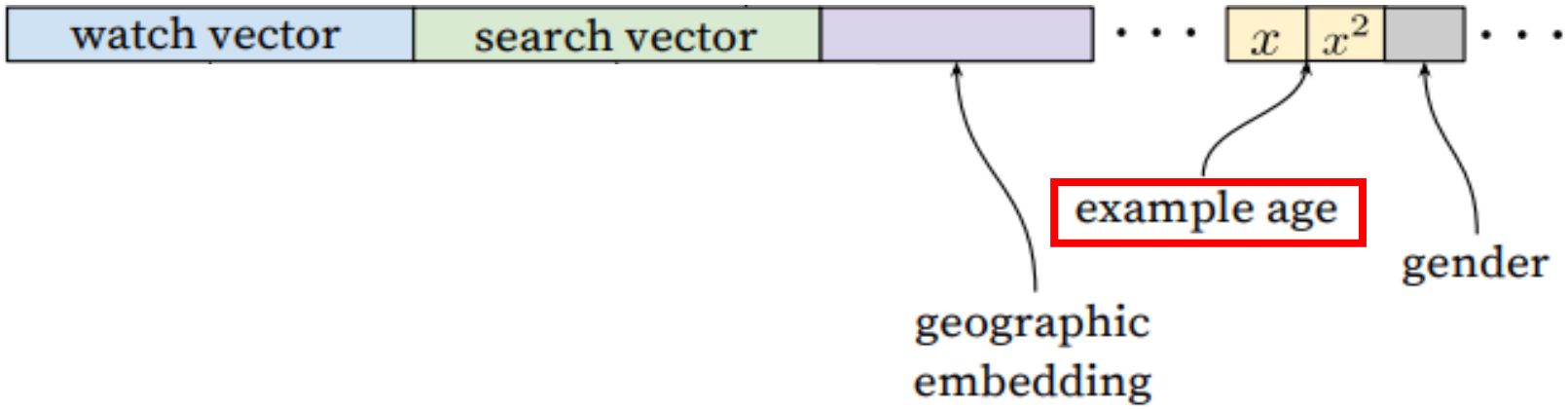
- 각 유저의 시청 이력과 검색 이력을 각각 Dense Embedding
- CBOW에 영감을 받아, Fixed-Vocabulary안에서 각 비디오에 대한 고차원의 임베딩을 학습
- Average를 통해 Input 사이즈로 변환
  - Sum, Concat 등 다양한 변환 방법 중 가장 우수한 성능
  - 이를 통해 마지막 검색어에 대한 Importance를 희석시켜줄 수 있음

3.2 Model Architecture



- Watch vector, Search vector, Geographic Embedding, Example age, gender 등을 모두 Concat한 Input 제작
- 여러 Fully Connected ReLu의 Layer **Tower**를 통과
- Gradient Descent Backpropagation update를 통한 DNN
- Output으로 user vector u가 나옴

