Journal of Korea Multimedia Society Vol. 22, No. 4, April 2019(pp. 443-454)
https://doi.org/10.9717/kmms.2019.22.4.443

# 아파치 스파크에서의 PARAFAC 분해 기반 텐서 재구성을 이용한 추천 시스템

임어진†, 용환승††

# PARAFAC Tensor Reconstruction for Recommender System based on Apache Spark

Eo-Jin. Im†, Hwan-Seung. Yong††

## ABSTRACT

In recent years, there has been active research on a recommender system that considers three or more inputs in addition to users and goods, making it a multi-dimensional array, also known as a tensor. The main issue with using tensor is that there are a lot of missing values, making it sparse. In order to solve this, the tensor can be shrunk using the tensor decomposition algorithm into a lower dimensional array called a factor matrix. Then, the tensor is reconstructed by calculating factor matrices to fill original empty cells with predicted values. This is called tensor reconstruction. In this paper, we propose a user-based Top-K recommender system by normalized PARAFAC tensor reconstruction. This method involves factorization of a tensor into factor matrices and reconstructs the tensor again. Before decomposition, the original tensor is normalized based on each dimension to reduce overfitting. Using the real world dataset, this paper shows the processing of a large amount of data and implements a recommender system based on Apache Spark. In addition, this study has confirmed that the recommender performance is improved through normalization of the tensor.

**Key words:** Tensor Reconstruction, Recommender System, PARAFAC Decomposition, Multi-dimensional Recommendation

## 1. 서 론

찰나의 순간에도 수많은 양의 데이터가 생성되는 빅데이터의 시대가 되었다. 사람들은 수많은 정보 속에서 자신에게 필요한 정보를 찾아 헤매는 수고로움을 덜기 위해 보다 자신에게 적합하고, 꼭 필요한 정보를 선별하여 제공하는 시스템에 대한 선호도가 커지고 있다. 또한, 빅데이터를 통해 나이 별, 지역 별, 성별 등의 생각이나 행동, 유행 등을 분석하고 예측할 수 있기 때문에 이를 기반으로 고객의 기존 구매 목록의 취향, 기호를 반영하여 다음 구매할 제품을 자동으로 추천 해 주는 시스템을 추천 시스템이라고 한다. 최근 다양한 분야에서 추천 시스템을 적용한 기업이 인기를 누리고 있으며, 기존의 기업들도 추천

※ Corresponding Author : Hwan-Seung Yong, Address: (03760) 52, Ewhayeodae-gil, Seodaemun-gu, Seoul, Korea, TEL : +82-2-3277-2592, FAX : +82-2-3277-2306, E-mail : hsyong@ewha.ac.kr
Receipt date : Dec. 20, 2018, Revision date : Feb. 22, 2019
Approval date : Mar. 16, 2019

† Dept. of Computer Science & Engineering., Graduate School, Ewha Womans University
(E-mail : erjin0326@ewhain.net)
†† Dept. of Computer Science & Engineering., Graduate School, Ewha Womans University
※ This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education. (2016R1D1A1B03931529)

시스템을 기반으로 한 여러 가지 사업을 진행하고 있으며 대표적인 기업으로는 Netflix, Amazon, Yelp 등이 있다. 이 기업들은 고객으로 하여금 구매한 상품에 대해 만족도 평가를 하도록 하고 이를 통해 고객이 아직 구매하지 않은 제품에 대한 평점을 예측한다. 예측된 평점 값을 기준으로 다음 구매 시 어떤 상품을 추천할 지 결정하는 방식이다. 기존에 쓰인 추천 시스템의 방식은 일반적으로 고객과 제품이라는 두 가지 요소만을 가지고 추천하는 2차원 행렬 기반 추천시스템이었다면, 최근에는 세 가지 이상의 요소를 기반으로 하는 텐서를 기반 추천 시스템의 연구가 활발한 추세이다[1-3]. 텐서란 3차원 이상의 고차원의 행렬 데이터를 의미하며, 다차원 행렬이라고도 한다. 고객이 제품을 구매하는데 있어 제품 자체의 기능성을 고려할 뿐만 아니라 시간 등과 같은 기타 요소를 고려하는 경향을 반영하기 위해 텐서를 기반으로 한 추천 시스템에 대한 연구를 하는데 예를 들어, 여름에 특히 공포 영화의 소비가 잦은 현상이나, 유행이나 기념일 같은 시간적인 요소를 반영하여 제품을 선택하는 현상은 국내뿐 아니라 외국에서도 만연한 소비 형태이기 때문이다. 그러므로 3차원 이상의 고차원의 데이터를 이용한 추천 시스템을 사용할 경우, 2차원 행렬 데이터에서는 발견하지 못했던 잠재 요소를 분석하여 추천에 적용할 수 있고 보다 현실에 적합한 추천을 할 수 있다는 이점이 있다.

또한 텐서를 사용하여 추천 시스템을 구현하게 되면 행렬을 기반으로 할 때보다 데이터의 크기나 양이 기하급수적으로 커지기 때문에 대용량 데이터를 처리하기 위해 빅데이터 분석도구를 활용해야 한다. 인메모리 빅데이터 시스템인 아파치 스파크(Apache Spark)는 RDD(Resilient Distributed Data)라는 분산 데이터 구조를 이용하여 병렬 배치 프로세스로인 메모리 시스템에서 실행이 가능하다. 또한 Scala 프로그래밍을 이용하여 복잡한 분산 처리 코드를 간결하게 구현할 수 있으며 컴퓨터 자체 RAM을 이용하여 데이터를 처리하고 저장하기 때문에 실행 속도가 빠르다는 장점이 있다. 그래서 본 연구에서는 인메모리 빅데이터 분석도구인 아파치 스파크를 이용하여 추천 시스템을 구현하였다.

본 연구에서는 PARAFAC(Parallel Factors Analysis)[4, 5] 분해 기법을 인 메모리 시스템에 맞게 변형한 S-PARAFAC[6]을 기반으로 한 추천 시스템을 제안한다. 초기의 텐서 데이터의 경우, 대부분의 값이 0으로 이루어져 있는데 이렇게 데이터의 대부분이 비어있는 성질을 데이터 희소성이라고 표현하며 행렬의 경우에는 협업필터링[23] 기법을 통해 이를 해소하는 방법이 보편적이며, 고차원의 텐서데이터에 대해서는 텐서 재구성(Tensor Reconstruction)기법을 이용하면, 비어 있는 텐서 값이 계산된 예측 값으로 채워지게 되어 결측 값(Missing value) 문제를 해결하고 텐서의 희소성을 줄일 수 있다. 본 논문에서 제안하는 방법은 다음과 같다. 먼저, 텐서에 정규화 알고리즘을 적용하여 정규화한 뒤, S-PARAFAC을 이용하여 3차원의 텐서 데이터를 분해하여 그에 따른 인수행렬을 구하고, 얻게 된 인수행렬을 이용하여 계산식을 통해 기존의 텐서를 재구성하는 방법이다. 그렇게 재구성된 텐서를 이용하여 사용자 기반의 상위 K개의 추천 목록을 제공하는 알고리즘을 적용하고 이를 통해 추천 성능을 평가하는 것으로 하였다.

