JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 21, No. 6 : 1183~1190 Jun. 2017

빅 데이터 처리 기법을 적용한 추천 시스템에 관한 연구

윤소영[1] · 윤성대[2*]

Recommendation System Using Big Data Processing Technique

So-Young Yun[1] · Sung-Dae Youn[2*]

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

요 약

네트워크와 IT 기술의 발전으로 사용자들은 장소에 구애 받지 않고 어디서든 본인이 원하는 아이템을 검색하고 구매하고 있다. 이에 따라 추천시스템에서 급증하는 데이터로 인한 확장성 문제를 어떻게 해결할 것인가에 대한 연구들이 다양하게 진행되고 있다. 본 논문에서는 Tag 가중치를 적용한 아이템 기반 협업 필터링 기법과 분산 병렬 처리 방식인 MapReduce 방법을 적용한 추천 기법을 제안한다. 제안하는 기법은 속도 향상과 효율성을 위해 전처리 과정에서 아이템을 카테고리별로 분류하고 노드 수에 맞게 그룹지은 후 사용한다. 각 분산 노드에서 4번의 Map-Reduce 단계를 통해 데이터 처리를 진행하는데 사용자에게 더 나은 아이템을 추천하기 위해 유사도 계산에서 아이템 Tag 가중치를 사용한다. 마지막 Reduce 단계를 거쳐 출력된 예측값 중 상위 N개의 아이템을 추천에 사용한다. 실험을 통해 제안 하는 기법이 대량의 데이터를 효율적으로 처리하며 기존의 아이템 기반 기법보다 추천의 적합성도 향상되는 것을 확인하였다.

ABSTRACT

With the development of network and IT technology, people are searching and purchasing items they want, not bounded by places. Therefore, there are various studies on how to solve the scalability problem due to the rapidly increasing data in the recommendation system. In this paper, we propose an item-based collaborative filtering method using Tag weight and a recommendation technique using MapReduce method, which is a distributed parallel processing method. In order to improve speed and efficiency, the proposed method classifies items into categories in the preprocessing and groups according to the number of nodes. In each distributed node, data is processed by going through Map-Reduce step 4 times. In order to recommend better items to users, item tag weight is used in the similarity calculation. The experiment result indicated that the proposed method has been more enhanced the appropriacy compared to item-based method, and run efficiently on the large amounts of data.

키워드 : 추천기법, 협업필터링, 맵리듀스, 확장성, 태그

Key word : Recommender Technique, Collaborative Filtering, MapReduce, Scalability, Tag

Received 27 January 2017, **Revised** 06 February 2017, **Accepted** 13 February 2017

* **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 https://doi.org/10.6109/jkiice.2017.21.6.1183 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

네트워크와 IT 기술이 발전하면서 다양한 모바일 기기 사용이 보편화되었다. 따라서 사용자들은 시간과 장소에 제한 없이 자신이 원하는 아이템이나 정보를 찾기 위해 모바일 기기를 이용한다. 기업들은 구매 형태의 다양화에 따라 온라인에서 사용자들이 이용 가능한 서비스나 제품들에 대해 가치 있는 정보를 찾는데 도움을 주기 위해 추천 시스템을 사용한다. 추천 시스템은 사용자들이 선호하는 아이템 리스트를 찾기 위해 입력 받은 선호정보를 사용한다[1].

협업 필터링(collaborative filtering) 기법은 추천 시스템으로 가장 많이 사용되는 방식[2]으로 목표 사용자와 유사한 선호도를 가진 사용자들을 추출하여 이들의 선호도에 기반 하여 목표 사용자에게 아이템을 추천하는 방식이다[3]. 그러나 전통적인 협업 필터링은 대규모의 데이터를 분석하거나 다룰 때 확장성의 문제를 갖고 있으며[4], 아이템에 대한 평가값만을 예측에 사용하므로 상품평과 같은 구매자의 적극적 의사 표현을 반영하지 못하는 문제점을 갖고 있다[5].