3.3 Heterogenous Signals

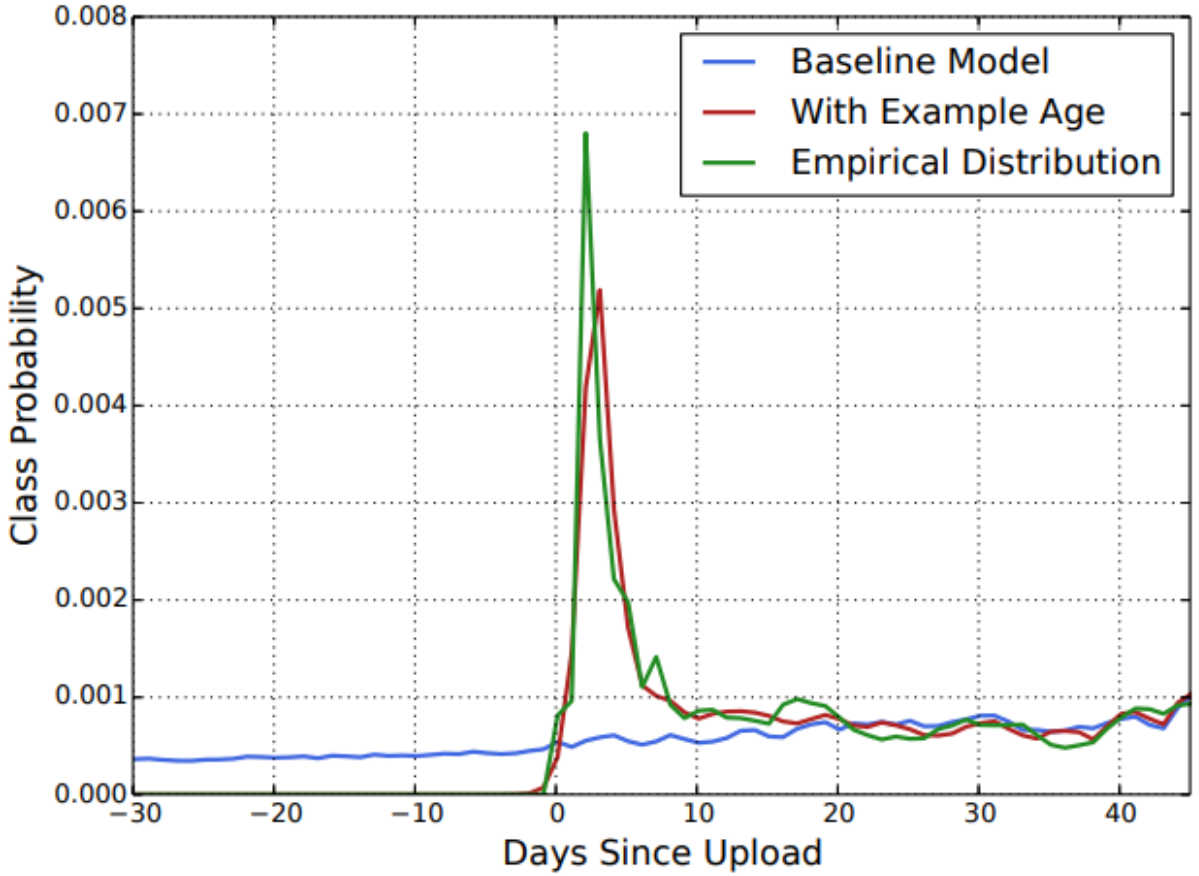


- MF를 DNN으로 사용하는 장점
  - 다양한 Continuous와 Categorical features를 모델에 쉽게 추가 가능
- Demographic 정보는 새로운 유저를 위해 아주 중요한 Feature
  - 방대한 User 데이터를 통해 Cold-Start 해결
- 검색 기록은 시청 기록과 유사하며, unigrams이나 bigrams을 활용하여 Embedding

3.3 Heterogenous Signals

“Example Age”

- User는 Fresh한 Contents를 선호
- 종종 과거 데이터에 많은 영향을 받으며 학습하여 과거의 아이템에 관련된 결과를 보여주는 bias가 생김
- 이를 시간적인 요소인 “Example Age” 즉, 영상의 나이를 고려해 보정



3.4 Label and Context Selection

Surrogate problem

- 개발한 서비스를 사용자로부터 피드백을 받아야 하지만, 대부분의 추천 시스템 개발 시 이런 과정을 거치는 것을 불가능
- RMSE 혹은 MAP와 같은 성능 지표를 활용하여 모델을 평가하는 과정