## 2. 관련 연구

### 2.1 PARAFAC 분해

고차원의 텐서를 분해하는 방법에는 여러 가지가 있는데 그 중에서 본 연구에서는 PARAFAC(Parallel Factors Analysis) 분해 기법을 사용하여 추천시스템을 구현하였다. PARAFAC분해는 행렬 분해 방법인 SVD(Singular Value Decomposition)의 다차원 버전으로 텐서를 랭크가 1인 텐서들의 합으로 분해하는 방법이다[4-6]. $I\times J\times K$ 의 크기를 갖는 3차원 텐서 $X\in R^{I\times J\times K}$의 경우, 랭크 $r$이 양의 정수이고 $a_r\in R^I, b_r\in R^J, c_r\in R^K$는 세 인수행렬을 나타낸다면 PARAFAC 분해는 다음 Fig. 1과 같이 나타낼 수 있고 식으로 표현하면 다음과 같은 식 (1)로 표현된다[4, 5]. 식 (1)에서 연산 °은 행렬의 내적을 의미한다.

$$Tensor X \approx \sum_{r=1}^{R} a_r \circ b_r \circ c_r \tag{1}$$

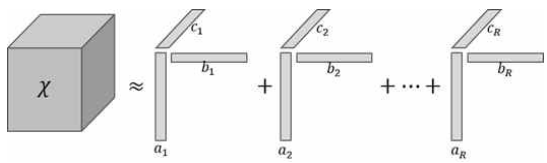


Fig. 1. PARAFAC decomposition of Three-dimensional tensor.

일반적으로 초기 텐서의 경우, 대부분의 값이 결측되었다는 문제가 있고 그로 인해 텐서 자체에 유효한 값이 희소하여(Sparsity) 이를 이용하는데 있어 텐서를 이용한 연구 분야에서 희소성의 문제가 큰 과제로 남아있다. 텐서 재구성[17]은 원래의 텐서로부터 비어있는 곳을 복구하는 것을 의미하며, 비어있는 텐서를 완성시키는 의미로 재구성(Reconstruction)이라는 용어를 이용한다. 이렇게 텐서나 행렬을 완성시키기 위한 방법은 여러 가지가 있지만 가장 우수한 성능을 보이는 것은 분해(Factorization)하는 방법이다. 그리하여 텐서 분해 식 (1)을 성분 별로 표현하면 식 (2)과 같이 표현할 수 있다. 먼저 텐서를 분해한 뒤에, 식 (2)을 참고로 하여 텐서 분해의 계산 과정을 역으로 수행하면 비어있는 텐서의 값을 복구하게 되므로 근사치의 예측 값으로 이루어진 텐서를 재구성하여 완성할 수 있게 된다.

$$Tensor X \approx \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr}, \text{ for } i = 1,\cdots, I, j = 1,\cdots, J, k = 1,\cdots, K \quad (2)$$

### 2.2 텐서 기반 추천 시스템

최근 기존의 2차원 행렬을 넘어 고차원의 텐서를 이용하기 위해 텐서를 분해하는 알고리즘에 대한 연구가 활발히 진행되고 있으며, 그로 인해 텐서 기반의 추천 시스템에 대한 연구 또한 큰 관심을 갖고 있는 추세이다. 하둡을 이용한 텐서 분해 알고리즘에 관한 논문인 BIGtensor[22]에서 언급하였듯이, 텐서를 분해하여 분석하였을 때, 공포 장르의 영화에 대해서 특정 시기에 수요가 증가하였다는 결과를 얻을 수 있었던 것처럼 텐서를 이용하여 추천에 적용하게 되면 2차원 데이터를 이용하였을 때는 발견하지 못한 잠재 요소나 특징을 발견할 수 있기 때문이다. 그리하여 본 논문에서 사용한 PARAFAC 알고리즘 외에도 널리 알려진 HOSVD(Higher Order Singular Value Decomposition)나 Tucker 분해 알고리즘 등을 이용한 추천 시스템을 제안하는 연구도 많이 행해지고 있다.

그 중 HOSVD은 가장 널리 알려진 텐서 분해 알고리즘인데 행렬을 기반으로 한 SVD(Singular Value Decomposition)의 고차원 텐서로의 확장 버전이기 때문이다. 태그 기반 추천시스템[12]이 이에 해당하고 사용자가 노래, 앨범 또는 아티스트와 같은 음악 항목에 태그를 지정하는 시스템의 데이터를 이용하여 사용자, 태그, 음악의 3차원 텐서를 HOSVD를 통해 분해하고 각 축의 관계를 포착하여 추천 모델을 제안하는 MusicBox[9]가 있다. 또한, 장소 기반 데이터를 이용하여 HOSVD를 기반으로 POI(Point-of-Interest) 추천[16]을 하는 연구도 있으며 3차원을 넘어, 사용자, 활동, 시간, 장소라는 4차원의 텐서를 이용한 장소 추천 기법[1]에 대한 연구가 있다. 이러한 장소나 위치를 추천하는 시스템을 POI 추천 시스템이라고 하는데 사용자가 어느 위치를 방문했을 때 다음에 방문할만한 상점이나 위치를 추천해주는 시스템을 말하며 시간대에 따른 사용자의 행동 흐름을 분석하고 단기적 및 장기적인 선호도를 고려하여 POI 추천[2]을 해주는 연구가 이에 해당한다. 게다가 Tucker 분해 알고리즘을 통해 텐서를 분해하여 사용자 맞춤 태그 추천을 위한 PITF(Pairwise Interaction Tensor Factorization)연구[13]도 제안되었고 텐서 분해 알고리즘에 확률의 이론을 적용하여 상품 추천[14]을 하는 연구도 있다. 그 외에도 스마트폰 감지기로부터 사용자의 위치 및 상황 정보와 실제 Facebook 친구 정보를 수집하여 Coupled Matrix and Tensor 분해 알고리즘을 통해 친구 추천 시스템[10]을 제안한 연구가 있으며, 이 분해 알고리즘을 통해 문맥 기반 추천을 바탕으로 학습 분석하는 모델[15]에 대한 연구도 있다. 또한, 사용자 그룹을 둘로 나누어 각 사용자의 기본 정보와 사용자 간의 상호작용에 대한 데이터를 텐서로 모델링하여 PARAFAC 분해 알고리즘을 이용하여 추천에 적용하는 방법으로 Tensor space model을 기반으로 하는 People-to-people 추천 시스템에 관한 논문이다. 그리하여 추천에 사용한 텐서 데이터는 사용자x, 사용자Y, 사용자Y의 속성과 그에 따른 사용자 간의 상호작용 점수로 이루어져 있다. 사용자X가 사용자Y에게 메시지를 보냈을 때 긍정적인 반응이면 1점, 부정적이거나 무응답인 경우 0점으로 표기하여 사용자 간의 네트워크에 관한 추천을 하는 것이다. 그리하여 본 논문에서는 PARAFAC 분해 알고리즘을 이용하여 사용자, 상품, 시간의 세 축과 그에 따른 평점 값을 가지는 텐서를 이용하였으며, 텐서의 희소성을 해소하기 위해 Dropout에서 착안한 정규화 기법을 텐서에 적용하여 정규화한 뒤 사용자 기반 상위 K개의 추천 목록을 반환하는 추천 시스템을 빅데이터 분석도구인 아

파치 스파크를 기반으로 구현하였으며 이를 통해 향상된 추천 성능을 보인다.