협업 필터링의 단점인 확장성 문제는 처리비용과 연결되는 문제로 최근 데이터의 폭발적인 증가로 인한 데이터 처리비용과 효율성 향상을 위해 빅데이터 분산 처리 프레임워크 하둡 맵리듀스(Hadoop Map- Reduce)를 적용한 다양한 연구들이 이루어지고 있다[6-9]. 또한 사용자들의 평가값 이외에 상품에 남긴 추가 정보인 상품평과 같은 태그(Tag) 정보를 분석하여 활용하는 연구들[4,5,10,11]도 다양하게 이루어지고 있다.

본 논문에서는 사용자에게 좀 더 정확한 추천을 위해 협업 필터링 기법에 태그 정보를 적용하고 더불어 확장성의 문제를 보완하기 위해 분산 처리 방식인 맵리듀스를 적용한 추천 기법을 제안하고자 한다,

제안하는 기법은 전처리과정을 통해 카테고리별로 아이템을 분류한 후 공통 평가 개수가 임계값 이상인 아이템 간의 유사도를 계산하고 임계값 이상의 유사도를 가진 아이템을 최근접 이웃을 지정한다. 최근접 이웃들의 평가값을 기반으로 목표 사용자가 평가하지 않은 아이템에 대한 예측값을 태그 가중치를 적용하여 계산하고 예측값이 높은 상위 N개의 아이템을 사용자에게 추천하는 방식으로 진행된다. 전체적인 흐름은 협업 필터링 기법의 추천 단계를 따르지만 대량의 데이터를 효율적으로 다루기 위해 n개의 노드들에 데이터를 분산한 후 각 노드에서 4단계의 MapReduce 단계를 거쳐 예측값을 출력하고 아이템을 추천한다.

제안하는 기법은 협업 필터링의 확장성 문제를 해결하기 위해 빅데이터 분산 처리 방식을 사용하면서도 좀 더 정확한 아이템 추천을 위해 카테고리별 아이템에 정제된 태그 가중치를 사용함으로써 추천의 정확성과 효율성을 향상키는 것이 가능하다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구인 협업 필터링과 MapReduce에 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 필터링 알고리즘은 사용자의 선호도와 유사한 선호도를 가진 다른 사용자들의 구매 정보를 사용하여 사용자에게 아이템이나 서비스를 추천하는 기법으로 추천 시스템으로 가장 널리 사용되고 있다[2].

협업 필터링 알고리즘에는 사용자 기반 알고리즘과 아이템 기반 알고리즘이 있다. 사용자 기반 알고리즘은 목표 사용자와 유사한 선호도를 가진 사용자들을 근접이웃으로 지정하고 이들의 아이템 평가값을 기반으로 아이템을 추천하는 기법으로 희소성과 확장성의 문제점을 가지고 있다. 아이템 기반 알고리즘은 아이템들 간에 유사도를 측정하여 목표 사용자가 선호하는 아이템과 유사한 아이템의 평가값을 기반으로 아이템을 추천하는 기법으로 사용자 기반 알고리즘의 단점을 많이 보완하지만 희소성과 cold-stat의 문제점을 가진다[12,13].

협업 필터링 기법의 추천은 4단계로 구성된다. 첫 단계에서는 사용자들이 아이템을 평가한 데이터를 기반으로 사용자-아이템 매트릭스를 생성하고 이를 기반으로 두 번째 단계에서 아이템 간에 유사도를 계산한다. 계산된 유사도를 기반으로 세 번째 단계에서 최근접 이웃을 선정하고 마지막 단계에서 최근접 이웃의 평가값을 기반으로 예측값을 생성하고 상위 N개의 아이템을 목표 사용자에게 추천한다.