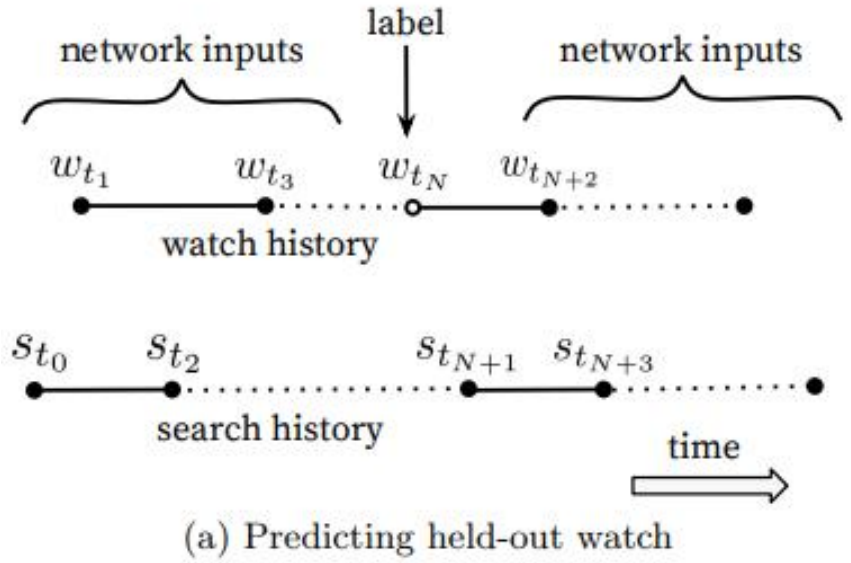
Training Examples

- 추천 영상 뿐만 아니라 모든 유튜브 시청 영상 포함(다른 경로의 유입도)
- 그렇지 않다면, 추천 결과에 bias가 생기고, Fresh한 contents를 추천하기 어려움
  - 추천에 의한 추천을 막고, Collaborative Filtering으로 다양한 추천 가능
- 유저 당 Fixed number of training examples
  - Heavy user의 영향을 줄임

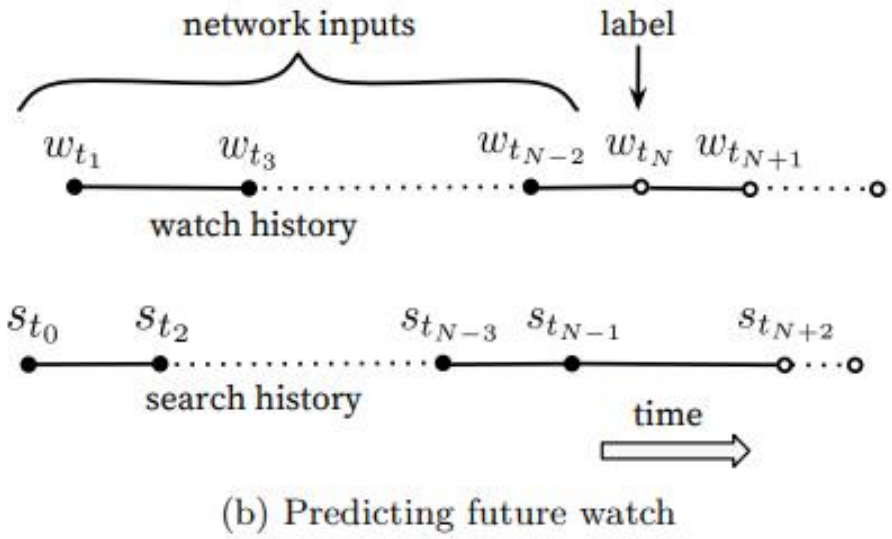
3.4 Label and Context Selection

Next Item Prediction

- 영상 시청 패턴은 매우 Asymmetric
  - Episodic series are usually watched sequentially
  - 노래는 Major → Minor
- Random한 Sampling 보다는 label 기준 이전의 데이터만을 Sampling하는 것이 효과적



