SESSION-BASED RCOMMENDATIONS WITH RECURRENT NEURAL NETWORKS

추천 시스템 Boot Camp 4기 강석우

ABSTRACT

추천 시스템이라는 새로운 도메인에 RNN을 적용한다. 실제 추천 시스템은 장 시간의 유저 히스토리 대신 짧은 세션 데이터에 기반해서 추천하는 문제에 종종 직면하게 된다. 이러한 상황에서는 MF 접근법은 정확하지 않다. 그래서 Item-to-Item 추천에 의존해서 유사한 아이템을 추천하게 된다.

전 세션을 모델링 해서 더 정확한 추천을 제공할 수 있다. 따라서 세션 기반 추천을 위한 RNN 추천을 제시한다. 제시한 방법은 Task의 실용적인 측면까지 함께 고려했고, Classic한 RNN에서 Ranking Loss 함수의 수정도 있었다.

두 데이터 셋에 실험 결과로 개선점 들을 보인다.

1. INTRODUCTION

사이트 내 장기간의 유저 행동을 추적하는 것은 데이터의 신뢰성이 낮고, 개인 정보 관련 문제가 있다. 그래서 대부분의 세션 기반 추천 시스템은 거의 유저의 마지막 클릭이나 마지막 선택을 바탕으로 간단한 메소드를 사용한다.

추천 시스템에서 사용되는 굉장히 간단한 메소드들은 Factor, Neighborhood 모델이 있다. 세션 기반 모델에서 Factor(MF) 모델은 유저 프로파일의 부재로 적용이 어렵다. Neightborhood(Similarity, Co-Occurrences) 모델은 세션 기반의 추천에서 광범위하게 사용된다.

논문에서는 Sparse Sequential 데이터 모델링과 새로운 Ranking Loss 함수로 RNN이 세션 기반 추천에서 좋은 결과를 보일 거라고 주장한다.

1. INTRODUCTION

사용자가 사이트 방문 후 첫번째로 클릭한 아이템을 추천 RNN 모델의 첫 입력 값으로 한다. RNN 모델은 이 초기 입력으로 추천을 하고, 사용자가 계속해서 클릭할 때마다 이전 클릭에 따라 추천 결과가 달라진다.

추천 해야할 아이템 수가 굉장히 많고 길고 클릭 데이터 셋은 굉장히 크기 때문에, 학습 time과 scalability가 굉장히 중요하다.

대부분의 검색이나 추천 환경 처럼, top-item을 많이 고려해야 하기 때문에, RNN 모델의 학습에 Ranking Loss 함수를 사용한다

2. RELATIVE WORK

2.1 SESSION-BASED RECOMMENDATION

세션 내에 함께 클릭한 아이템으로 아이템 유사도 매트릭스를 미리 계산해서 사용했다. > 효과적인 접근법이지만, 과거 클릭들을 무시하는 한계가 있었다.

상태 전이 확률 모델인 MDP(Markov Decision Process)로 아이템 간 전이 확률을 사용한다. > 모든 유저가 아이템을 선택할 시퀀스를 고려해서 모델링하기 어렵다는 이슈가 있다.

General Factorization Framework(GFF)는 세션 내 발생한 이벤트 전체를 사용해서 아이템 latent와 아이템 자체를 모델링한다.
> 세션 내 순서를 고려하지 못했다.

