JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지 Vol. 22, No. 5: 715~721, May 2018

Spark를 이용한 항목 추천 기법에 관한 연구

윤소영[1] · 윤성대[1*]

Item Recommendation Technique Using Spark

So-Young Yun[1] · Sung-Dae Youn[1*]

[1*]Department of Computer Engineering, Pukyong National University, Pusan 48513, Korea

요 약

모바일 기기의 확산으로 소셜 네트워크 서비스나 전자상거래 사이트의 사용자 수가 급증하고 있고 사용자들이 남긴 데이터의 양도 기하급수적으로 증가하고 있다. 그로 인해 전자 상거래 기업들은 사용자들이 남긴 방대한 양의 데이터로부터 어떻게 유용한 정보를 추출할 것인가 하는 과제를 갖게 되었다. 이러한 문제를 해결하기 위해 추천 시스템에 빅 데이터 처리 기법을 적용한 다양한 연구들이 이루어지고 있다. 본 논문에서는 Apache Spark 플랫폼에서 Tag 가중치를 적용한 협업 필터링 기법을 사용한 추천방식을 제안한다. 제안하는 기법은 추천의 정확성을 높이기 위해 전처리 과정에서 Tag 데이터를 정제하고 아이템을 분류한 후 아이템 평가값에 기간 정보와 Tag 가중치를 적용하여 사용한다. RDD(Resilient Distributed Dataset)를 생성한 후 아이템 유사도와 예측값을 구하고 사용자에게 아이템을 추천한다. 실험을 통해 제안 하는 기법이 대량의 데이터를 빠르게 처리하고 추천의 적합성도 향상되는 것을 확인하였다.

ABSTRACT

With the spread of mobile devices, the users of social network services or e-commerce sites have increased dramatically, and the amount of data produced by the users has increased exponentially. E-commerce companies have faced a task regarding how to extract useful information from a vast amount of data produced by the users. To solve this problem, there are various studies applying big data processing technique. In this paper, we propose a collaborative filtering method that applies the tag weight in the Apache Spark platform. In order to elevate the accuracy of recommendation, the proposed method refines the tag data in the preprocessing process and categorizes the items and then applies the information of periods and tag weight to the estimate rating of the items. After generating RDD, we calculate item similarity and prediction values and recommend items to users. The experiment result indicated that the proposed method process large amounts of data quickly and improve the appropriateness of recommendation better.

키워드 : 추천기법, 협업필터링, 아파치 스파크, 확장성, 태그

Keywords : Recommendation Technique, Collaborative Filtering, Apache Spark, Scalability, Tag

Received 28 March 2018, **Revised** 8 April 2018, **Accepted** 7 May 2018

*** Corresponding Author** Sung-Dae Youn(E-mail:sdyoun@pknu.ac.kr, Tel:+82-51-629-6242)
Department of Computer Engineering, Pukyong National University, Pusan 48513, Korea

Open Access http://doi.org/10.6109/jkiice.2018.22.5.715 pISSN:2234-4772

Ⅰ. 서 론

정보통신기술의 발달과 모바일기기의 확산으로 소셜 네트워크 서비스나 전자상거래 사이트의 사용자 수가 급증하고 있다. 온라인 구매가 증가하면서 사용자들이 원하는 제품을 보다 쉽게 검색하고 구매할 수 있도록 많은 전자상거래 기업들이 추천 시스템을 사용하고 있다. 추천 시스템을 사용하는 전자 상거래 기업들은 시스템을 통해 수집되는 방대한 양의 구매 데이터로부터 유용한 정보를 추출하여 상품 추천에 활용하기 위해 다양한 노력을 기울이고 있다.

추천시스템은 사용자들이 선호하는 상품 리스트를 찾기 위해 입력 받은 선호 정보를 사용하며[1], 협업 필터링(collaborative filtering) 기법은 추천 시스템 중 가장 많이 사용되는 방식으로 목표 사용자와 유사한 선호도를 가진 사용자들을 추출하여 이들의 선호도에 기반하여 목표 사용자에게 아이템을 추천한다[2].