(a) Predicting held-out watch



(b) Predicting future watch

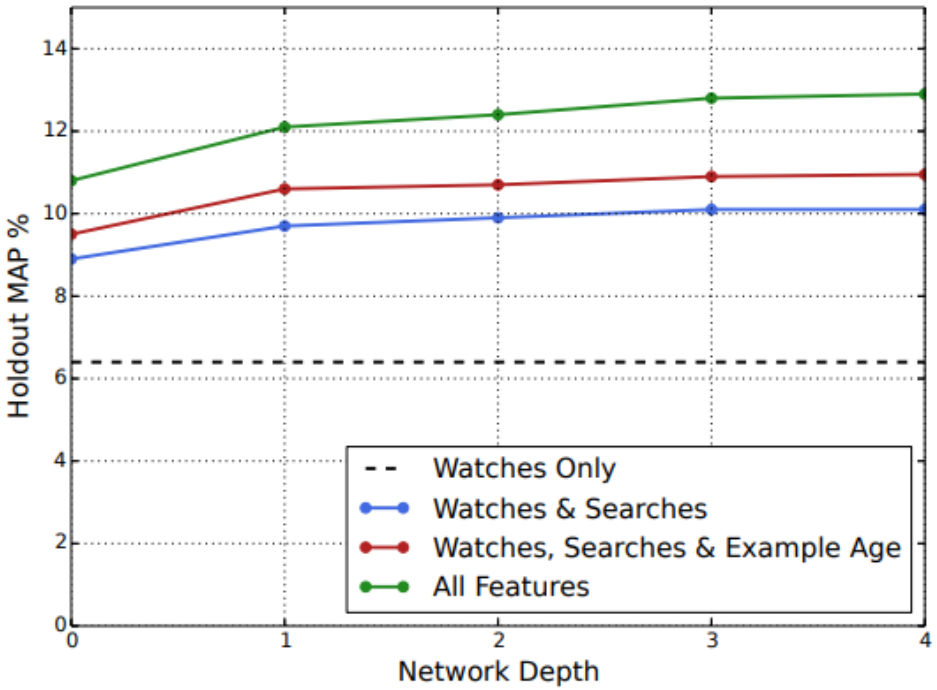
3.5 Experiments with Features and Depth

Experiment

- 1M의 video와 search token 사용
- Output dimension : 256
- Tower
  - Layer의 dimension이 점차 감소
  - Depth가 0인 것은 이전 시스템과 유사한 Linear Factorization

