다양한 임베딩을 활용한 협업 필터링

김민식
국민대학교 AI빅데이터경영통계학과
(gby-1349@kookmin.ac.kr)

김건우
국민대학교 AI빅데이터경영통계학과
(author2@hankuk.ac.kr)

최근 다양한 추천시스템 기술이 등장했지만, CF 기반 모델이 여전히 좋은 성능을 보인다. 이런 CF 기반 모델의 핵심은 "사용자와 아이템 사이 상호작용을 어떻게 표현하고 모델링 하는가"이다. 그래서 이전 선행 연구에서 사용자와 아이템 사이의 다양한 임베딩 방법(ex: Matrix, Metric)과 종류(ex: Dual, Historical, ConvNCF-SVD++ 등) 활용해서 성능을 높여왔다. 임베딩에 대한 다양한 시도로 추천시스템의 성능 향상 및 발전하였음에도 불구하고, 다양한 연구에서 등장하는 임베딩 방법을 포괄적으로 활용하여 최적의 임베딩 조합을 찾는 실험, 연구는 아직 없는 상황이다.

그래서 본 논문을 통해 선행연구에서 등장한 다양한 종류의 임베딩을 실험할 일반적인 아키텍처를 제시한 후 각 임베딩별 개별적인 성능을 비교하여 추천시스템 성능의 가장 큰 영향을 미치는 요소들을 확인해본다. 그리고 최적 조합을 찾기 위해서 회귀에서 변수선택법으로 사용되는'후진 제거법'을 활용한다. 추가적으로 기존 연구에서는 주로 임베딩 벡터 간의 내적 혹은 결합(Concatenation)을 한 후 MLP를 통해 representation을 추출하였는데, 본 연구에서는 아이템과 사용자 사이의 더 많은 상관관계를 관찰하기 위해 외적을 적용한 한 후 CNN을 활용해 모델링하였다.

2개의 데이터를 기반으로 한 다양한 실험을 통해서 각 임베딩별의 성능과 최적의 조합을 찾고, 최적의 조합에 대한 성능이 최근 딥러닝 기반 추천시스템에 대비 성능이 뛰어나는 것을 입증한다. 이를 통해, 사용자와 아이템 사이의 더 많은 암시적인 정보를 활용하여 최적의 모형을 제시할 수 있다.

주제어: 추천시스템, 딥러닝, 협업 필터링, 임베딩, 서베이

논문접수일 : 2022년 6월
투고유형 : 국문일반

1. 개요

추천시스템은 사용자와 상품의 정보를 수집하여 고객을 대상으로 더 나은 개인화된 서비스 추천을 진행한다.

특히 최근 인터넷의 급속한 발전으로 인해 많은 정보가 생성 및 활용할 수 있게 되었고, 이들 중에 정확하고 필요한 정보를 사용자에게 제공하는 것은 매우 중요한 일이 되었다. 또한 인터넷에서 사용되는 전자 상거래, SNS, OTT 서비스 등 고부가 가치적인 정보를 인공지능 학습에 사용한다면 큰 수익을 기대할 수 있다.

실제로 추천시스템을 통해 아마존에서는 매출의 약 30%가 증가하였고, 넷플릭스는 연 10억 달러 이상의 수익이 추천시스템에 의해 만들어지고 있다 [1]. 이처럼 추천시스템의 지난 수십 년간 많은 연구와 발전을 이룩하고 있다.

협업 필터링(Collaborative Filtering, CF)은 가장 널리 사용되며, 다양한 추천시스템 기술이 등장하고 있는 현재에도 좋은 성능을 보이는 방법이다 [2].

CF 기반 모델의 핵심은 "사용자와 아이템 사이 상호작용을 어떻게 표현하고 모델링 하는가?"이다.

그래서 기존에 추천시스템 많은 연구에서는 기존 연구에서는 사용자와 아이템 사이의 다양한 상호관계를 추출하기 위한 여러 임베딩 방법(ex: Matrix, Metric)과 종류(ex: Dual, Historical 등), 모델링 방식(MLP 등)을 연구하여 방식을 발전시켰고, 추천시스템의 성과를 높였다.