식(1)은 아이템간의 유사도를 계산하는 피어슨 상관계수 식으로 $r_{u,i}$, $r_{u,j}$ 는 각각 아이템 i와 j에 대한 사용자 u의 평가값이고 U는 아이템 i와 j를 모두 평가한 사용자들의 집합을 의미한다. 식(2)는 목표 사용자 u의 평가하지 않은 아이템t에 대한 평가값을 예측하는 계산하는 식이다.

$$sim(i,j) = \frac{\sum_{u \in U}(r_{u,i} - \overline{r_i})(r_{u,j} - \overline{r_j})}{\sqrt{\sum_{u \in U}(r_{u,i} - \overline{r_i})^2}\sqrt{\sum_{u \in U}(r_{u,j} - \overline{r_j})^2}} \quad (1)$$

$$P_{ut} = \frac{\sum_{i=1}^{c} R_{ui} \times Sim(t,i)}{\sum_{i=1}^{c} Sim(t,i)} \quad (2)$$

2.2. MapReduce

하둡은 대규모의 데이터 처리 분석을 위한 소프트웨어 프레임워크를 제공하는 아파치의 오픈 소스 프로젝트로서 분산 파일 시스템인 HDFS(Hadoop Distribu -ted File System)와 분산 처리 시스템인 맵리듀스로 구성되어 있다[14]. HDFS와 맵리듀스 모두 마스터(master)/슬래이브(slave) 구조를 가진다. HDFS에서는 마스터를 네임노드(NameNode)로 슬래이브를 데이터노드(DataNode)라고 부르며 독립적으로 분산 파일 시스템으로 쓰일 수 있다.

맵리듀스는 마스터를 잡 트래커(Job Tracker)로 슬래이브를 태스크 트래커(Task Tracker)로 부르며, HDFS와 같은 분산파일 시스템을 데이터 읽기와 쓰기를 위해 반드시 필요로 한다. 또한 맵리듀스는 맵(Map)과 리듀스(Reduce)의 두 단계로 구성이 된다. 맵 단계에서는 <key, value> 형태의 데이터를 입력받아 처리를 통해 새로운 <key, value>로 변환하여 출력한다.

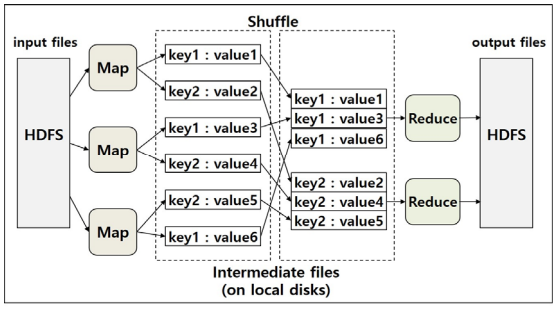

Fig. 1 Basic behavior of Map and Reduce

리듀스 단계에서는 맵에서 출력된 레코드들에서 같은 키 값을 갖는 레코드들을 묶은 <key, list of values>를 입력받아 처리한 후 새로운 <key, value>를 출력한다. 맵과 리듀스는 여러 대의 머신에서 실행 가능하며 프레임워크가 알아서 처리해준다[15]. 그림 1은 맵과 리듀스의 기본 동작을 보여준다.

Ⅲ. 빅데이터 처리 기법을 적용한 추천기법

본 장에서는 추천의 정확성과 효율성을 높이기 위해 아이템 기반 협업 필터링 기법에 아이템 태그 가중치와 분산 처리 방식을 적용한 제안 기법에 대하여 설명한다. 제안하는 기법은 전처리 단계, 예측값 계산 및 추천 단계로 이루어지며 두 번째 단계에서 4번의 맵과 리듀스가 실행된다. 그림 2는 시스템의 구성도이다.