2. RELATIVE WORK

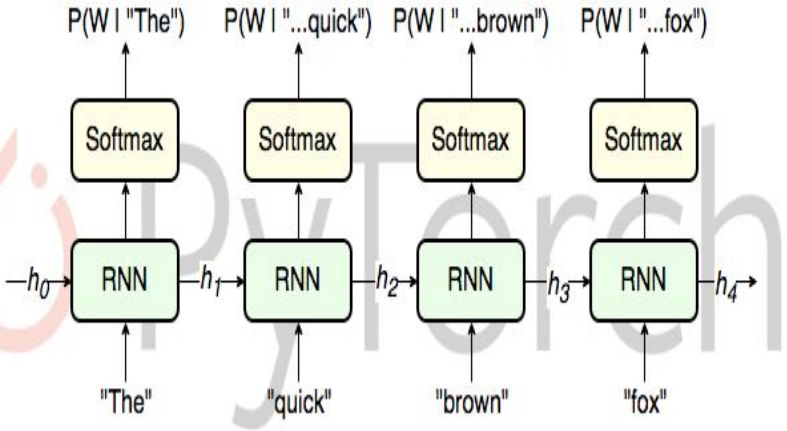
2.2 DEEP LEARNING IN RECOMMENDERS

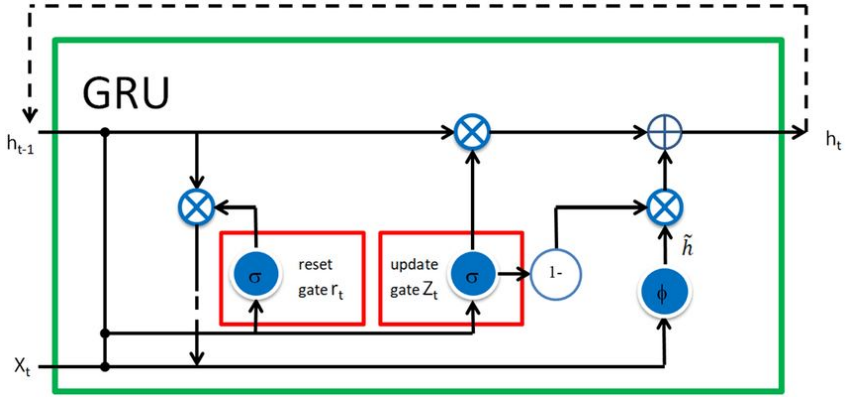
CF(Collaborative Filtering)를 위한 RBM(Restricted Boltzmann Machine)은 RBM을 사용해서 유저-아이템 상호작용을 모델링한다.

CF 모델들과 함께 사용해서 구조화 되지 않은 컨텐츠(음악, 이미지)에서 피처들을 뽑아내는데 Deep 모델을 사용했다.

> 이 접근법들은 유저-아이템 상호 작용 정보가 부족한 상황에서 특히 유용하게 사용 된다.

3. RECOMMENDATIONS WITH RNNs





RNN은 길이가 정해져 있지 않은 시퀀스 데이터를 모델링 하기 위해 모델을 구성하는 유닛 내부에 hidden state h를 가지고 학습하게 된다.

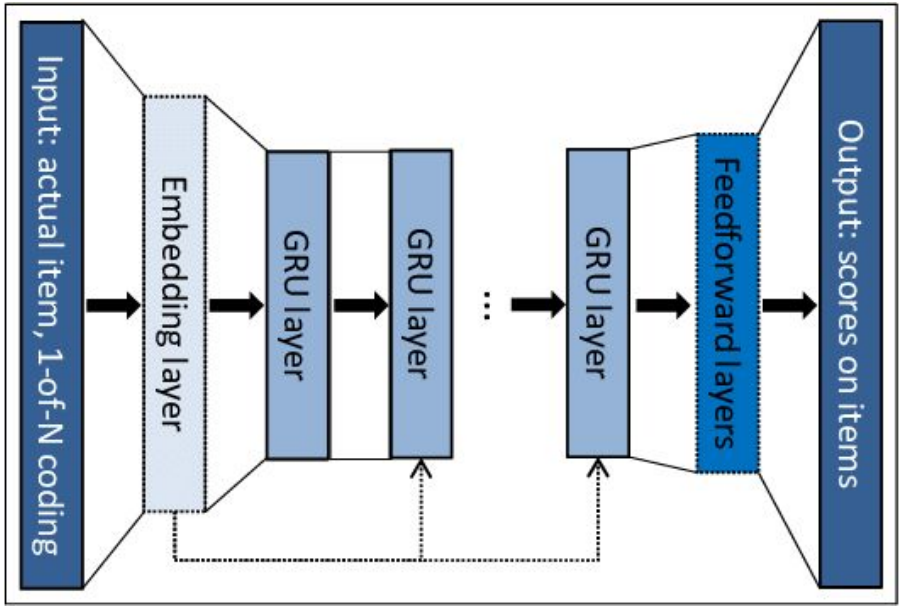
RNN의 Vanishing Gradient 문제를 해결하기 위해 hidden state h를 언제 얼마나 업데이트 해야하는 지도 함께 학습한다.