그러나 협업 필터링 기법은 대규모의 데이터를 처리하는 과정에서 확장성 문제와 평가 데이터의 희소성 문제, 새로운 아이템의 추천과 관련된 cold start 문제 등을 갖고 있다[3]. 이러한 협업 필터링의 문제점을 해결하기 위해 다양한 연구가 지속적으로 이루어지고 있다. 최근에는 특히 폭발적으로 증가하고 있는 데이터를 효율적으로 처리하기 위해 분산처리 프레임 워크인 하둡(Hadoop)이나 빠른 계산 작업을 수행할 수 있는 클러스터링 연산 플랫폼인 아파치 스파크(Apache Spark)를 사용한 다양한 연구들[4][5][6][7][8]이 이루어지고 있다.

본 논문에서는 추천의 정확성을 높이고 확장성 문제를 줄이기 위해 Tag 정보를 적용한 협업 필터링 기법을 아파치 스파크 플랫폼에서 처리하는 방법을 제안하고자 한다.

제안하는 기법은 추천의 정확성을 향상시키기 위해 아이템에 대한 평가값만을 추천에 사용하는 기존의 아이템 기반 협업필터링 기법에 사용자들이 아이템에 대한 선호를 좀 더 적극적으로 표현하기 위해 남긴 Tag 데이터를 가중치로 사용한다. Tag 데이터는 전처리 과정에서 데이터를 정제하여 <TagID, Tag> 형태의 텍스트 문서를 생성하고, 정제된 Tag를 사용하여 기존의 Tag 데이터를 <userID, itemID, TagValue> 형태의 문서로 만든다. 또한, 사용자들의 평가 데이터에 시간의 흐름에 따른 선호도 변화를 적용하기 위해 기간 정보를 적용하여 아이템을 신규 아이템과 일반 아이템으로 분류하고, 일반 아이템으로 분류된 데이터만을 예측값 생성을 위해 사용한다. 일반 아이템의 평가값에 TagValue를 가중치로 적용하여 RDD 객체를 생성 한 후 아이템 유사도와 예측값을 구하고, 최종적으로 Top-N개의 아이템을 사용자에게 추천한다.

제안하는 기법은 대량의 데이터를 사용함에 따라 발생하는 확장성의 문제를 줄이고, 추천과정을 빠르고 안정적으로 처리하기 위해 인메모리 방식의 아파치 스파크 플랫폼에서 병렬로 추천 과정을 처리함으로써 추천의 적합성과 효율성을 향상시키는 것이 가능하다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구인 협업필터링과 아파치 스파크에 관하여 기술하고, 3장에서는 제안하는 기법에 대하여 기술한다. 4장에서는 실제 데이터를 사용하여 제안하는 기법의 성능을 평가한다. 마지막으로 5장에서는 결론을 기술한다.

Ⅱ. 관련 연구

2.1. 협업 필터링

협업 필터링은 아이템에 대한 목표 사용자의 평가값과 다른 사용자의 평가값을 사용하여 목표 사용자가 좋아할 만한 아이템을 추천하는 기법으로[2] 데이터의 처리 방식에 따라 메모리 기반 알고리즘과 모델 기반 알고리즘으로 분류된다[9].

메모리 기반 알고리즘은 예측을 생성하기 위해 전체 user-item 데이터베이스를 활용한다. 이 알고리즘은 목표 사용자와 유사한 평가나 구매 이력을 가진 이웃을 찾고 이웃들의 제품에 대한 선호를 기반으로 예측 또는 상위 N개의 추천을 생성한다. 사용자 기반 협업 필터링, 아이템 기반 협업 필터링이 이 알고리즘 방식을 사용하며 가장 선호되는 방식이다. 사용자 기반 협업 필터링은 목표 사용자와 유사한 선호도를 가진 사용자들을 근접이웃으로 지정하고 이들의 아이템 평가값을 기반으로 아이템을 추천하는 기법이고, 아이템 기반 협업필터링은 아이템들 간에 유사도를 측정하여 목표 사용자가 선호하는 아이템과 유사한 아이템의 평가값을 기반으로 아이템을 추천하는 기법이다[10].

모델 기반 알고리즘은 사용자 데이터베이스를 사용하여 모델을 추정하거나 학습 한 다음 예측에 사용한다

[9]. 확률적 접근법을 사용하고 베이지안 네트워크, 클러스터링, 규칙 기반 접근법과 같은 기계학습(machine learning) 알고리즘에 의해 실행된다[10].