임베딩에 대한 다양한 시도로 추천시스템의 성능 향상 및 발전하였음에도 불구하고, 다양한 연구에서 등장하는 임베딩 방법을 포괄적으로 활용하여 최적의 임베딩 조합을 찾는 실험, 연구는 아직 없는 상황이다.

이러한 문제를 해결하기 위해, 본 논문에서 암시적 피드백(Implicit Feedback)에서 얻을 수 있는 다양한 임베딩에 대한 최적의 조합을 찾아 제안하고 한다. 이때 기존 Matrix Factorization(MF) 방식의 한계를 보완하기 위해 Metric Factorization 임베딩도 추가한다. 그리고 임베딩 벡터 간 Concatenation 하여 MLP 모델링하는 것 이외에도 아이템과 사용자 사이의 더 많은 유의미한 정보를 포착하기 위해 외적과 CNN을 활용해 Representation을 얻는 모델을 제안한다.

본 논문에 주요 Contribution은 다음과 같다.

- 사용자와 아이템 사이의 다양한 representation을 만드는 임베딩에 대한 최적의 조합을 찾는 survey 성격의 최초 논문이다.
- 임베딩 벡터 간 내적이나 Concatenation 후 MLP와 GMF 등으로 모델링한 기존 방식과 달리, 아이템과 사용자 간의 더 많은 상관관계를 학습하기 위해 Latent Vector에 대해, Concatenation과 외적을 사용해 MLP과 CNN으로 모델링을 했다.
- Yelp, Book 데이터를 활용하여 추천시스템에 대한 개별 임베딩의 성능을 비교할 수 있으며, 최적의 조합에 대해서는 최신 추천시스템 모델 대비 성능이 우수하다는 것을 실험을 통해 입증한다.

2. 관련 연구

과거 CF 방식에는, Matrix Factorization(MF)를 활용해 아이템과 사용자 임베딩을 구한 뒤 내적을 통해 상호작용을 모델링한 방식이 있었다 [3]. 하지만 MF의 경우는 상호작용을 포착할 때 고정된 크기의 내적을 이용한다는 점 때문에, 복잡한 사용자와 아이템 사이의 상호작용을 포착하기에 어려움이 있다.

그래서 딥러닝의 발전함에 따라서, 딥러닝 모델을 추천시스템에 적용하기 시작하였다. 아이템과 사용자에 대한 비선형적(non-linear) 상호작용 학습하기 위해 MLP로 모델링한 MLP [4], 기존 CF 모델과 MLP 모델을 혼합해서 만든 Neural Matrix Factorization(NeuMF) [4]. 모델링하기 위해 임베딩 벡터간의 연산으로 사용한 Concatenation 방식 혹은 내적은 아이템과 사용자 간의 제한적인 상호작용 정보만을 가진다는 한계를 가져 더 많은 상관관계를 포착하기 위해 외적 연산을 사용 후 CNN으로 모델링한 ConvNCF 방식이 등장하였다 [5].

이후 사용자와 아이템 간의 임베딩하기 위한 새로운 요소를 찾기 위한 다양한 방식이 등장하였다. ConvNCF-SVD++의 경우, 사용자에 대한 임베딩을 사용자 ID뿐만 아니라 Historical interaction 정보를 결합한 방식을 제안하였다 [1]. DNCF의 경우, 사용자와 아이템 각각 ID와 Historical interaction에 대한 임베딩에 대해 연산을 통해 결합한 새로운 방식의 Dual Embedding을 제안하여 NCF의 적용하였다 [6].

반면 DDFL의 경우, 기존 Matrix Factorization 방식의 한계점 triangular inequality를 만족하지 못하는 문제를 해결하기 위해 Metric Factorization을 적용하여 아이템과 사용자에 대해 Distance Embedding을 통해 저차원의 거리로 맵핑 후 상호

작용을 유클리디안 거리와 MLP 구조를 활용해 포착하고자 하였다 [7].

3. 제안 방법론

이 장에서 먼저 사용할 입력값과 다양한 임베딩에 대한 개념과 내용을 소개한 이후 아키텍처를 통해 전반적인 프레임워크에 관해서 설명하고자 한다.