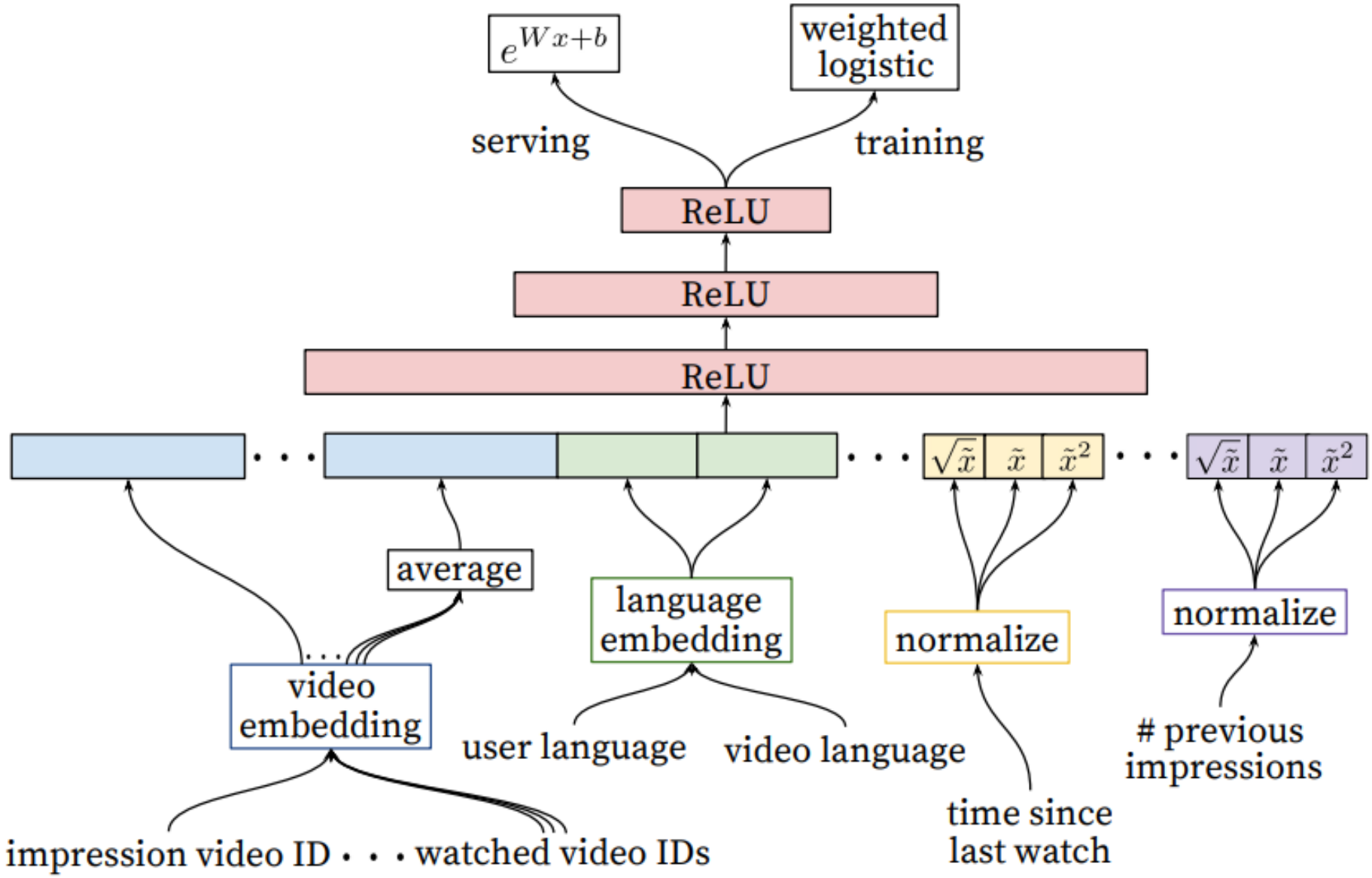
Precision은 Features와 Depth에 비례

- 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



4. Ranking

4. Ranking


e^{Wx+b}
weighted logistic
serving
training
ReLU
ReLU
ReLU
$\sqrt{\tilde{x}}$ $\tilde{x}$ $\tilde{x}^2$
$\sqrt{\tilde{x}}$ $\tilde{x}$ $\tilde{x}^2$
average
language embedding
normalize
normalize
video embedding
user language
video language
previous impressions
time since last watch
impression video ID
watched video IDs

4. Ranking

- Ranking Model의 주 역할
  - 각 사용자의 impression data를 사용해서 후보 아이템을 특성화하고 조정
    - Candidate Generation Model과 유사한 구조이나 각 아이템마다 score 할당
    - 후보 아이템에 score를 할당 후 정렬

- Ranking Model에서 사용하는 아이템은 Candidate Generation을 통해 Hundreds로 줄어 아이템에 대한 사용자의 방대한 Features 사용 가능

- 추천된 아이템들은 A/B test를 통해 조정
  - click-through rate를 예측하는 것이 아닌 watch time per impression 예측
  - watch time per impression이 사용자 참여도를 더 잘 표현

4.1 Feature Representation

데이터 분류

- 형태에 따라
  - Categorical features
    - Binary 데이터: User의 로그인 여부
    - Others: 사용자의 마지막 검색 기록
    - Univalent: 점수를 부여할 아이템 ID(1개)
    - Multivalent: 최근 본 N개의 아이템 ID(여러 개)
  - Continuous features

- 의미에 따라
  - Query: User나Context에 대한 features
  - Impression: 영상에 대한 features

4.1 Feature Engineering

Ranking Model

- 주로 수백개의 Feature들이 사용됨
- 딥러닝으로 Feature Engineering이 많이 필요하지 않지만 어느정도 전처리 필요
  - Ex) Raw Data를 직접 잘라서 사이즈를 줄임

가장 중요한 과제

- 사용자 행동에 대한 temporal sequence를 반영할지
- 어떻게 사용자 행동을 아이템 점수화와 연관시킬지

중요한 Signals(특히, User의 과거 action을 설명하는 continuous signal)

- 특정 채널에서 얼마나 많은 영상을 보았는지
- 최근에 이 주제에 대한 영상을 얼마나 보았는지
- 과거 추천 목록에 해당 영상이 얼마나 등장했는지