메모리 기반 알고리즘은 희소성, 확장성, cold start의 문제점을 가지고 있고 모델 기반 알고리즘은 메모기 기반 협업필터링의 단점을 보완하지만 모델 설정에 소요되는 시간이 매우 크며 추가 데이터에 따라 모델 재설정에 소요되는 시간이 매우 커 모델이 빈번하고 급속하게 업데이트 되어야하는 환경에는 적합하지 않다는 문제점이 있다[10][11].

2.2. Apach Spark

UC Berkeley AMPLab에서 개발한 아파치 스파크는 고급 분석을 수행하기 위한 분산형 인메모리 오픈소스 클러스터 컴퓨팅 프레임워크이다[12]. 병렬 컴퓨팅 모델로 널리 사용되던 하둡의 반복 계산과 실시간 계산에서의 낮은 효율성을 극복하기 위해 메모리 기반으로 설계되어 대량의 데이터를 처리하는 연구에 새로운 해결책을 제공한다[13]. 아파치 스파크는 연산 과정을 클러스터 전체에 걸쳐 자동으로 병렬화해 분산 배치된 연산 작업들의 모음으로 표현하는데 이 모음을 탄력적 분산 데이터 세트(RDD, Resilient Distributed Dataset)라 부른다. RDD는 분산 데이터와 연산을 위한 스파크의 핵심적인 개념이다. 스파크에서 모든 작업은 새로운 RDD를 만들거나, 존재하는 RDD를 변형하거나, 결과 계산을 위해 RDD에 연산을 호출하는 것 중의 하나로 표현되며 내부적으로 스파크는 자동으로 RDD에 있는 데이터들을 클러스터에 분배하며 클러스터 위에서 수행하는 연산들을 병렬화한다. 스파크는 기존에 각각 분리된 분산 시스템에서 돌아가던 배치 애플리케이션, 반복 알고리즘, 대화형 쿼리, 스트리밍과 같은 다양한 작업 타입을 동시에 커버할 수 있게 설계되었다. 또 Python, Java, Scala, SQL API 및 강력한 라이브러리를 내장해 지원하며 다른 빅데이터 툴과도 매우 잘 연동되어 있다[14]. 그림 1은 Spark 구성요소를 나타낸다.

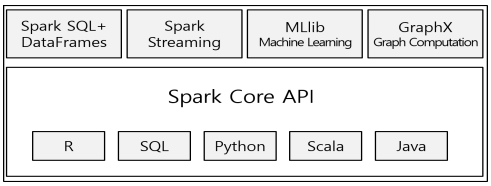

Fig. 1 Spark Architecture

Spark SQL은 Spark Core 상단의 컴포넌트로, 구조화된 데이터와 반구조화된 데이터를 지원하는 모듈이다. Spark Streaming은 실시간으로 배치 및 스트리밍 데이터를 처리하거나 분석하고 대화형 및 분서 애플리케이션을 수해할 수 있는 또 다른 모듈이다. MLlib (Machine Learning Library)는 실용적인 머신러닝을 확장가능하고 사용하기 쉽게 만들고자 개발되었다. Graphx는 그래프 기반 시스템을 구축하기 위한 새로운 Spark API며, 그래프-병렬 처리 계산 엔진 및 분산 프레임워크이다[12].

Ⅲ. Apache Spark을 이용한 추천기법

본 장에서는 아이템의 선호도 변화와 아이템에 대한 Tag 가중치를 적용한 협업 필터링 기법을 아파치 스파크 플랫폼에서 처리하는 제안 기법에 대하여 설명한다.

제안하는 기법은 전처리 단계, 예측값 계산 및 추천 단계로 이루어진다. 그림 2는 시스템의 구성도이다.