3.1 임베딩

(0) 입력값

- **ID:** 암시적 피드백(Implicit Feedback)을 가진 아이템과 사용자의 고유한 ID 정보로, 사용자와 아이템을 각각 x_u와 x_i로 정의한다.
- **Historical Interaction**: 사용자와 아이템 과거 암시적 피드백(Implicit Feedback) 정보로, 사용자와 아이템을 각각 h_u와 h_i로 정의한다. (예: h_u는 과거 사용자 u의 피드백 받은 모든 아이템)

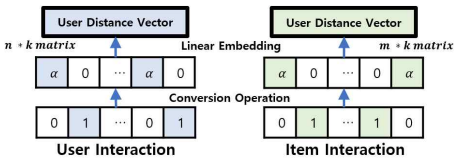
(1) Linear Embedding

아이템과 사용자의 Historical Interaction 정보를 거리로 변환하기 위해 수식 (1)번 연산을 적용한다. 각각 d_u^U, d_i^I로 정의하면 다음과 같다 [7].

$$d_{ui} = \alpha(1 - h_{ui}) \quad (1)$$

d_u^U, d_i^I을 Linear Embedding Layer를 거친 이후 수식 (2)와 같이 각각 k 차원 Dense representation을 얻는다. [5] Figure 1의 Linear Embedding이다.

$$p_u^d = P^T d_u^U \ (P: n \times k \ \ matrix) \quad (2)$$
$$q_i^d = Q^T d_i^I \ (Q: m \times k \ \ matrix)$$



<Fig 1> The architecture of linear Embedding

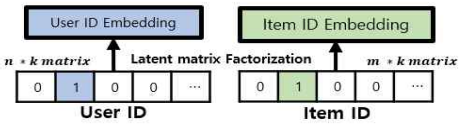
(2) MF Embedding

아이템과 사용자의 고차원이면서 희소한 ID와 Historical Interaction 정보를 latent Factor 행렬을 곱하여 저차원의 Dense Latent Vector를 얻는다. 아래 4개가 MF에서 생성할 Embedding 종류이다.

- **ID Embedding**: 가장 기본적인 형태의 임베딩으로 사용자와 아이템의 ID를 임베딩 한 결과이다. 수식 (3)과 같이 정의한다.

$$p_u^{id} = P_{id}^T x_u, \quad q_i^{id} = Q_{id}^T x_i \quad (3)$$

사용자와 아이템 각각의 Representation을 의미하며, Figure 2와 같다

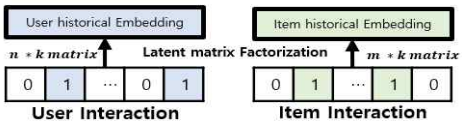


<Fig 2> The architecture of ID Embedding

- **Historical Embedding**: 사용자와 아이템의 과거 상호작용했던 정보를 임베딩한 결과로 수식 (4)로 정의한다.

$$p_u^h = P_h^T h_y, \quad q_i^h = Q_h^T h_i \quad (4)$$

사용자와 아이템에 대한 상호작용한 과거 정보에 대한 Representation을 나타내며, Figure 3과 같다

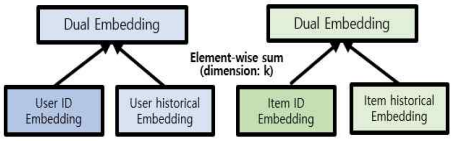


<Fig 3> The architecture of Historical Embedding

- **Dual Embedding**: 앞서 사용자와 아이템 ID와 Historical Interaction 정보를 통해서 구했던 임베딩을 결합한 새로운 임베딩이다.

$$p_u^{dual} = g(p_u^{id} \circ p_u^h),\ q_u^{dual} = g(q_u^{id} \circ q_u^h) \quad (5)$$

수식은 (5)번과 같으며, ID Embedding과 Historical Embedding 모두 k차원으로 결합할 수 있다. $g(\circ)$ 두 임베딩을 결합하는 함수이며, 선행연구 실험에서 준수한 성과를 보인 Element-wise로 두 임베딩을 결합한다 [6]. ID와 과거 정보를 포함한 Representation이며, Figure 4처럼 표현한다.