## 3. 제안 기법

### 3.1 정규화 알고리즘

추천 시스템 분야에서 결측 값으로 인한 문제는 늘 해결해야 할 문제였다. 이를 효율적으로 처리하는 것이 중요하며 그렇지 않으면 추천 시스템의 성능이 저하된다. 그리하여 드롭아웃 (Dropout) [20, 21] 기법에서 착안한 정규화 모델을 제안한다. 드롭아웃 기법은 일정 비율로 데이터를 삭제하고 나머지만을 다음 단계에 참여시키는 방법으로, 모든 데이터를 가지고 훈련하는 방법보다 전체 데이터 중 성능을 저하시킬 수 있는 일부 요소를 낙오시킴으로써 보다 더 높은 성능을 도출하고자 하는 방법이다. 따라서 본 연구에서 제안하는 정규화 알고리즘은 텐서에서 비어있는 부분을 0이나 음수 같은 부정적인 요소로 표기하는 것이 아니라 드롭아웃의 이론과 같이 낙오된 데이터로 처리하는 방법이다. 일반적인 추천 시스템의 경우, 행렬이나 텐서로 된 데이터를 분해하고 재구성하는 재구성 알고리즘을 통해 추천에 적용하는데 본 연구에서 제안하는 정규화 기법은 재구성 알고리즘을 적용하기 전에 처리하는 방식이다.

2차원 행렬로 예를 들면, 사용자와 상품이라는 두 가지의 축을 기준으로 행렬이 있을 때 각각의 축을 기반으로 정규화를 진행하는 방법이다. 그리하면 사용자 기반 정규화 행렬과 상품 기반 정규화 행렬을 얻을 수 있고 이를 각각 행렬 분해 및 재구성의 과정을 거쳐 추천 알고리즘에 적용할 수 있다. 본래의 입력 데이터 자체가 가지고 있는 값의 개수가 적고 희소하기 때문에 이 0이 아닌 값에 가중치를 주는 방식이며 이 가중치는 드롭아웃에서 데이터를 낙오시킬 비율 $p$을 정해 데이터에 $p$을 나누어 주는 방식에서 착안하여, 데이터의 빈 값을 낙오시키고 0이 아닌 (Non-zero) 값에 가중치를 주는 방식이며 여기서는 비율 $p$ 는 0이 아닌 (Non-zero) 평점 값의 개수에 행이나 열의 크기를 나눈 값이라 정의하였다. 따라서 사용자-상품 행렬 M에 대하여 i번째 행은 $M_i$, j번째 열은 $M^j$라고 표기한다고 하고 사용자의 개수를 $X$, 상품의 개수를 $Y$ 라고 하며 $\|.\|_0$은 각 행이나 열의 0이 아닌 항목의 개수를 의미한다고 할 때, 사용자 기반 정규화 행렬은 u번째 행 $M_u$ 대신 각 요소에 $\frac{Y \bullet M_u}{\| M_u \|_0}$을 계산한 값을 갖는 행렬이 되고 상품 기반 정규화 행렬은 i번째 열 $M^i$대신 $\frac{X \bullet M^i}{\| M^i \|_0}$ 의 값을 갖는 행렬이 된다. 텐서는 세 가지 이상의 축을 가진 고차원 버전의 행렬이기 때문에 위의 정규화 방식을 적용하기 위해 전체 축 중에 두 개씩 쌍을 이루어 진행한다. 그리하여 본 연구에서는 3차원 텐서를 사용하였는데 User×Item×Time으로 축이 구성되어 있다. 따라서 (User, Item), (User, Time), (Item, User), (Item, Time), (Time, User), (Time, Item)의 6가지 경우로 나누어 정규화를 적용할 수 있고 이는 Algorithm 1과 같다.

**Algorithm 1:** Normalization for Tensor

**Input:** Tensor $X \in \mathbb{R}^{I \times J \times K}$

**Output:** Normalized Tensor $X^{ui}, X^{ut}, X^{iu}, X^{it}, X^{tu}, X^{ti} \in \mathbb{R}^{I \times J \times K}$

1: $U \leftarrow$number of Users in I dimension
2: $I \leftarrow$number of Items in J dimension
3: $T \leftarrow$number of Time in K dimension
4: $X^{ui} \leftarrow \frac{I \cdot X^u}{||X^u||_0}$
5: $X^{ut} \leftarrow \frac{T \cdot X^u}{||X^u||_0}$
6: $X^{iu} \leftarrow \frac{U \cdot X^i}{||X^i||_0}$
7: $X^{it} \leftarrow \frac{T \cdot X^i}{||X^i||_0}$
8: $X^{tu} \leftarrow \frac{U \cdot X^t}{||X^t||_0}$
9: $X^{ti} \leftarrow \frac{I \cdot X^t}{||X^t||_0}$
10: **return** $X^{ui}, X^{ut}, X^{iu}, X^{it}, X^{tu}, X^{ti}$

### 3.2 텐서 재구성

본 연구에서 제안하는 방법은 아파치 스파크에서의 정규화 텐서를 바탕으로 텐서를 재구성하고 사용자 기반 추천 알고리즘을 적용하는 것이다. 이를 위해서는 먼저 텐서를 정규화하고 그 다음, PARAFAC 분해를 통해 텐서를 분해한 뒤, 마지막으로 분해된 텐서를 다시 재구성한다. 그런 다음 사용자 기반 Top-K 추천 목록을 도출하여 추천 시스템을 구현한다. 이를 S-PARAFAC을 이용하여 분해하게 되면 세 개의 인수행렬을 결과로 얻게 되는데 본 논문에서는 이 세 가지 인수행렬들을 입력으로 받아 2장에서 언급했던 식(2)을 통해 계산함으로써 텐서를 재구성하여 텐서 데이터의 희소성 문제를 해결하고 비어있는 값에 대한 예측 값을 얻을 수 있다. 인수행렬 데이터를 인 메모리 시스템인 아파치 스파크에 올리기 위해 분산 데이터 모델인 RDD(Resilient Distributed Dataset)형태로 변환하는 작업을 거쳐야 하는데 행

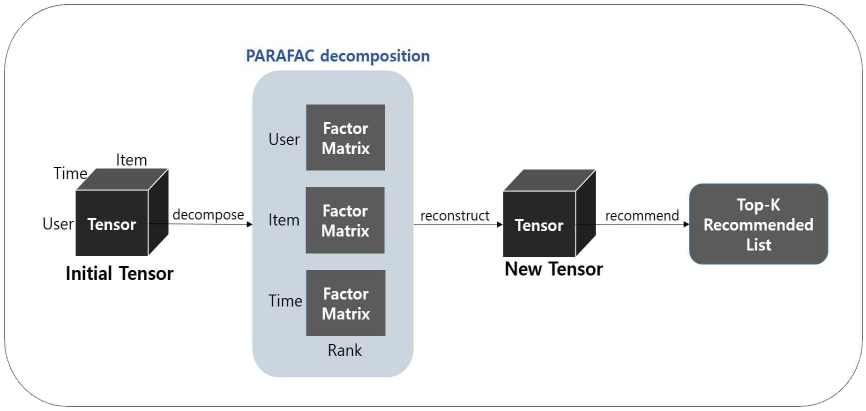


Fig. 2. Recommender System using Tensor Reconstruction.