+본 논문에서는 GRU을 사용한다

.

3. RECOMMENDATIONS WITH RNNs

3.1 CUSTOMIZING THE GRU MODEL



네트워크의 입력은 세션의 현재 상태지만, 출력은 세션의 다음 아이템이다. 여기서 말하는 세션의 현재 상태는 실제 이벤트의 아이템이거나 세션 내 이벤트일 수 있다.

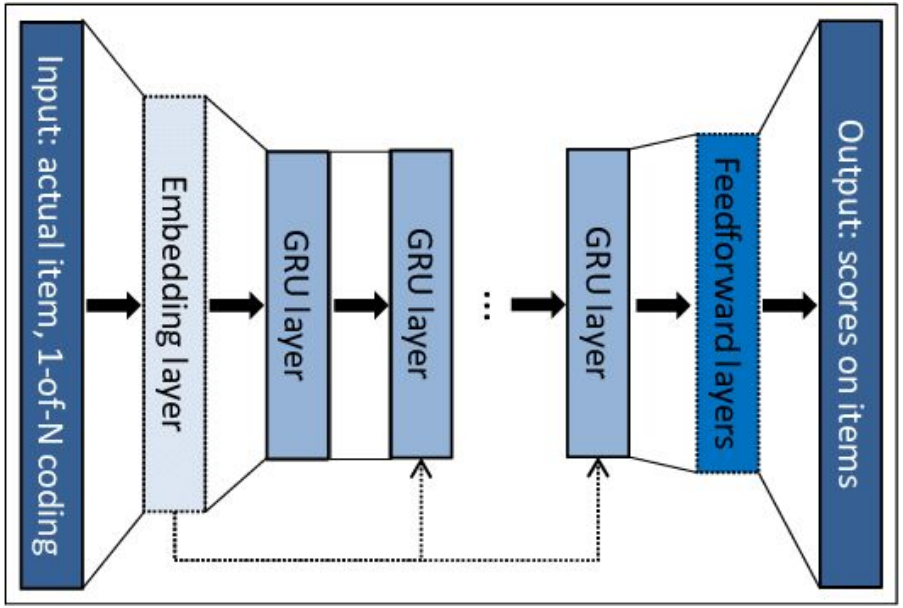
입력이 아이템일 때는 1-of-N 으로 인코딩해서 해당 아이템 만 1로, 나머지는 0으로 변환해서 사용 한다.

입력이 세션 내 이벤트 일 경우, 먼저 일어난 이벤트가 감소 되는 형태인 가중합을 사용한다. 이렇게 함으로 메모리 효과를 강화할 수 있다.

임베딩을 추가 해봤으나, 1-of-N 인코딩이 더 좋은 성능을 보였다.

3. RECOMMENDATIONS WITH RNNs

3.1 CUSTOMIZING THE GRU MODEL



본 논문에서는 RNN의 One-to-One 형태로 추천을 진행하려고 한다.

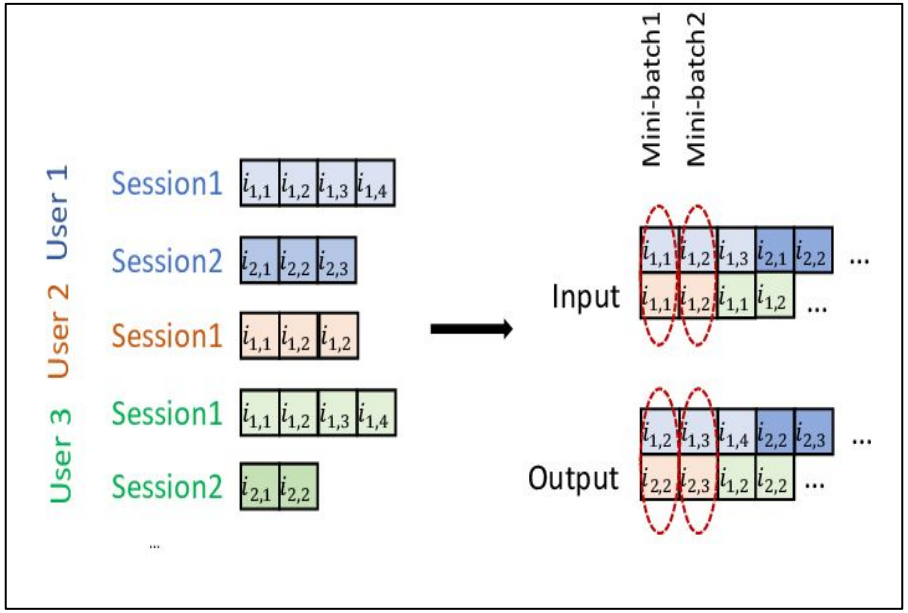
네트워크의 핵심은 GRU 레이어와 output가 마지막 레이어에 추가할 수 있는 Feedforward 레이어들이다.