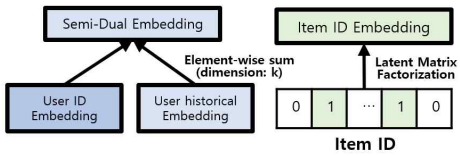


<Fig 4> The architecture of Dual Embedding

- **Semi-Dual Embedding**: 아이템의 경우에는 ID Embedding으로 구성하고, 사용자의 경우만 사용자의 ID와 Historical Embedding을 결합해 구성한다. 수식으로 표현하면 (6)과 같다 [1].

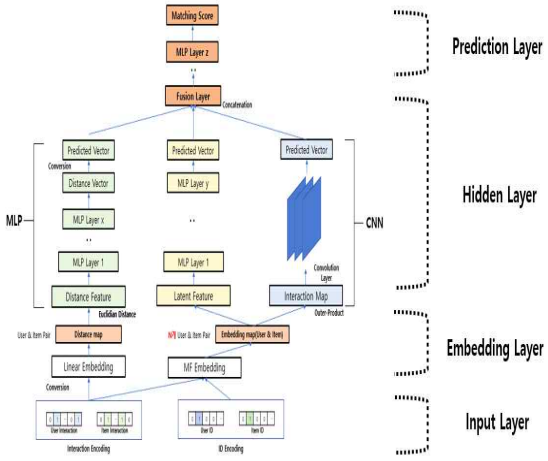
$$p_u^{semi} = g(p_u^{id} \circ p_i^h) \quad (6)$$

사용자는 자신의 고유한 ID 정보 이외의 과거 구매했던 아이템에 대한 정보를 포함한 Representation을 가지게 되며 Figure 5와 같이 표현한다.



<Fig 5> The architecture of Semi-Dual Embedding

3.2 아키텍처 설명



<Fig 5> The Overall architecture

Figure 6이 최적의 임베딩을 찾기 위해 제안할 모델의 아키텍처 구조이다. 총 4개의 층을 거치며 층마다 설명은 다음과 같다.

- **Input Layer**: 사용자(u)와 아이템(i)에 대한 ID와 Historical interaction 받는다. 이후 ID는 원-핫 인코딩 Historical interaction은 멀티-핫 인코딩을 진행하여 고차원의 희소 벡터를 얻는다. (binary, 0과 1로 구성)
- **Embedding Layer**: Input Layer에서 얻은 고차

원의 희소한 정보를 저차원의 Dense한 임베딩 벡터로 변환한다. 아이템은 각각 서로 다른 m × k 행렬을 사용자는 서로 다른 n × k 행렬을 곱해 k의 임배딩 차원으로 만든다. 앞선 정의된 것처럼 총 2개의 임베딩으로 구분되는데 Linear Embedding에서는 distance Representation을 갖는 1개의 임베딩이 MF Embedding은 Latent Representation을 갖는 4개의 임베딩이 나온다.

- **Hidden Layer**: Embedding Layer에서 생성된 임베딩을 가지고 사용자와 아이템 사이의 Representation을 학습하는 단계이다. 각 임베딩마다 서로 다른 Hidden Layer를 거치며 다음과 같다.

[Metric Function Learning]:

Linear Embedding에서 얻은 사용자와 아이템 사이 distance Representation(Map)에 대해 유클리디안 거리를 구한다 [7].

$$d_0 = \left((d_{u1} - d_{1i})^2, \cdots, \cdots, (d_{uk} - d_{ki})^2\right) \quad (7)$$

수식은 위와 같으며, 이후 MLP 거쳐 distance Vector를 구한 뒤 변환 연산을 역으로 진행한다. b_n , $W_{n.}$ 은 각 층별 편향, 가중치이다

$$\begin{aligned} d_1 &= ReLU(W_1^T d_0 + b_1) \quad (8)\\ &\vdots \\ d_n &= ReLU(W_n^T f_{n-1} + b_n) \\ d_{final} &= 1 - \frac{d_n}{a}\,(a: distace\ \ factor) \end{aligned}$$

본 논문에서는 사용자와 아이템의 Historical Interaction에 대한 1개의 distance Feature를 통해 Predicted vector를 만든다.

[Matching Function Learning]

MF Embedding에서 얻은 사용자와 아이템 사이의 Latent Vector에 대해서 각각 MLP와 CNN을 적용한다.