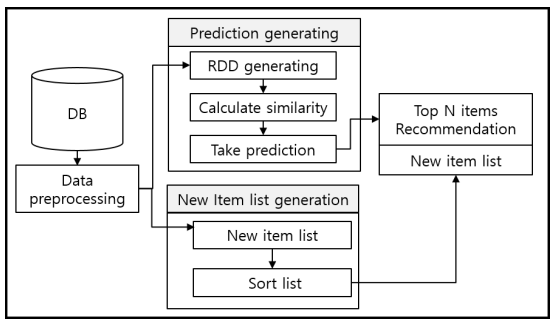

Fig. 2 System Structure Diagram

3.1. 전처리 단계

Tag는 평가값 이외에 사용자가 해당 아이템에 대한 선호를 좀 더 적극적으로 표현한 것으로 다른 사용자들의 구매에도 영향을 줄 수 있다. 따라서 본 논문에서는 사용자에게 더 정확한 추천을 제공하기 위해 사용자들이 아이템에 남긴 Tag 데이터를 정제하여 사용한다. 먼저 아이템에 대한 Tag 정보를 값으로 사용하기 위해 R을 사용하여 Tag 데이터에서 무의미한 단어, 기호, 숫자 등을 제거한다. 남은 Tag 데이터는 의미가 같은 여러 표현들을 하나의 명사로 표현하는 과정을 거친다. Tag데

이터를 정제한 후 Tag 데이터 관리를 위해 <TagID, Tag> 파일로 저장한다. 정제된 Tag 파일을 사용하여 <userID, itemID, TagValue> 파일(itemTag)을 생성한다. 이때 TagValue는 Tag의 중요도를 판단하는 것이 아닌 해당 아이템에 대한 의미 있는 태그의 개수를 의미하므로 값을 1로 지정한다.

다음으로 아이템에 대한 사용자들의 선호 변화를 적용한 추천을 위해 아이템을 스파크 SQL을 사용하여 전처리한다. 사용자들이 아이템에 남긴 평가데이터와 아이템 등록일, 판단 기준일을 지정하고 기준일에서 아이템 등록일까지 값이 임계값(α) 이하이면 신규 아이템으로 나머지는 일반 아이템으로 분류한다.

신규 아이템은 <userID, itemID, rating> 데이터를 아이템 등록일을 기준으로 정렬하여 파일(newItem)로 저장한다. 일반 아이템들은 평가값이 3이상인 데이터만을 사용하여 각 아이템에 대해 기준일까지의 평가 수와 전체 평가 수를 구한다. 전체 평가 수 중 기준일까지의 평가 수의 비율이 임계값(β) 이하이면 지속적으로 사용자들에게 선택되어지는 아이템이므로 해당 데이터만을 선택하여 파일(itemRating)로 저장한다. 마지막으로 아이템 평가값에 Tag 가중치를 적용하여 유사도 계산에 사용하기 위해 itemRating 파일과 itemTag 파일을 사용하여 user-item 매트릭스 RDD (itemM)를 생성한다.

3.2. 예측값 계산 및 추천 단계

전처리 단계를 거쳐 생성된 RDD를 사용하여 유사도를 계산하고 예측값을 생성하기 위해 아이템 기반 협업 필터링 기법을 사용한다. 먼저 조정 코사인 유사도 계산식을 사용하여 아이템 간 유사도를 계산하고 아이템 유사도 매트릭스 RDD(itemSim)를 생성한다. 생성된 매트릭스에서 상위 N개의 근접 이웃을 선정한다. 식(1)은 유사도를 계산하는 식으로 $\overline{R}_u$는 사용자 u의 평가값 평균이고, $R_{u,i}$, $R_{u,j}$ 는 각각 사용자 u의 아이템 i와 j에 대한 평가값을 나타낸 것이다.

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_u)(R_{u,j} - \bar{R}_u)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_u)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_u)^2}} \quad (1)$$

그림 3은 아이템 유사도 매트릭스를 생성하고 근접 이웃을 선정하는 과정을 나타낸 것이다.

```
input :
  itemM : The RDD representing the user-item matrix.
  k : Select the most similar k items.
  Sim(i,j) : The function to calculate the adjust cosine similarity
    between item i and item j.
  NearestNeighbors(K, k) : The function that takes in the list of neighbor
    items to K, and outputs the k items with the highest similarity to K.
output :
  itemSim : The RDD representing item similarity matrix.

  itemSim ← Map(Sim( ), item)
  itemSimGrp ← GroupByKey(itemSim)
  itemSim ← FlatMap(NearestNeighbors(k), itemSimGrp)
```