Output은 아이템들의 예측된 선호도이다. 다음 세션에서의 모든 아이템의 likelihood 다.

GRU 레이어를 많이 사용 할수록 성능 증가를 보인다.

3. RECOMMENDATIONS WITH RNNs

3.1.1 SESSION-PARALLEL MINI-BATCHES



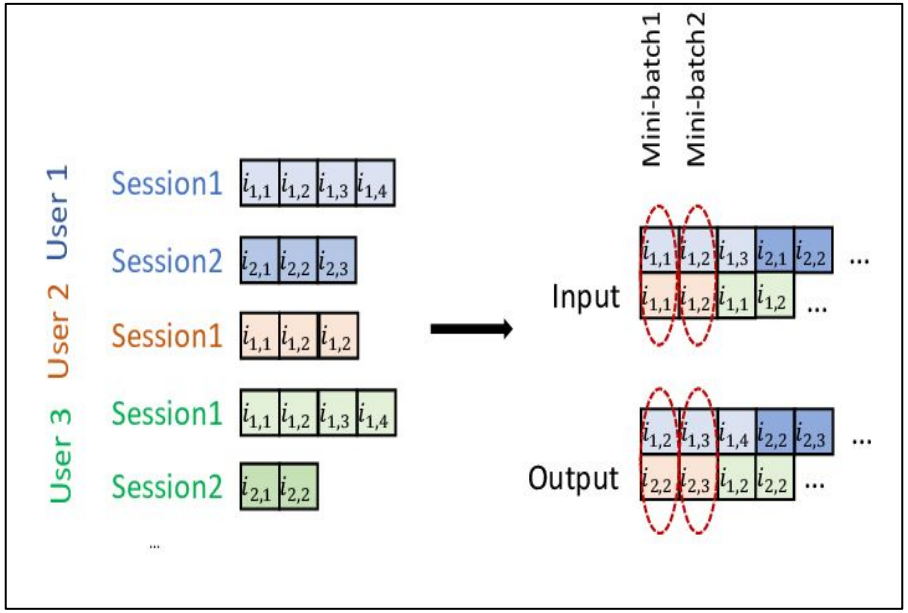
NLP에서 RNN은 in-sequence mini-batches를 사용한다. sliding window를 사용하는 컨셉은 여기서는 맞지 않는다. 우선 세션 들 간 길이 차이가 크다 (2 - ?00). 그리고 Goal는 시간이 지남에 따른 세션 변화를 모델링 하는 것이기 때문에 세션을 조각 내는 것은 의미가 없다.

그래서 session-parallel mini-batch를 사용한다. 세션의 위한 순서를 만든다.첫 X 세션의 첫 이벤트로 첫번째 미니 배치를 구성한다. 두번째 세션으로 두번째 미니 배치를 구성한다. 반복 한다.

세션들은 독립적이라고 가정해서, hidden state를 세션이 바뀔 때 초기화 해야한다.

3. RECOMMENDATIONS WITH RNNs

3.1.2 SAMPLING ON THE OUTPUT



전체 아이템 들의 점수를 각각 계산하는 것은 알고리즘의 scale을 만든다. 그래서 output 중 일부만 점수를 계산 해야 한다. 이러면 일부 weight만 업데이트 될 수 있어서, negative example 도 점수를 계산해서 weight를 수정해야 한다.

missing event에 대한 자연적 해석은 “유저가 아이템을 모른다.” 지만 아이템을 싫어하는 것이다. 그래서 아이템의 유명도로 샘플링 해야 하기 때문에, 다른 mini-batch에서 negative example 을 가져온다. 계산 시간을 위해 이 과정은 건너 뛸 수도 있다.
(popularity-based sampling)

3. RECOMMENDATIONS WITH RNNs

3.1.3 Ranking LOSS

추천 시스템의 핵심은 관련도 기반의 아이템 랭킹이다. 태스크는 classification으로, Learning-To-Rank 접근법을 할 수 있다. Pointwise Ranking은 아이템 각각의 랭킹을 측정하는 것이다. Pairwise Ranking은 Positive와 Negative 아이템을 비교하는 것. Listwise는 모든 아이템의 랭크와 완벽한 리스트 랭크를 비교하는 것이다.