단위로 계산하기 위해 랭크와 동일한 크기의 벡터로 바꾸어 아파치 스파크에서 제공하는 *IndexedRow Matrix*타입으로의 데이터 변환 과정을 거친다. *IndexedRowMatrix*의 *IndexedRow*는 행렬에서 행의 번호를 나타내는 *long*타입과 행의 요소를 묶어서 나타내는 벡터가 함께 있는 형태로, PARAFAC 알고리즘을 통해 분해하고 재구성을 할 때, 같은 행과 열의 값끼리 연산을 해야 하기 때문에 행의 번호와 열의 위치가 중요한 부분이므로 이 방법을 사용하였다. 또한, 인수행렬A는 텐서 데이터에서 사용자에 대한 행렬인데, 본 실험에서는 텐서를 재구성하게 되면 재구성된 텐서의 크기가 매우 커 실험을 진행하는데 무리가 있어, 임의로 선택된 사용자에 대해서만 텐서 재구성을 진행하였다. 그리하여 랜덤 함수를 이용하여 전체 사용자 중 일부를 무작위로 선택하고 선택된 사용자의 데이터만 변환하였다.

데이터 변환과정을 마치면, PARAFAC Tensor Reconstruction의 핵심 부분인 아다마르 연산(Hadamard)과 합산을 수행하게 된다. 아다마르 연산은 두 벡터를 피 연산자로 하여, 벡터의 가중치를 구하는 계산이며 동일한 자리에 위치한 벡터의 요소끼리만 곱셈 연산을 가능하도록 해 준다. Algorithm 2의 7번째 줄부터 10번까지가 이에 해당하는데, 먼저 인수행렬A와 B를 연산하고 그 결과를 가지고 C와 다시 연산하는 방법이다. 이 때 아파치 스파크 머신러닝 라이브러리 (Machine Learning Library, Mllib)에서 제공하는 *Elementwise Product*를 사용하여 아다마르 연산을 구현하였으며, 이를 이용하여 피 연산자 인수행렬A를 변환하였고 변환된 A′을 B와 곱집합(Cartesian Product, ×)연산을 수행하였고, 그 결과를 다시 나머지 인수행렬 C와 곱집합 연산을 수행하고 이를 합산하여 텐서를 재구성하였다. 텐서 재구성을 마치면 재구성된 텐서는 초기의 텐서와 같은 각 축의 번호와 그에 해당하는 평점 값의 형태로 저장된다. 이 때, 추천 시스템에서는 양수의 값만 고려하기 위해 재구성된 텐서의 값이 음수인 경우는 제외하고 저장한다. 3차원의 정규화 텐서는 Fig. 3과 같이 세 차원의 각각의 번호를 의미하는 숫자 세 개와 그에 해당하는 평점 값의 (user, item, time, rating) 형태로 이루어져있다. 이를 PARAFAC 알고리즘을 이용하여 분해하게 되면 세 개의 인수 행렬을 결과로 얻게 되는데 S-PARAFAC을 이용하여 텐서를 분해하면 인수행렬 데이터는 Fig. 4와 같은 형태로 이루어져 있으며, 앞에서부터 몇 번째 인수행렬인지를 나타내는 인수행렬 코드, 행렬의 행 번호, 행렬의 열이자 랭크 r의 값, 분해된 값을 의미한다. 본 논문에서는 세 개의 인수 행렬을 입력으로 받아 2장에서 언급했

**Algorithm 2:** PARAFAC Tensor Completion

**Input:** Factor Matrices $A,B,C$, rank $R$
**Output:** Tensor $X \in \mathbb{R}^{I\times J\times K}$
1: **U** ←Select Random Users
2: Data Preprocessing: From the factor matrices **A, B, C**, we construct indexed row.
3: $A \leftarrow toIndexedRowByRank(R, A)$
4: $A_U \leftarrow$When matrix A is transformed to IndexedRow by Rank, the rows are filtered by selected users U to reconstruct for selected users
5: $B \leftarrow toIndexedRowByRank(R, B)$
6: $C \leftarrow toIndexedRowByRank(R, C)$
7: $A'_U \leftarrow ElementwiseProduct(A_U)$
8: $AB \leftarrow A'_U \times B$
9: $C' \leftarrow ElementwiseProduct(C)$
10: $X \leftarrow AB \times C'$
11: **return** $X$

| | | | |
|---|---|---|---|
| 27 | 524 | 30 | 4 |
| 27 | 564 | 53 | 3 |
| 27 | 610 | 59 | 4 |
| 27 | 853 | 67 | 5 |
| 27 | 875 | 22 | 4 |
| 27 | 888 | 18 | 5 |
| 27 | 902 | 29 | 2 |
| 27 | 935 | 34 | 5 |
| 27 | 967 | 7 | 3 |
| 27 | 972 | 87 | 5 |
| 27 | 1064 | 72 | 5 |
| 27 | 1289 | 48 | 5 |
| 27 | 1296 | 73 | 4 |
| 27 | 1343 | 7 | 3 |
| 27 | 1400 | 7 | 5 |
| 27 | 1429 | 37 | 5 |
| 27 | 1460 | 29 | 3 |
| 27 | 1503 | 58 | 4 |
| 27 | 1532 | 6 | 3 |
| 27 | 1563 | 23 | 4 |
| 27 | 1651 | 11 | 4 |
| 27 | 1850 | 34 | 4 |
| 27 | 1873 | 29 | 4 |
| 27 | 1903 | 67 | 4 |
| 27 | 1975 | 2 | 5 |
| 27 | 2209 | 24 | 4 |
| 27 | 2212 | 25 | 4 |
| 27 | 2265 | 6 | 4 |
| 27 | 2315 | 68 | 4 |

Fig. 3. Normalized 3-way Tensor dataset example (User, Item, Time, Rating).

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | -0.08136277666936953 |
| 1 | 1 | 2 | 1.1782160151125007 |
| 1 | 1 | 3 | 0.007100185506245041 |
| 1 | 1 | 4 | 0.037924497135977404 |
| 1 | 1 | 5 | -7.573552477402751 |
| 1 | 1 | 6 | 1.7875085278745655 |
| 1 | 1 | 7 | -0.48106089013394465 |
| 1 | 1 | 8 | 0.11186831984358463 |
| 1 | 1 | 9 | -0.5410416739295297 |
| 1 | 1 | 10 | -1.087982528973154 |
| 1 | 2 | 1 | -2.550993112919671E-7 |
| 1 | 2 | 2 | 1.2646057432315953E-7 |
| 1 | 2 | 3 | 6.78750547728363E-9 |
| 1 | 2 | 4 | -1.0179279902464291E-6 |
| 1 | 2 | 5 | -6.10118330002114E-8 |
| 1 | 2 | 6 | 1.7029359869586154E-8 |
| 1 | 2 | 7 | -3.3252231824111445E-8 |
| 1 | 2 | 8 | 2.2701823093313672E-7 |
| 1 | 2 | 9 | -3.528448055376198E-8 |
| 1 | 2 | 10 | -1.8035537348103947E-7 |

Fig. 4. Example of Factor matrix with rank value of 10.