(9)번 수식과 같이 먼저 p_u, q_i로 정의된 MF Embedding을 거쳐 나온 아이템과 사용자 임베딩을 Concatenation을 진행한다. 이후 MLP를 적용한다. b_n , $W_{n.}$ 은 각 층별 편향, 가중치이다

$$\begin{aligned} m_0^{MLP} &= p_u \oplus q_i \quad (9)\\ m_1{}^{MLP} &= ReLU(W_1^T m_0 + b_1) \\ &\vdots \\ m_n^{MLP} &= ReLU(W_n^T m_1^{MLP} + b_n) \end{aligned}$$

CNN을 적용하기 위해서는 수식에서 $\otimes$로 표현한 두 임베딩 간 외적을 진행한다. 외적을 통해서 얻은 $k \times k$ 행렬을 CNN를 통해 Representation을 추출한다. 최종 m_{final}^{CNN}의 경우는 앞선 Hidden Layer와 크기를 맞추기 위해, 최종 결과는 "$1 \times 1 \times$ 채널"로 설정하며 나오며 마지막 채널이 벡터 크기이다.

$$m_0^{CNN} = p_u \otimes q_i \quad (10)$$

본 논문에서는 MF Embedding를 각각의 사용자와 아이템 간 ID와 Historical interaction에 대한 총 4개(ID, Historical, Dual, Semi-Dual)를 활용하였고, 최종 8개의 prediction vector가 나온다.

- **Prediction Layer**: Hidden Layer에서 얻은 Prediction Vector를 가지고 사용자(u)와 아이템(i)의 y($\acute{y_{ui}}$)값을 예측한다. Embedding을 거쳐서 얻은 Prediction Vector($v_1, v_2 \cdots v_n$) concatenation을 적용한 이후 MLP를 거쳐 예측 스코어를 낸다.

$$\begin{aligned} f_0 &= d_{final} \oplus m^{MLP} \oplus m^{CNN} \quad (11)\\ f_1 &= \alpha_1(W_1^T f_0 + b_1) \\ &\vdots \\ f_n &= \alpha_n(W_n^T f_{n-1} + b_n) \\ \acute{y_{ui}} &= sigmoid(W_{out}^T f_n + b_{out}) \end{aligned}$$

수식은 (11)번과 같으며, d_{final}은 Hidden layer에서 Linear Embedding을 MLP에 넣어서 나온

Predicted Vector를, m^{MLP}과 m^{CNN}는 각각 MF Embedding을 MLP와 CNN을 거쳐서 나온 Predicted Vector이다. 총 9개가 나온다.

4. 실험 및 결과

본 장에서는 사용자와 아이템 사이의 다양한 Representation을 표현하는 여러 종류의 임베딩에 대한 최적 조합을 찾고자 한다. 그리고 최적의 임베딩 조합의 성능을 종합적으로 평가하기 위해 다음과 같은 Research Question을 설정 후 실험을 통해 답을 찾는다.

RQ1: 사용자와 아이템 사이의 최적의 임베딩 조합은 무엇인가?

RQ2: 제시된 최적의 임베딩 조합이 최신 추천 모델의 성능을 능가하는가?

4.1 데이터 세트

본 연구에서는 총 2개의 데이터를 가지고 실험을 진행한다. 데이터의 통계자료는 Table 1과 같다.

- **Yelp**: Yelp Challenge data로, Yelp에서의 연구 목적으로 제공한 데이터이다. 해당 데이터의 경우, 이전 연구에서 전처리가 완료된 데이터[1]를 그대로 사용한다.
- **Book**: 사용자가 도서를 구매한 이후, 남긴 평가 정보를 수집한 데이터이다. 본 연구에서는 아이템과 상호작용이 5번 이상 등장한 사용자만 학습할 수 있도록 전처리를 진행한다.

<Table 1> Statistics of the Dataset

Dataset	Users	Items	Interactions	Density
Book	26,607	66,001	242,767	98.61%
Yelp	25,815	25,677	730,791	99.88%

4.2 평가 방식

추천시스템의 성능을 평가하기 위해 가장 널리 사용되는 “leave one out” 방식을 사용하여, 각 조합의 성능을 측정한다. 사용자별로 가장 최근에 상호작용한 아이템은 테스트로, 나머지는 훈련용 데이터로 구성한다. 평가 시 모든 항목에 대해 순위를 매기는 것이 어려워 이전 연구처럼 [5], 사용자마다 상호작용하지 않은 아이템 중 99개를 랜덤하게 샘플링한 후, 100개 항목의 순위를 예측한다.