Pointwise Ranking은 이 네트워크에 적절하지 않고 Pairwise Ranking이 좋다고 한다.

BPR : Bayesian Personalized Ranking

$$L_s = -\frac{1}{N_s} \cdot \sum_{j=1}^{N_s} log(\sigma(\hat{r}_{s,i} - \hat{r}_{s,j}))$$

TOP1

$$L_s = \frac{1}{N_s} \cdot \sum_{j=1}^{N_s} \sigma(\hat{r}_{s,j} - \hat{r}_{s,i}) + \sigma(\hat{r}_{s,j}^2)$$

4. EXPERIMENTS

RecSys Challenge 2015 데이터셋

Youtube-like OTT 비디오 서비스 플랫폼 데이터셋

다음 이벤트의 아이템의 랭크를 체크로 evaluation을 한다. GRU의 hidden state는 세션이 끝나면 0으로 초기화 한다.

recall@20과 MRR@20(Mean Reciprocal Rank)을 사용했다.

Baselines 모델로 POP, S-POP, Item-KNN, BPR-MF을 비교했다.

4. EXPERIMENTS

Table 1: Recall@20 and MRR@20 using the baseline methods

Baseline	RSC15		VIDEO	
	Recall@20	MRR@20	Recall@20	MRR@20
POP	0.0050	0.0012	0.0499	0.0117
S-POP	0.2672	0.1775	0.1301	0.0863
Item-KNN	0.5065	0.2048	0.5508	0.3381
BPR-MF	0.2574	0.0618	0.0692	0.0374

Table 2: Best parametrizations for datasets/loss functions

Dataset	Loss	Mini-batch	Dropout	Learning rate	Momentum
RSC15	TOP1	50	0.5	0.01	0
RSC15	BPR	50	0.2	0.05	0.2
RSC15	Cross-entropy	500	0	0.01	0
VIDEO	TOP1	50	0.4	0.05	0
VIDEO	BPR	50	0.3	0.1	0
VIDEO	Cross-entropy	200	0.1	0.05	0.3

Hyp-Opt는 100번의 실험으로 진행했다.

Hidden Unit을 100개 사용했다. 최적 파라미터들은 서로 다른 hidden layer 수를 적용했다.

Weight 초기화는 [-x, x] 로 matrix의 row와 column 에 따라 다르다.

rmsprop과 adagrad Opimizer를 사용했다.

classic RNN과 LSTM도 함께 사용했으나 GRU가 더 좋은 성능을 냈다.

4. EXPERIMENTS

Table 3: Recall@20 and MRR@20 for different types of a single layer of GRU, compared to the best baseline (item-KNN). Best results per dataset are highlighted.

Loss / #Units	RSC15		VIDEO	
	Recall@20	MRR@20	Recall@20	MRR@20
TOP1 100	0.5853 (+15.55%)	0.2305 (+12.58%)	0.6141 (+11.50%)	0.3511 (+3.84%)
BPR 100	0.6069 (+19.82%)	0.2407 (+17.54%)	0.5999 (+8.92%)	0.3260 (-3.56%)
Cross-entropy 100	0.6074 (+19.91%)	0.2430 (+18.65%)	0.6372 (+15.69%)	0.3720 (+10.04%)
TOP1 1000	0.6206 (+22.53%)	**0.2693 (+31.49%)**	**0.6624 (+20.27%)**	**0.3891 (+15.08%)**
BPR 1000	**0.6322 (+24.82%)**	0.2467 (+20.47%)	0.6311 (+14.58%)	0.3136 (-7.23%)
Cross-entropy 1000	0.5777 (+14.06%)	0.2153 (+5.16%)	–	–