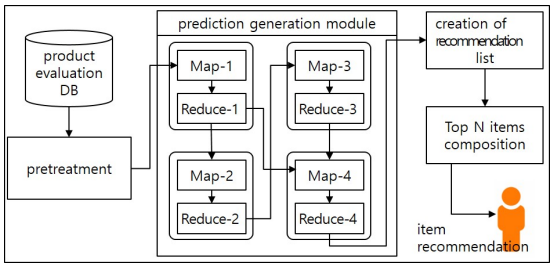

Fig. 2 System Structure Diagram

3.1. 전처리 단계

추천 시스템에는 매일 수많은 아이템들이 등록되고 있으며 아이템 중에는 지속적으로 사용자들에게 선택되는 아이템이 있는 반면 등록 후 초기 얼마간만 선택되고 이 후에는 선택이 되지 않는 아이템들이 있다. 본 논문에서는 추천의 정확성을 높이기 위해 아이템을 카테고리별로 분류한 후 시간의 흐름에 따른 아이템에 대한 사용자들의 평가값 변화를 적용한다. 기준일(i_{fd})을 정하여 기준일로부터 등록일(i_{rd}) 까지 값을 계산하여 임계값(α) 이하일 경우에는 기존의 아이템 평가값 평균을 구하는 식을 사용하고 i_{fd}와 i_{rd}의 계산값이 α 이상이고 아이템 수가 임계값(β) 이상일 경우 아이템 등록일 이후 전체 평가값 평균과 일정 기간 동안의 평가값 평균을 모두 반영한 식을 사용하여 평가값 평균을

구하고 그 결과가 3이상인 아이템만을 대상으로 사용한다.

또한 아이템에 대한 태그 정보는 사용자의 아이템에 대한 선호를 추가적으로 표현하는 것으로 다른 사용자의 아이템 구매에도 영향을 줄 수 있다. 그러나 사용자들은 매우 다양한 방식의 표현을 사용하며 동일한 사용자가 동일한 아이템에 여러 번 태그를 남기거나 구매하지 않은 아이템에 대해 태그를 남기기도 한다. 따라서 아이템에 대한 사용자들의 태그입력 횟수를 유사도 계산에 가중치로 사용하기 위해 아이템 별로 동일한 사용자 아이디로 동일한 날짜에 남겨진 태그는 그 수에 상관없이 1회로 지정하고 평가값 없이 남겨진 태그는 실제 구매유무가 모호하므로 제외한다.

식(3)은 시간 변화가 적용된 평가값 평균 $\overline{R_i}$를 계산하는 식으로 l은 아이템 i를 평가한 사용자들의 집합이고 j는 아이템 i를 평가한 j번째 사용자를 의미한다. N은 전체 기간 동안 해당 아이템을 평가한 사용자의 수를 의미하고 Np는 기준일 동안 해당 아이템을 평가한 사용자의 수를 의미한다.

$$\begin{cases} if\left(i_{fd}-i_{rd}\right)\le\alpha, \\ \quad \overline{R_i}=\dfrac{\sum_{j=1}^{l}R_{i,j}}{N} \\ elseif\left(\left(i_{fd}-i_{rd}\right)>\alpha\ and\ N_p\ge\beta\right), \\ \quad \overline{R_i}=\left(\dfrac{\sum_{j=1}^{l}R_{i,j}}{N}+\dfrac{\sum_{j=1}^{m}R_{i,j}}{N_p}\right)/2 \\ else\ \ 0 \end{cases} \tag{3}$$

위의 조건으로 데이터를 전처리하여 카테고리별로 (itemID, userID, rating, tagCnt) 값을 텍스트파일로 저장한다.

3.2. 맵리듀스를 이용한 예측값 계산 및 추천 단계

전처리 단계를 거쳐 카테고리별로 저장된 파일들을 4번의 맵리듀스 단계를 거쳐 예측값을 생성한다.