평가 지표로는 Hit Ratio@k(HR) 와 Normalized Discounted Cumulative Gain@k(NDCG) 2가지를 사용하며, 두 지표 모두 상위 10개(k = 10) 항목만을 측정하여 평가한다. HR@10은 “테스트 아이템이 추천 상위 10개 항목에 있는지” 여부를 보여주는 직관적인 지표이고, NDCG@10은 “테스트 아이템이 상위 순위에 있을수록 더 높은 점수를 부여”하는 지표이다. 사용자별로 두 평가 지표를 계산한 이후, 평균을 취해 성능을 계산한다.

4.3 비교 모델

실험을 통해 얻은 최적의 임베딩 조합의 성능이 우수한지 알아보기 위해, 아래 모델과 성능 비교를 한다.

- **MLP** [4]: 사용자와 아이템 ID에 대한 임베딩을 Concatenation 한 후, 사용자와 아이템 사이의 non-linear interaction을 학습하기 위해 MLP를

1) https://github.com/duxy-me/ConvNCF/tree/master/Data

사용한 NCF 기법

- **NeuNF** [4]: 사용자와 아이템 간 상호작용 관계를 학습하기 위해 GMF와 MLP 방식을 결합한 모델
- **ConvNCF** [5]: 사용자와 아이템 사이의 더 많은 상관관계(Correlation) 활용하기 위해, 임베딩 간 외적(Outer product)을 적용하여 2차원 형식의 Feature Map을 만들고 고차원 정보를 학습하기 위해 CNN 활용한 아키텍처
- **DeepCF** [8]: Matching Function 기반 CF와 representation 기반 CF를 결합한 아키텍처
- **DNCF** [6]: 사용자와 아이템의 ID와 Historical Interaction 정보를 모두 임베딩한 후 결합한 Dual Embedding을 Latent Vector로 사용한 모델
- **DDFL** [7]: 기존 Matching Function 기반 CF의 단점을 보완하기 위해, 사용자와 아이템을 저차원 좌표계의 점으로 간주해 유클리디안 거리를 사용하여 두 사이의 관계를 모델링하는 Metric Function Learning을 제안. DDFL은 기존 Matching Function 기반 CF와 Metric Function 기반 CF를 결합한 아키텍처

4.4 파라미터 설정

Pytorch[2]를 기반으로 임베딩과 모델 구조를 구현하였다. 실험 결과의 정당성을 확보하기 위해 각 실험 별로 하이퍼 파라미터 (Hyper Parameter)를 똑같이 설정해 성능을 평가했다. 하이퍼 파라미터를 설정한 기준은 모델에 전체 임베딩을 넣은 실험에서 최적의 성능을 보인 파라미터로 결정하였다.

하이퍼 파라미터 설정은 아래 Table 2과 같다. 모델의 파라미터는 가우스 분포(평균 0, 표준편차 0.01)에 따라 랜덤하게 초기화한다. Hidden Layer의 CNN은 이전 연구처럼 [6] 6개의 Layer로 구성되며 각 Layer 채널은 32이고, stride가 2고, 크기가 2 * 2 Convolution을 사용한다. MLP의 경우 3개의 Fully Connected Layer로 설정한다. Predicted Layer에서는 각 Hidden Layer에서의 얻은 다수의 Predicted Vector를 Concatenation을 진행하고, predicted Layer의 MLP 역시 3개의 Fully Connected Layer로 구성한다.

<Table 2> Parameter Setting

Validation set	2 User-Item Interaction (1 positive sample + 1 negative sample)		
Test set	100 User-Item Interaction (1 positive sample + 99 negative sample)		
loss function	BPR Loss		
epoch	500	batch size	512
Optimizer	Adagrad	learning rate	0.001
distance factor(α)	0.9	regularization coefficient	1e-5
Predicted Vector size	32	embedding size	64

4.5 실험 계획(진행 방향)

(1) 개별적인 임베딩 성능 확인 (RQ1)

임베딩마다 개별적으로 성능을 테스트한 이후 결과를 비교한다. 개별적인 성능 확인을 위해 각 임베딩을 모듈화하여 불러온 이후 해당 임베딩만 학습을 진행하도록 실험을 진행한다. 실험 결과는 아래 Table 3과 같다.