Fig. 3 Compute similarity and select neighbors

다음으로 user-item 매트릭스에서 선정된 이웃 아이템의 평가값을 사용하여 예측값을 계산하고 상위 N개의 추천 아이템을 생성한다. 식(2)는 예측값 계산식으로 $P_{u,i}$는 사용자 u의 아이템 i에 대한 예측값이고, T_u는 사용자 u에 의해 평가된 아이템 집합을 나타낸다.

$$P_{u,i} = \frac{\sum_{j \in T_u} sim(i,j) \times R_{u,j}}{\sum_{k \in T_u} |sim(i,j)|} \quad (2)$$

그림 4는 예측값 계산 및 Top-N개의 추천 아이템 생성 과정을 나타낸 것이다.

```
input :
  itemM : The RDD representing the user-item matrix.
  itemSim : The RDD representing the item similarity matrix.
  nearestItems : The top k similarity items for each item.
  n : The number of item recommendation to return.
  Prefunc(K,i) : The function to calculate the item recommendations for
    each user K using formula(2)
output :
  itemRec : The RDD representing Top-N item recommendation for the
    active user u.

  itemMS ← GroupByKey(itemM)
  itemRec ← Map(Prefunc(nearestItems, n), itemMS)
```

Fig. 4 Compute Top-N recommendation items

위의 단계를 모두 마친 후 신규 아이템은 newItem 파일을 사용하여 최근 등록일 순으로 사용자에게 신규 아이템으로 추천하고 기준일 이전에 등록된 일반 아이템은 추천단계를 거쳐 추출된 Top-N개의 아이템을 사용자에게 추천한다.

Ⅳ. 실험 및 평가

4.1. 실험 환경 및 실험 데이터

실험은 1개의 마스터 노드와 5개의 슬레이브 노드로 구성된 클러스터 환경에서 실행된다. 각 노드는 Intel Core i5-2400 3.1GHz, 8GB 메모리로 구성되고, 각 노드들에는 Apache Spark 2.2.0 와 scala 2.11.8이 설치되었다.

제안 기법의 성능 평가를 위해 MovieLens 10M 데이터 셋을 사용하였다. 데이터 셋은 71,567명의 사용자가 10,681개의 영화를 선호도 1~5의 값으로 평가한 10,000,000개의 평가 값과 95,580 개의 태그들을 가진다. 실험을 위해 데이터를 80% training data와 20% test data로 나눈 후 training data를 사용하여 추천 아이템을 추출하고 그 결과를 test data를 사용하여 비교한다.

4.2. 실험 결과

실험에서는 제안 기법의 확장성을 평가하기 위해 병렬 시스템의 확장성과 효율성 측정에 가장 많이 사용되는 speedup[15]을 사용하고, 추천 성능을 평가하기 위해서 협업필터링 기법에서 많이 사용되는 RMSE(Root Mean Squard Error)를 사용한다.

확장성 평가를 위한 Speedup은 계산 노드 수의 증가에 따른 효율성의 변화를 나타내는 것으로 동일 데이터를 사용하여 노드 수에 변화를 주었을 때 speedup의 값이 더 높을수록 병렬 효율성과 성능이 더 좋고 할 수 있다[16]. 식(3)은 speedup 계산식으로 T_s는 단일 노드에서 알고리즘 실행 시간을 나타내고, T_p는 p개의 노드에서의 실행시간을 나타낸다.

$$S_p = \frac{T_s}{T_p} \tag{3}$$

본 논문에서는 아이템 크기에 따른 speedup 성능 측정을 위해 데이터 셋의 아이템 수를 10K, 100K, 500K, 1M개로 다르게 구성하고 노드 수를 1~5로 변화를 주어 실험을 진행하였다.