첫 번째 맵리듀스는 전처리 된 텍스트를 입력받아 사용자아이디별로 아이템 아이디, 평가값, 태그카운트를 리스트로 출력하는 단계이다. 매퍼는 입력 파일을 userID를 키로 (itemID, rating, tagCnt)를 값으로 지정하여 결과를 출력하고 리듀서는 튜플들을 userID를 기준으로 역 색인을 적용한 후 userID를 기준으로 (itemID, raing, tagCnt) 리스트를 출력한다. 다음 그림 3은 사용자아이디별로 평가한 아이템들의 리스트를 생성하는 과정을 나타낸 것이다.

```
procedure: Mapper-I
input : pdata(itemID,userID,rating,tagCnt) from m users to n items
output : tuples mℜ (<userIDs, (itemID,rating, tagCnt)..>)
  for each userID i in pdata
    key =userID
    value=(itemID,rating,tagCnt)
  end for
  emit tuples<key,value>

procedure: Reducer-I
input : tuples mℜ (<userIDs, (itemID,rating,tagCnt)..>)
output : tuples rℜ (<userIDs, Listitem>)
  Listitem ← List(itemID,rating,tagCnt)
  for each userID i=1 to m do
      templist ← new list
      for each userID j=1 to m do
        if i==j then
          add (itemID,rating,tagCnt) of tuples mℜ in templist
          Listitem=templist
      end for
  end for
  emit <userID, Listitem>
```

Fig. 3 Create list by userIDs

두 번째 맵리듀스는 임의의 두 아이템 간의 유사도를 계산하고 그 결과를 출력하는 단계이다. 매퍼는 이전 단계의 결과를 사용하여 임의의 두 아이템을 공통 평가한 모든 사용자의 평가값을 사용하여 아이템 아이디를 키로 각 아이템의 평가값과 태그 카운트를 값으로 한 튜플 <(itemID$_i$, itemID$_j$),($rating_i$, tagCnt$_i$ $rating_j$, tagCnt$_j$)> 을 출력한다. 리듀서는 튜플들을 사용하여 태그 가중치를 적용한 아이템간의 유사도를 계산한 후 임의의 두 아이템들에 대한 <(itemID$_i$, itemID$_j$), sim$_{ij}$)> 튜플들을 출력한다. 식(4)는 유사도 계산식으로 식(1)을 사용하여 유사도를 구한 후 공통 평가 사용자 수 임계값(γ)과 태그가중치(w_t)를 적용하여 최종 유사도를 계산하는 식이다. $U(i\cap j)$는 아이템 i와 j를 공통 평가한 사용자의 수를 의미한다.

$$S(i,j)=\frac{\max\left|U(i\cap j),\gamma\right|}{\gamma}\cdot sim(i,j)\cdot w_t \tag{4}$$

세 번째 맵리듀스는 이전 단계의 출력을 기반으로 최근접 이웃 아이템을 선정하는 단계이다. 매퍼는 아이템

별로 <$itemID_i$, (($itemID_j$, sim_{ij})> 튜플들을 출력한다. 리듀서는 아이템 별로 유사도 값을 기준으로 정렬하고 유사도 값이 임계값(θ) 이상인 아이템들을 근접이웃으로 선정하고 $itemID_i$에 대한 ($itemID_j$, sim_{ij}) 리스트를 출력한다. 그림 4와 그림 5는 각각 아이템간 유사도를 계산하고 유사도를 기반으로 최근접 이웃을 선정하는 과정을 나타낸다.

```
procedure: Mapper-II
input : tuples rℜ (<userIDs, Listitem>)
output : simCalList
  len = length(Listitem)
  simCalList ← a new list
  for each itemID i=1 to len do
     for each itemID j=i+1 to len do
       t=(itemID_i,itemID_j),(rating_i,rating_j,tagCnt_i,tagCnt_j)
      simCalList.push(t)
     end for
  end for
  emit simCalList

procedure: Reducer-II
input : simCalList
output : tuples sℜ (<(itemID_i,itemID_j),sim_ij>)
  for each itemID i=1 to len do
     compute the similarity of every to items using formula(4)
     key = (itemID_i,itemID_j)
     value = sim_ij
  end for
  emit tuples <key,value>
```