개별적인 성능 결과를 통해서 (1) 각 임베딩마다 상대적인 중요도를 파악하며 (2) 최적의 조합(전진 선택법의 순서)을 찾기 위한 과정으로 활용한다

2) https://github.com/pytorch/pytorch

<Table 3> Individual performance for each embedding

	Book		Yelp	
	HR@10	NDCG@10	HR@10	NDCG@10
Distance	-	-	-	-
ID + MLP	-	-	-	-
ID + CNN	-	-	-	-
History + MLP	-	-	-	-
History + CNN	-	-	-	-
Dual + MLP	-	-	-	-
Dual + CNN	-	-	-	-
Semi Dual + MLP	-	-	-	-
Semi Dual + CNN	-	-	-	-

(2) 최적의 임베딩 조합 (RQ1)

앞선 실험에서 나온 개별 임베딩 성능을 고려해 최적의 임베딩 조합을 찾는 실험을 진행한다. 최적의 조합을 찾는 방식은 회귀분석의 변수 선택법으로 "후진 제거법" 방식을 활용한다. 전체 임베딩 결과를 측정한 이후 개별 성능이 떨어지는 임베딩을 하나씩 제거하면서 결과를 측정한다. Table 4가 후진 제거법으로 진행했을 때 실험 결과이다.

<Table 4> Performance by Embedded Combinations(Backward Elimination)

	Book		Yelp	
	HR@10	NDCG@10	HR@10	NDCG@10
All Embedding	-	-	-	-
8 Elements	-	-	-	-
...	...	...	...	...
N+1 Elements	-	-	-	-
N Elements	-	-	-	-

(3) 최적 조합의 성능 확인(RQ2)

RQ1을 통해서 찾은 최적의 임베딩 조합의 성능이 우수한지 확인하기 위해 다른 베이스라인 모델과 비교한다. Table 5가 실험 결과로, 최적의 조합 이외에도 전체 임베딩을 활용한 성능도 측정하여 비교를 진행한다. 이때 기존의 성능 지표 HR@10과 NDCG@10 이외에도 모델의 Inference Time과 파라미터 크기를 보조로 확인해서 비교를 진행한다.

베이스라인 모델과의 비교를 통해 (1) 최적의 임베딩 조합의 성능이 얼마나 효과적인지 보며, 전체 임베딩을 사용한 모델과 성능 비교를 통해 (2) 최적의 임베딩 조합의 성능이 얼마나 효율적인지 보고자 한다.

<Table 5> Comparison results of different methods

	Book			
	HR@10	NDCG@10	inference-time	Parameter
MLP	-	-	-	-
NeuNF	-	-	-	-
ConvNCF	-	-	-	-
DeepCF	-	-	-	-
DNCF	-	-	-	-
DDFL	-	-	-	-
proposed (Best)	-	-	-	-
proposed (All)	-	-	-	-

5. 결론 및 향후 연구

논문에서는 사용자와 아이템 사이의 다양한 Representation을 생성하는 기존 임베딩 방식의 최적 조합에 대한 실험을 진행했다. 본 연구의 Contribution은 첫 번째 기존 연구에서 제안한 다양한 Embedding 방식(Metric, Matrix)과 내용(ID, Historical, Dual, Semi Dual)에서의 최적의 조합을 찾을 수 있다는 점이다. 최적의 조합을 찾는 실험을

통해서 개별적인 Embedding 방식의 성능을 직관적으로 확인할 수 있으며, 최적의 조합에 대해서 의미론적으로 해석을 제시할 수 있다. 두 번째 기존 연구에서 내적과 concatenation을 적용 후 MLP 사용해 제한적인 representation을 추출한 것과 외적을 활용해 2차원 행렬을 만들어 CNN을 적용해 풍부한 Representation을 생성한다는 점이다. 이는 개별 성능 조합을 통해서 증명하고자 한다.