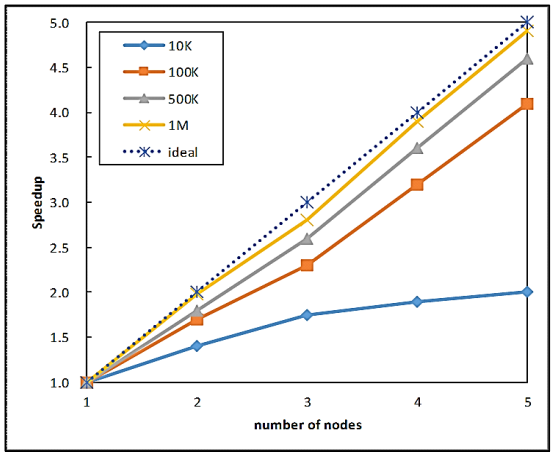

Fig. 5 Result of speedup test

그림 5는 speedup 실험 결과를 나타내는 것으로 10K 데이터 셋을 제외하고는 노드 수가 증가함에 따라 speedup 효과를 보여준다. 10K 데이터 셋은 아이템 수가 많지 않아 노드 수가 3개 이하일 때까지는 확장성 효과를 보이지만 4개 이상에서는 노드 추가에 따른 효과를 볼 수 없었다.

추천 성능을 평가하기 위한 RMSE는 예측값과 실제 평가값 사이의 편차를 나타내는 것으로 그 값이 작을수록 추천이 정확하다. 식(4)는 RMSE를 계산하는 식으로 N은 모든 사용자들이 평가한 아이템의 수를 나타내고, $p_{u,m}$은 아이템 m에 대한 사용자 u의 예측 평가값이다. $r_{u,m}$은 사용자 u의 아이템 m에 대한 실제 평가값이다 [17].

$$\mathrm{SE} = \sqrt{\frac{1}{\mathrm{N}}\sum_{\mathrm{u,m}}(\mathrm{p_{u,m}} - \mathrm{r_{u,m}})^2} \tag{4}$$

RMSE 성능측정을 위해 근접 이웃 수의 변화에 따른 기존 협업필터링 기법인 아이템 기반 협업필터링, 사용자 기반 협업 필터링 기법들과의 비교 실험으로 진행하였다. 그림 6은 실험 결과를 나타낸 것으로 아이템 기반, 사용자 기반 협업필터링 기법과의 비교에서 제안하는 기법의 추천 성능이 약 2% ~ 7% 향상됨을 보여준다.

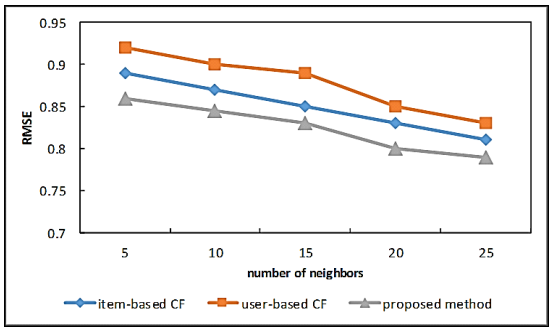

Fig. 6 Comparison of RMSE

Ⅴ. 결론 및 향후 연구 방향

본 논문에서는 추천의 정확성을 높이고 빠른 처리를 위해 Tag 정보를 적용한 협업 필터링 기법을 분산 플랫폼에서 처리하는 방법을 제안하였다. 제안 기법에서는 정확한 추천을 위해 데이터 전처리 과정에서 Tag 가중치와 아이템에 대한 사용자들의 선호도 변화를 적용하였고, 전처리된 데이터를 사용하여 유사도 계산 및 예측값 계산 단계를 거쳐 추천 아이템을 생성하였다. 대량의 데이터를 빠르게 분산처리하기 위해 아파치 스파크 플랫폼 상에서 아이템 분류 및 연산 과정을 처리하였다. 제안 기법의 성능 평가를 위한 실험에서 제안 기법이 대규모 데이터 처리 시 확장성이 더 향상되며, 기존의 협업 필터링 기법보다 추천의 정확성도 2% ~7% 정도 향상되는 결과를 볼 수 있었다.

스파크 플랫폼을 사용한 제안 기법은 연산을 위한 노드의 수를 추가하면 처리속도가 향상되는 결과를 보였지만 노드 추가는 비용을 발생시키는 문제와 직결되므로 향후에는 좀 더 효율적으로 노드를 관리하며 정확한 추천이 이루어질 수 있는 추천기법에 대하여 연구하고자 한다.