던 식(4)을 통해 계산함으로써 텐서를 재구성하였다. 이를 통해 텐서 데이터의 희소성 문제를 해결하고 비어있는 값에 대한 예측 값을 계산할 수 있다. 예를 들어, 인수행렬 $A, B, C$ 가 있을 때, 식(4)을 통해 텐서를 재구성하는 방법은 Fig. 5와 같다.

## 4. 실 험

### 4.1 실험 데이터 및 환경

본 연구에서 실험을 위해 사용한 데이터 셋은 실제 데이터인 Yelp[7] 데이터와 MovieLens[8] 데이터를 사용하였다. 텐서 데이터는 3차원 텐서 형태로 축 I, J, K는 각각 사용자, Yelp의 경우에는 상호 명, MovieLens는 영화 제목, 그리고 K축은 시간을 의미하는 번호로 구성되어 있으며, 텐서의 값은 사용자가 해당 시간에 해당 상호 혹은 영화에 대해 매긴 평점으로 0.5에서 5사이의 값으로 이루어져 있다. 이 평점은 0이 아닌 값(Non-zero)으로 이루어져 있으며, Table 1에서 Non-zero는 0이 아닌 값의 개수를 의미하고 이를 통해 데이터의 희소성과 밀집 도를 판단할 수 있다. 이 수치를 보면 텐서 데이터의 경우, Yelp 데이터 셋의 Non-zero 개수가 334,166개, MovieLens 데이터 셋은 10,000,054개이며 이를 퍼센티지로 표현하면, Yelp 데이터 셋은 0.0003%, MovieLens 데이터 셋은 0.0083%로 행렬 데이터에 비해 매우 희소함을 알 수 있고, 두 텐서 데이터 셋 중에서는 MovieLens 데이터 셋이 더 밀집도가 크다는 것을 알 수 있다. Table 1의 텐서 데이터 셋으로 PARAFAC분해 알고리즘을 수행하여 텐서를 재구성할 경우, 초기의 데이터 셋에 비해 매우 많은 결과 값을 얻게 된다. Table 1의 텐서 데이터 셋을 보면 Yelp데이터의 경우, 재구성 시 70,817×15,579×108에 해당하는 1,190억 개의 결과 값을 갖게 되고 이를 데이터 사이즈로 환산하면 3.6TB에 육박하게 된다. 마찬가지로, MovieLens데이터 셋도 초기의 텐서 데이터는 희소하기 때문에 크기가 187MB밖에 차지하지 않지만, 재구성하여 완성된 텐서를 얻게 되면 71,567×10,681×157에 해당하는 1,200억 개의 결과 값을 갖게 되며 이는 3.7TB의 용량을 차지하게 된다. 이를 기반으로 계산 해 본 결

Table 1. Tensor Dataset

| Name | I | J | K | Type | Non-zero | | Size |
|---|---|---|---|---|---|---|---|
| Yelp | 70K | 15K | 108 | Input | 334K | 0.0003% | 5.2MB |
| | | | | Output | 119B | 100% | 3.6TB |
| MovieLens | 71K | 10K | 157 | Input | 10M | 0.0083% | 187MB |
| | | | | Output | 120B | 100% | 3.7TB |

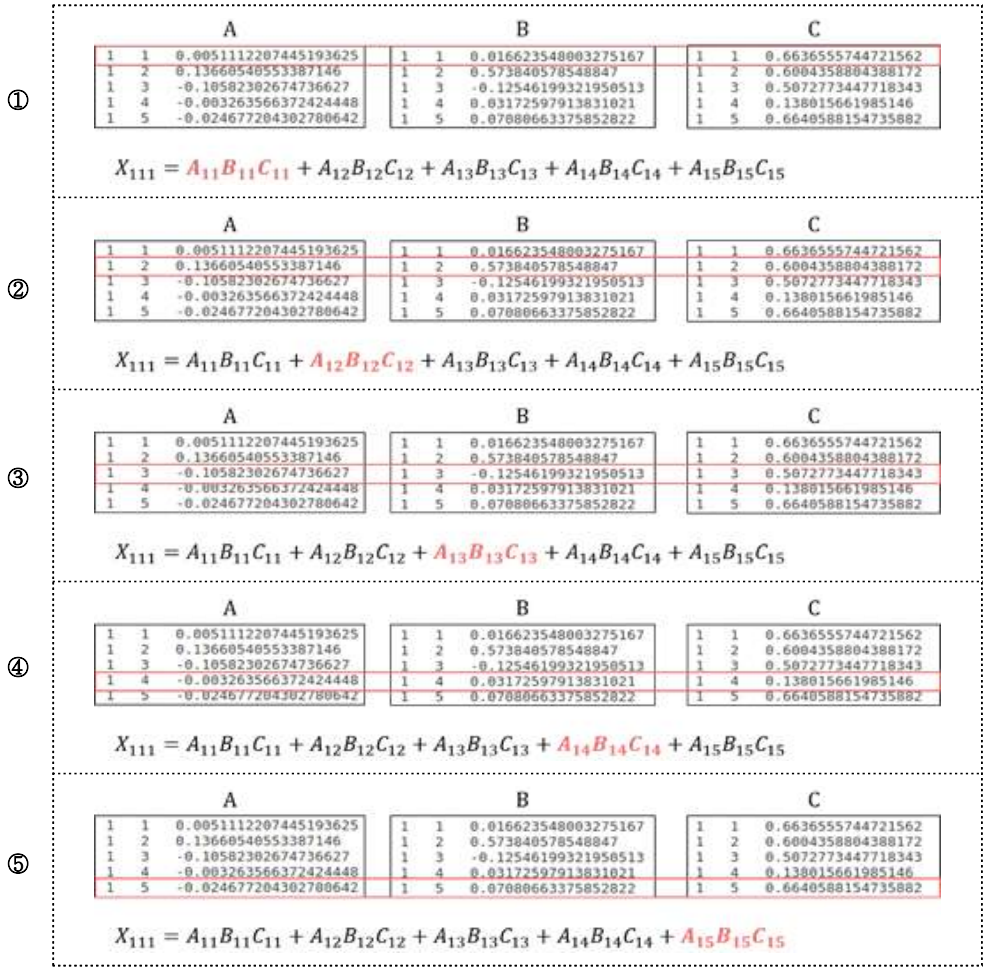


Fig. 5. PARAFAC Tensor reconstruction example with rank value of 5.

과, 사용자 한 명에 해당하는 텐서 데이터의 크기는 대략 50~60MB의 용량을 차지하지 때문에 본 연구에서는 보다 효율적인 연구를 위해 모든 사용자에 대해서 텐서를 재구성하는 대신에 사용자 인수행렬 A의 값을 기준으로 내림차순 정렬하여 상위 50개의 데이터 중 랜덤 함수를 통해 무작위로 선택된 사용자 20명에 대해서만 텐서를 재구성하였다.