향후 연구과제로, 다음과 같은 문제를 해결하고자 한다. 첫 번째로 본 연구의 실험 대상이 되는 Embedding 종류는 매우 적고 제한적이다. 그렇기에 선행 연구의 Dual이나 Semi Dual Embedding과 같이 새로운 Representation을 나타내는 임베딩에 대해서 추가 제안 및 탐구할 예정이다. 두 번째로 새로운 임베딩 방법과 종류가 등장할 때마다 최적의 조합을 찾기 위한 실험은 갱신해야 한다. 하지만 조합을 찾기 위한 실험의 경우 많은 Inference time을 요구하기에 이를 개선하기 위한 방식으로 Attetnion Mehanism을 활용한 실험을 진행하고자 한다. 마지막으로 데이터별로 각 임베딩의 성능과 최적의 조합 실험 결과가 상이하게 다를 경우, 이를 일반화하기 어렵다는 한계점이 있다. 그렇기에 더 많은 데이터 세트를 수집 후(5개 이상) 실험을 진행하여 최적의 임베딩 조합에 대한 많은 실험을 진행한다. 그래서 이런 최적의 조합 과정에서 공통으로 등장하는 임베딩과 그것의 특징은 무엇이며, 일반화가 가능 여부를 탐구할 예정이다.

참고문헌

[1] Fajie Yuan, Jinhui Tang, Tat-Seng Chua, Xiangnan He, Xiaoyu Du, Zhiguang Qin, "Modeling Embedding Dimension Correlations via Convolutional Neural Collaborative Filtering", *ACM Transactions on Information Systems*, Vol.37, no 4(2019), 1-22.

[2] B. Smith, G. Linden, J. York, "Amazon.com recommendations: Item-to-item collaborative filtering", *IEEE Internet Comput*, Vol.7, no.1(2003), 76-80.

[3] C. Volinsky, R. M. Bell, Y. Koren, "Matrix factorization techniques for recommender systems", *IEEE Computer*, Vol.42, no 8(2009), 30-37.

[4] L. Nie,He, L. Liao, H. Zhang, T. Chua, X. Hu, "Neural collaborative filtering", *Proceedings of the 26th International Conference on World Wide Web*, WWW 2017, April 3-7, 2017, 173-182.

[5] Feng Tian, Jinhui Tang, Tat-Seng Chua, Xiangnan He, Xiaoyu Du, Xiang Wang, "Outer Product-based Neural Collaborative Filtering", *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, July 13, 2018, 2227-2233.

[6] Dongxing Zhao, Gongshan He, Lixin Dinge "Dual-embedding based Neural Collaborative Filtering for Recommender Systems", A*CM Transactions on Knowledge Discovery from Data*, Vol.15, no.5(2021), 1-24

[7] Jianjun Li, Guohui Li, Syed Tauhid Ullah Shah, Quan Zhou, Zhiqiang Guo, "DDFL: A Deep Dual Function Learning-Based Model for Recommender Systems", *Database Systems for Advanced Applications: 25th International Conference*, September 24-27, 2020, 590-606

[8] Chang-Dong Wang, Jian-Huang Lai, Ling Huang, Philip S. Yu, Zhi-Hong Deng, "DeepCF: A Unified Framework of Representation Learning and Matching Function Learning in Recommender System", The Thirty-ThirdAAAI Conference on Artificial Intelligence, AAAI 2019, Thirty-First

Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, January 27 - February 1, 2019, pp. 61 - 68.

Abstract

Neural Collaborative filtering Using Multilateral Embedding

Min Sik Kim* · Kun Woo Kim*

The core of the CF-based model is "how to represent and model interactions between users and items." Despite the performance improvement and development of recommendation systems with various attempts on embeddings, there are still no studies on finding the optimal method for various embeddings.

So, through this paper, the individual performance of various types of embeddings that appeared in previous studies is confirmed, and then the optimal combination is found through the 'reverse removal method'. Additionally, extrinsic and CNN were used in modeling to observe more correlations between items and users.

We find the optimal combination of each embedding with two data, and demonstrate that the performance is superior to the recent deep learning-based recommendation system.

Key Words : Recommendation System, Deep learning, Embedding, Collaborative Filtering, Survey

* School of Department of Big Data Convergence Management, Kookmin University