4.1 Embedding Categorical Features

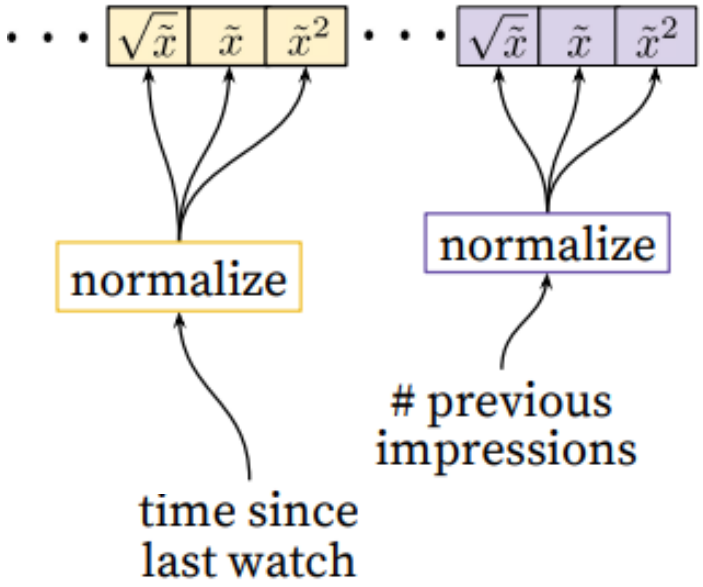
Embedding

- Categorical feature들을 신경망에서 쓰기 좋은 Dense Representation을 통해 Embedding
- Categorical feature와 Continuous feature는 유사한 중요도
- 매우 큰 cardinality space는 top N개만 Embedding
  - Video IDs, Search query terms 등
  - Click의 빈도수 기반으로 정렬하여 선정
  - Out-of-vocabulary인 item은 Zero Embedding
- Multivalent categorical feature embeddings은 average도 적용
  - 더 많은 Features 사용

4.1 Normalizing Continuous Features

Normalization

- Neural Networks는 Input의 scaling과 distribution에 민감
- 그렇기에 Continuous feature에 대해 적절한 정규화 필요



Proper Normalization

- x with distribution f 는 $\tilde{x} = \int_{-\infty}^{x} df$를 통해 $[0,1)$사이로 Scaling
  - Quantiles of the feature에서 linear interpolation로 근사

- $\tilde{x}^2, \sqrt{\tilde{x}}$를 input에 추가하여 super- and sub-linear 값들도 학습
  - 오프라인 실험 시 정확도 향상을 보여줌

4.2 Modeling Expected Watch Time

Expected watch time 예측

- Positive and Negative training sample 모두 사용
  - User의 Impression contents의 click 여부에 따라 Pos와 Neg를 나눔
- Weighted logistic regression을 사용해서 예측 가능
  - Cross-entropy loss를 사용
- Positive Sample에는 user의 동영상 시청 시간이 존재
  - 시청 시간에 따른 weight
- Negative Sample은 시청 시간이 없기에 unit weight를 부여

4.2 Modeling Expected Watch Time

The Odds는 logistic regression으로 학습

- N: # of training samples
- k: # of positive impressions
- T_i: time of the ith impression

Learned Odds: $\frac{\sum T_i}{N-k} = \frac{p(x)}{1-p(x)}$

Learned Odds

- The fraction of positive impressions이 매우 작다고 가정
- Learned Odds를 $E[T](1+P)$로 근사
  - P: Click probability, $\mathrm{E[T]}$: Watch time of the impression의 기댓값
  - P가 작다면 $E[T](1+P)$을 $E[T]$로 근사 가능
- e^x을 final activation function로 사용해 odds 생성

4.3 Experiments with Hidden Layers

Next-day holdout data

- Considering both positive (clicked) and negative (unclicked) impressions
- 각 impression 모두 scoring
  - Neg의 점수 > Pos의 점수
    - Positive impression을 통해 예측한 시청 시간은 mispredict
- Weighted per-user loss
  - 얼마나 시청시간을 mispredict 했는지

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.

Hidden layer의 width와 depth는 정확도와 비례

5. Conclusion

5. Conclusion

- **DNN을 Youtube 추천 시스템에 적용**
  - Candidate generation and Ranking Model
  - 이전에 사용했던 MF와 비교하여 많은 signals과 interaction을 활용 가능하며,더 좋은 성능을 보여줌
- **“Example age” 사용**
  - 과거 데이터에 편향된 추천을 제거하여 Freshness 증가
  - Time dependent한 행동을 모델에 적용시킬 수 있게 됨

5. Conclusion

- **Ranking Model & Logistic regression**
  - Feature engineering을 아직까지는 진행해야 하지만 이전의 linear and tree-based 모델보다 좋은 성능을 보임
  - 추천 시스템은 사용자의 과거 행동 feature가 있다면 더 좋은 성능을 발휘
  - CTR을 예측하는 metric보다 positive sample의 시청 시간을 학습하는 weighted logistic regression이 더 좋은 성능을 보임