본 실험에 사용된 환경은 하둡(Apache Hadoop) 2.6버전의 스파크(Apache Spark) 1.6.1버전을 사용하였으며, 스칼라(Scala) 2.10.6버전을 이용하여 구현하였다. 또한, Intel Core i-7 6700k의 CPU, Ubuntu 18.04 운영체제, 메모리 16GB의 환경에서 실험을 수행하였으며 텐서를 분해할 때 반복횟수를 10회, 랭크 값을 10으로 설정하고 실험을 진행하였다.

### 4.2 평가방법

본 실험에서는 추천 시스템의 성능을 평가하기 위해 전체 데이터의 90%를 무작위로 선택하여 학습 데이터로 사용하고 나머지 10%의 데이터는 학습된 데이터를 검증할 수 있는 테스트 데이터로 사용하였으며 5겹 교차검증(5-fold cross validation)을 하였다. 그리고 학습 데이터로 정규화 과정을 진행한 뒤, 텐서 분해와 재구성 과정을 거쳐 완성된 새로운 텐서를 만든 후, 예측된 값을 기반으로 사용자 기반 Top-K 추천 목록을 생성하여 테스트 데이터 중 3점 이상의 평가를 받은 데이터를 필터링한 결과와 평가를 진행하였다. 추천 알고리즘의 역할은 훈련 데이터를 기반으로 관련 항목 목록을 사용자에게 제공하는 것이므로 추천 성능 평가는 추천 목록을 기반으로 진행

된다. 이에 사용된 평가 방법은 Precision@K, Mean Average Precision(MAP)과 NDCG(Normalized Discounted Cumulative Gain)[18]이다. Precision@K는 테스트 데이터의 추천 목록 중에 예측된 추천 목록에 있는 상위 K개의 비율을 측정하는 방법이다. 이 기준에서는 추천 목록의 순서는 고려되지 않는다. MAP은 추천 문서의 순서가 고려되는 방법으로 매 Recall단계마다 Precision의 값을 계산하여 평균 낸 값을 의미한다. NDCG@K는 이상적인 추천 결과에 대해서 예측된 추천 결과의 성능을 상대 평가하는 방법이다. 이는 기존의 Precision이나 Recall이 추천 결과의 순서와는 관계없이 절대적인 점수를 도출하는 방식으로는 순위에 따른 차별 점을 부과하기 어렵다는 점에 기반을 두어 나온 방법으로 추천 목록의 순서에 따른 패널티를 적용한 점수 값을 도출한다.

### 4.3 실험 결과 및 분석

본 연구의 실험 결과는 다음과 같으며 Top-K 추천 목록에서 K의 값을 최소 5에서 최대 50으로 하여 그에 따른 추천 성능을 보고자 하였다. 3차원 텐서의 경우 본 논문에서 제안하는 정규화 기법을 적용하는 경우 6개의 정규화 텐서를 반환하게 되고 각각 User-Item 텐서, User-Time 텐서, Item-User 텐서, Item-Time 텐서, Time-User 텐서, Time-Item 텐서라 지칭하였다. 행렬과 마찬가지로 Precision, MAP, NDCG를 이용하여 성능을 평가하였고 Precision과 NDCG는 추천 목록의 개수 K에 따라 성능을 확인할 수 있도록 하였다. Precision과 NDCG의 그래프가 상이한 것을 볼 수 있는데, 이는 사용된 평가방법의 특성에 따라 달리 나온 것으로 해석할 수 있다. Precision의 경우, 추천 목록에 있는 항목의 순서와 관계없이 일치하는 개수만을 고려한 평가 방법이며, NDCG의 경우에는 추천 항목의 순서에 따라 패널티를 부여하는 계산 방식이기 때문에 이 같은 결과를 보인 것이다. Fig. 6(a)에서 Yelp 데이터 셋의 Precision의 결과는 Top-5 추천 목록을 반환하였을 때 User-Item 정규화 텐서가 정규화하지 않은 텐서보다 1.3배 높은 것을 알 수 있지만 그 외의 부분에서는 낮은 것으로 확인되었다. Fig. 6(b)에서와 같이 NDCG 성능 평가를 적용하였을 때는 정규화하지 않은 원래의 텐서보다 User-Item 정규화 텐서가 월등한 성능을 나타냈으며 상위 20개의 추천 목록 기준으로 최대 3.4배까지 향상되는 결과를 보였다. 게다가 Precision에서는 두각을 보이지 못했던 Item-Time 정규화 텐서 또한 원래 텐서보다 향상된 성능을 보였으며 상위 5개의 추천 목록에서 약 1.8배까지 향상되었다. Fig. 6을 보면 Yelp 데이터 셋의 성능 평가 그래프가 K값이 20인 지점을 기점으로 그래프의 모양 변한다는 것을 알 수 있다. Precision의 경우에도 정규화하지 않은 원래 텐서의 성능이 증가하는 추세를 보이다가 K=20인 지점부터 하락하는 모습을 보이며, 나머지 정규화 텐서들의 경우에도, 급격히 감소하다가 K=10과 20의 사이를 지나면서부터 완만해지는 결과를 보인다.

또한, MovieLens 텐서 데이터 셋의 성능 평가 결과를 Fig. 7로 나타내었는데, Precision을 이용하여 성능을 평가했을 때, 정규화한 텐서의 성능이 정규화하지 않는 텐서보다 낮아지는 형세를 보였다. 그러나 Fig. 7(b)에서처럼 추천 항목의 순서를 고려하는 NDCG 방법을 적용했을 때는 상위 5개부터 20개까지의 추천 목록에서는 낮았으나, User-Item 정규화 텐서와 User-Time 정규화 텐서에서 K=30을 기점으로 성능이 점차 향상되는 양상을 뗐다. Fig. 7을 모두 살펴보았을 때 MovieLens 데이터 셋은 K값이 30인

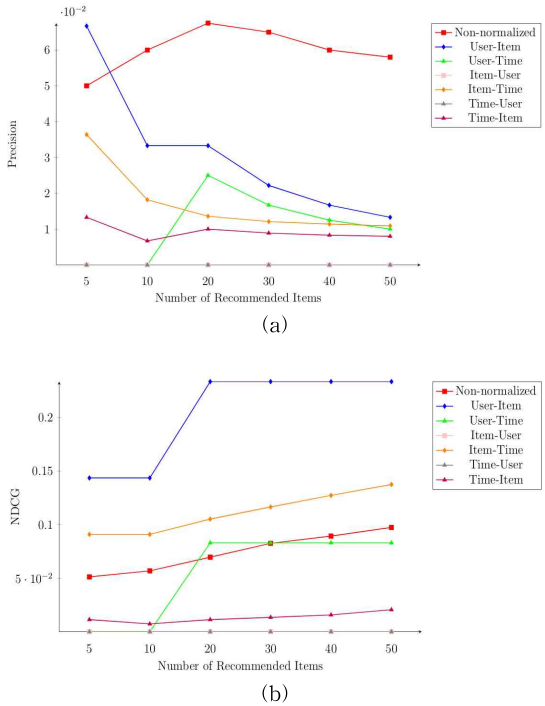


Fig. 6. Precision(a) and NDCG(b) evaluation of Yelp dataset.