ACKNOWLEDGEMENT

This work was supported by a Research Grant of Pukyong National University(2017 year)

REFERENCES

[1] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, “Grouplens: an open architecture for collaborative filtering of Netnews,” *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pp. 175 - 186, New York, NY, USA, Oct. 1994.

[2] G. Linden, B. Smith, and J. York, “Amazon.com recommendations: item-to-item collaborative filtering,” *IEEE Internet Computing*, vol. 7, no. 1, pp. 76 - 80, January/ February 2003.

[3] S. Meng, W. Dou, X. Zhang, and J. Chen, "KASR : A Keyword-Aware Service Recommendation Method on MapReduce for Big Data Applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no.12, pp.3221-3231, Dec. 2014.

[4] Y. Shang, Z. Li, W. Qu, Y. Xu, Z. Song, and X. Zhou, “Scalable Collaborative Filtering Recommendation Algorithm with MapReduce,” *2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing*, pp.103 -108, Aug. 2014

[5] P. Ghuli, A. Ghosh, and R. Shettar, “A collaborative filtering recommendation engine in a distributed environment,” *2014 International Conference on Contemporary Computing and Informatics(IC3I)*, pp. 568-574, Nov. 2014.

[6] B.Kupisz, O. Unold, “Collaborative Filtering Recommendation Algorithm based on Hadoop and Spark”, *2015 IEEE International Conference on Industriial Technology(ICIT)*, pp.1510-1513, Mar. 2015.

[7] C. Sardianos, I. Varlamis, and M. Eirinaki,“ Scaling Collaborative Filtering to large-scale Bipartite Rating Graphs,” *2017 IEEE Third International Conference on Big Data Computing Service and Applications*, pp.70-79, Apr. 2017.

[8] S. Yun, S. Youn, “Recommendation System Using Big Data Processing Technique,” *Journal of Korea Institute of Information and Communication Engineering*, Vol. 21, No.6, pp.1183-1190 , Jun. 2017.

[9] J.S.Breese, D.Heckerman, and C. Kadle, “Empirical Analysis of Predictive Algorithms for Collaborative Filtering,” in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pp.43-52, Jul. 1998.

[10] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based Collaborative Filtering Recommendation Algorithms," in *Proceedings of the 10th International World Wide Web Conference, ACM Press*, pp. 285-295, 2001.

[11] O.J. Lee, E.S. You, “Predictive Clustering-based Collaborative Filtering Technique for Performance-Stability of Recommendation System”, *Journal of Intelligence and Information Systems*, vol.21, no. 1 , pp.119-142, Mar. 2015.
[12] S.K. Gorakala, *Building Recommendation Engines*, Birmingham, Packt Publishing, 2016.
[13] S. Yang, B. Wu, “Large Scale Video Data Analysis Based on Spark,” 2015 *International Conference on Cloud Computing and Big Data(CCBD)*, pp.209-212, Nov. 2015.
[14] M. Zaharia, H. Karau, A. Konwinski, and P. Wendell *Learning Spark*, Sebastopol. CA: O’REILLY, 2015.
[15] U. Ramachandran, H. Venkateswaran, A. Sivasubramaniam, and A. Singla, “Issues in understanding the scalability of parallel systems,” *in Proceedings of the First International Workshop on Parallel Processing*, pp.399-404. Dec. 1994.
[16] S.H.H. Ding, B.C.M. Fung and P. Charland, “Kam1n0: MapReduce-based Assembly Clone Search for Reverse Engineering,” *KDD’16 Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 461-470, Aug. 2016.
[17] A. Stanescu, S. Nagar and D. Caragea “A Hybrid Recommender System: User Profiling from Keywords and Ratings,” *WI-IAT ’13 Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence(WI) and Intelligent Agent Technologies(IAT)*, vol. 01, pp. 73-80, Nov. 2013.

윤소영(So-Young Yun)

2007.2. 부경대학교 경영대학원 국제물류학과 경영학석사
2011.2. 부경대학교 전자상거래 협동과정 공학박사
※관심분야 : 추천시스템, m-commerce, 빅데이터 등

윤성대(Sung-Dae Youn)

제18권 2호 참조
1989~현재 부경대학교 컴퓨터공학과 교수
※관심분야 : 병렬처리, 멀티캐스팅통신, 데이터마이닝 등