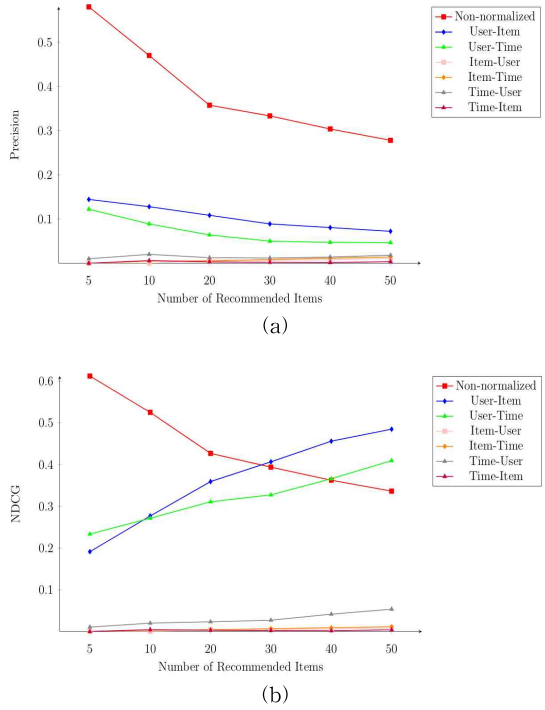


Fig. 7. Precision(a) and NDCG(b) evaluation of MovieLens dataset.

부분을 기점으로 하여 성능에 변화가 있는 것을 알 수 있다.

게다가 원래의 텐서와 정규화 텐서들끼리의 MAP 성능 결과를 Fig. 8에서 살펴보면, MAP 또한 NDCG와 마찬가지로 추천 목록에 있는 항목의 순서를 고려

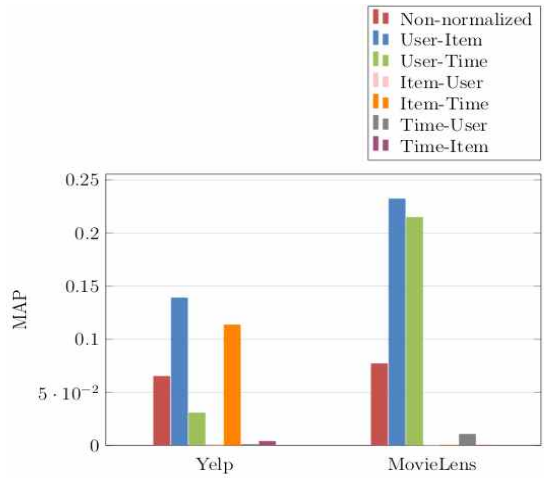


Fig. 8. MAP (Mean Average Precision) evaluation of Tensor dataset.

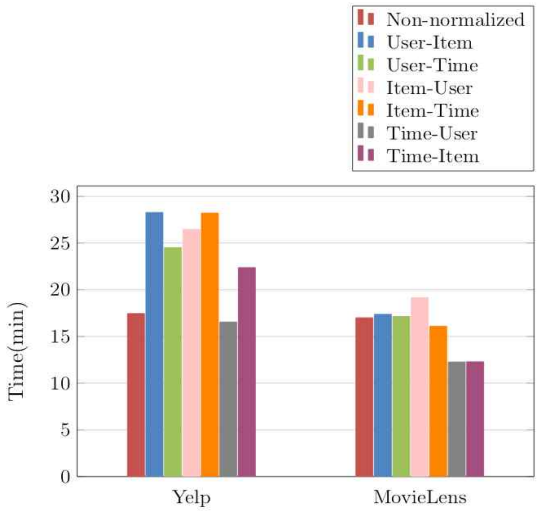


Fig. 9. Execution time of reconstruction of normalized Yelp, MovieLens tensor.

하는 평가 방법이기 때문에 NDCG와 유사한 결과가 도출되었다. 그리하여 Yelp 데이터 셋의 경우, User-Item 정규화 텐서의 추천 성능이 정규화하지 않은 원래의 텐서를 이용하여 추천을 했을 때보다 약 2.1배의 향상된 결과를 보였으며, Item-Time 정규화 텐서의 경우에도 성능이 약 1.7배 향상된 것으로 나타났다. 마찬가지로 MovieLens 데이터 셋의 경우에도 MAP 평가방법을 적용하였을 때, 정규화 텐서를 이용한 추천 시스템의 성능이 정규화하지 않은 텐서 기반 추천 시스템의 성능보다 높게 나타나는 것으로 확인되었다. User-Item 정규화 텐서의 경우, 약 3배 향상되었으며, User-Time 정규화 텐서의 경우에는 약 2.8배 높아짐을 보였다. 또한, 분해된 텐서를 다시 재구성하는데 걸리는 시간을 측정하여 그래프로 나타낸 것은 Fig. 9와 같다. 어떻게 정규화 하는지에 따라 시간의 차이가 다소 있었으나 Yelp 데이터의 경우 최소 17분, 최대 28분 정도 소요되었으며 평균적으로는 23분의 재구성 소요시간이 걸리는 것을 알 수 있었다. 마찬가지로 MovieLens 데이터의 경우에도 재구성하는데 걸리는 시간이 최소 12분, 최대 19분으로 평균 16분의 시간이 소요되었다.

## 5. 결론 및 향후 연구

본 논문에서는 3차원 텐서 데이터를 이용하여 정규화 알고리즘을 적용하고 인 메모리 빅데이터 시스

템인 아파치 스파크 기반에서 텐서 재구성 알고리즘을 이용한 추천시스템을 제안하였다. 아파치 스파크 기반의 PARAFAC 분해 방법인 S-PARAFAC으로 텐서 분해하여 텐서 재구성 기법을 이용해 텐서를 재구성하여 데이터 대부분의 값이 비어있던 텐서를 완성하였다. 또한, 이를 통해 사용자 기반 Top-K 추천 리스트를 반환하는 추천 시스템을 제안하였다. 또한 추천 시스템에서 성능을 평가하는 데 있어 주요한 방법으로 쓰이는 Precision@K, 추천 목록의 항목의 순서를 고려하는 평가 방법인 MAP와 NDCG방법을 통해 추천 성능을 평가하였고 선행연구로 진행했던 정규화하지 않은 3차원 텐서를 이용한 추천시스템[19]의 성능과 비교하였는데 Yelp 텐서 데이터 셋의 경우, Precision은 최대 약 3.4배, NDCG의 경우 최대 2.1배까지 향상된 결과를 보였으며, MovieLens 텐서 데이터 셋의 경우, MAP 평가방법을 적용하였을 때, 추천 성능이 최대 3배 향상되는 것을 확인할 수 있었다. 이를 통해, 정규화한 데이터를 이용하여 추천 알고리즘을 적용하였을 때 성능이 향상되는 효과를 보임을 알 수 있었고 데이터의 특성에 따라 상위 몇 개의 추천 목록을 반환하느냐에 따라 추천 성능이 달라짐을 알 수 있었고, 어떤 데이터에 대해서는 과적합이 일어나 K값이 커질수록 성능이 낮아지는 결과를 보였다. 그리하여 데이터의 특성에 따라 각기 다른 최적의 정규화 방식이 존재한다.

따라서, 축을 기준으로 정규화를 진행하는 알고리즘을 적용한 결과, 정규화 행렬과 정규화 텐서가 어느 축을 기준으로 했는지에 따라 각기 다른 성능 결과를 보였는데 향후 이 부분에 대한 분석을 진행하여 데이터에 특성에 맞는 정규화 기법에 대해 고안하고자 한다. 또한 추천 시스템 분야에서 여러 가지 우수한 알고리즘을 혼합 적용하는 연구 사례가 생겨나고 있기 때문에 텐서 분해의 결과인 인수행렬을 분석할 수 있는 알고리즘에 대한 연구를 하여 성능을 향상시킬 수 있는 방안을 찾고자 한다.

## REFERENCE

[1] P. Bhargava, T. Phan, J. Zhou, and J. Lee, “Who, What, When and Where: Multi-Dimensional Collaborative Recommendations Using Tensor Factorization on Sparse User-Generated Data,” *Proceedings of the 24th International Conference on World Wide Web,* pp. 130-140, 2015.

[2] X. Li, M. Jiang, H. Hong, and L. Liao, “A Time- Aware Personalized Point-of-Interest Recommendation via High-Order Tensor Factorization,” *Journal of Association for Computing Machinery Transactions on Information Systems - Special Issue: Search, Mining and their Applications on Mobile Devices,* Vol. 35, Issue 4, pp. 1-23, 2017.

[3] F. Zhang, N.J. Yuan, K. Zheng, D. Lian, X. Xie, and Y. Rui, “Exploiting Dining Preference for Restaurant Recommendation,” *Proceedings of the 25th International Conference on World Wide Web,* pp. 725-735, 2016.

[4] H. Richard A, “Foundations of the PARAFAC Procedure: Models and Conditions for an ’Explanatory’ Multi-modal Factor Analysis. *UCLA working papers in phonetics* 16: pp. 1-84, 1970.

[5] T.G. Kolda and B.W. Bader, “Tensor Decompositions and Applications,” *Society for Industrial and Applied Mathematics Review,* Vol. 51, No. 3, pp. 455-500, 2009.

[6] H. Yang and H. Yong, "S-PARAFAC: Distributed Tensor Decomposition Using Apache Spark," *Journal of Korean Institute of Information Scientist and Engineers,* Vol. 45, No. 3, pp. 280-287, 2018.

[7] Yelp Tensor Dataset, https://datalab.snu.ac.kr/bigtensor/yelp.php (accessed May, 12, 2018).

[8] MovieLens Tensor Dataset, https://datalab.snu.ac.kr/bigtensor/movielens.php (accessed May, 12, 2018).

[9] A. Nanopoulos, D. Rafailidis, P. Symeonidis, and Y. Manolopoulos, “MusicBox: Personalized Music Recommendation based on Cubic Analysis of Social Tags,” *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 18, No. 2, pp. 407-412, 2010.

[10] K. Kim, T. Kim, and S.J. Hyun, “Friend Recommendation Using Offline and Online Social

Information for Face-To-Face Interactions," *Proceeding of 2016 IEEE International Conference on Smart Computing,* pp. 1-5, 2016.

[11] S. Kutty, L. Chen, and R. Nayak, "A People-To-People Recommendation System Using Tensor Space Models," *Proceeding of the 27th Annual Association for Computing Machinery Symposium on Applied Computing*, pp. 187-192, 2012.

[12] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos, "Tag Recommendations based on Tensor Dimensionality Reduction," *Proceeding of the 2008 Association for Computing Machinery Conference on Recommender Systems*, pp. 43-50, 2008.

[13] S. Rendle and L. Schmidt-Thieme, "Pairwise Interaction Tensor Factorization for Personalized Tag Recommendation," *Proceeding of the Third Association for Computing Machinery International Conference on Web Search and Data Mining*, pp. 81-90, 2010.

[14] N. Ifada and R. Nayak. "Tensor-based Item Recommendation Using Probabilistic Ranking in Social Tagging Systems," *Proceeding of the Companion Publication of the 23rd International Conference on World Wide Web Companion*, pp. 805-810, 2014.

[15] F.M. Almutairi, N.D. Sidiropoulos, and G. Karypis, "Context-aware Recommendation-Based Learning Analytics Using Tensor and Coupled Matrix Factorization," *IEEE Journal of Selected Topics in Signal Processing,* Vol. 11, No. 5, pp. 729-741, 2017.

[16] S. Maroulis, I. Boutsis, and V. Kalogeraki, "Context-aware Point of Interest Recommendation Using Tensor Factorization," *Proceeding of 2016 IEEE International Conference on Big Data,* pp. 963-968, 2016.

[17] H. Ge, K. Zhang, M. Alfifi, X. Hu, and J. Caverlee, "DisTenC: A Distributed Algorithm for Scalable Tensor Completion on Spark," *Proceeding of 2018 IEEE 34th International Conference on Data Engineering,* pp. 137-148, 2018.

[18] Evaluation Metrics in Ranking systems, https://spark.apache.org/docs/1.6.1/mllib-evaluation-metrics.html (accessed Dec., 12, 2018).

[19] E. Rim and H. Yong, "Recommender System Using Tensor Reconstruction of Three-dimensional Tensor Data," *Proceeding of Korea Information Science Society Conference,* pp. 113-115, 2018.

[20] H. Lee and J. Lee, "Scalable Deep Learning-based Recommendation Systems," *Informations and Communications Technology Express (ICT) Express,* 2018.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Reserch*, Vol. 15, No. 1, pp. 1929-1958, 2014.

[22] N. Park, B. Jeon, J. Lee, and U Kang, "BIGtensor: Mining Billion-Scale Tensor Made Easy," *Proceeding of the 25th Association for Computing Machinery International Conference on Information and Knowledge Management*, pp. 2457-2460, 2016.

[23] J. Kim, "A Recommendation Method of Similar Clothes on Intelligent Fashion Coordination System," *Journal of Korea Multimedia Society*, Vol. 12, No. 5, pp. 688-698, 2009.

임 어 진

2015년 이화여자대학교 컴퓨터공학 졸업(학사)
2015년~2017년 롯데정보통신 재직
2019년 이화여자대학교 컴퓨터공학 졸업(석사)

관심분야 : 데이터 마이닝, 추천 시스템, 데이터베이스 시스템, 딥 러닝

용 환 승

1983년 서울대학교 컴퓨터공학과 졸업(학사)
1985년 서울대학교 컴퓨터공학과 졸업(석사)
1985년~1989년 한국전자통신연구소 연구원
1994년 서울대학교 컴퓨터공학과 졸업(박사)
2002년 IBM T.J.Watson 연구소 방문연구원
1995년~현재 이화여자대학교 컴퓨터공학과 교수
관심분야 : 데이터베이스 시스템. 데이터마이닝. 사물인터넷 컴퓨팅, 딥 러닝