Fig. 4 Compute similarity between items

```
procedure: Mapper-III
input : tuples sℜ (<(itemID_i,itemID_j),sim_ij>)
output : tuples sℜ2 (<itemID_i, (itemID_j, sim_ij)..>)
   for each itemID i=1 to len do
     key=itemID_i
     value=itemID_j,sim_ij
   end for
   emit tuples<key,value>

procedure: Reducer-III
input : tuples sℜ2 (<itemID_i, (itemID_j,sim_ij)..>)
output : tuples nℜ (<itemIDs, neList>)
  θ ← similarity threshold
  neList← new list
  for each itemID i=1 to len do
     for each itemID j ∈ sℜ2
        if sim_ij ≥ θ then
          add(itemID_j, sim_ij) of tuples sℜ2 in templist
        neList=templist
     end for
  end for
  emit <itemID_i, neList>
```

Fig. 5 Selection of similarity-based neighborhood

네 번째 맵리듀스는 사용자가 평가하지 않은 아이템을 추천하기 위해 근접이웃의 아이템 평가값과 유사도를 사용한 예측값을 계산하고 리스트를 출력하는 단계이다. 맵퍼는 첫 번째 리듀서의 결과 파일과 세 번째 리듀서의 결과 파일을 사용하여 userID를 기준으로 (itemID, rating, sim) 리스트를 출력한다. 리듀서는 uerID별로 평가하지 않은 아이템에 대한 예측값을 식 (2)를 사용하여 계산한 후 <userID,list(itemID, P_t)>를 출력한다. 그림 6은 예측값 계산 및 출력과정을 나타낸다.

```
procedure: Mapper-IV
input : tuples nℜ (<itemIDs, neList>), tuples rℜ (<userIDs, Listitem>)
output : tuples pℜ (<userIDs, pcList>)
  pcList ← new list
  for each userID i  ∈ rℜ
      t=(userID i,(itemID_j, rating_ij,sim_ij)
      pcList.push(t)
  end for
  emit <userIDs,pcList>

procedure: Reducer-IV
input : tuples pℜ (<userIDs, pcList>)
output : tuples ℝ (<userIDs, rList>)
   rList ← new list
   P← predictscore of itemIDs
   for each userID i ∈ pℜ
      compute the prediction using formula(2)
      t= (userID_i,(itemID_j,P_ij))
     rList.push(t)
  end for
  emit <userIDs,rList>
```

Fig. 6 Compute predictscore and output the result

마지막으로 생성된 예측값 파일로부터 목표 사용자가 평가하지 않은 아이템 중 예측값이 높은 상위 N개에 해당하는 아이템을 추출하여 사용자에게 추천한다.

Ⅳ. 실험 및 평가

4.1. 실험 환경 및 실험 데이터

실험은 1대의 마스터 노드와 5대의 데이터노드로 구성된 하둡 클러스터 환경에서 실행된다. 마스터 노드는 Intel Core i5-5200 CPU 2.2GHz, 메모리 8GB, 250 Gbyte SSD로 구성되고, 데이터 노드는 Intel Core i5-2400 3.1GHz, 메모리 4GB, 500Gbyte 디스크로 구성된다. 각 노드들은 모두 Ubuntu Linux 14.0을 기반으로

ssh서버, java 1.8.0, Hadoop 2.7.2 버전이 설치되었다.

제안 기법의 성능 평가를 위해 GroupLens에 의해 수집된 MovieLens 데이터 셋을 사용하였다. 데이터 셋은 71,567명의 사용자가 10,681개의 영화를 선호도 1~5의 값으로 평가한 10,000,000개의 평가값과 95,580 개의 태그들을 가진다. 실험에서는 평가 데이터를 80% training data와 test data로 나눈 상태의 데이터 셋을 사용하였다.

4.2. 실험 결과

실험에서는 분산 처리 기법의 효율성을 평가하기 위해서 병렬 시스템의 확장성과 효율성을 평가하는 여러 항목 들 중 가장 많이 사용되는 speedup[16]을 사용하고 추천의 적합성을 평가하기 위해서 F_1-measure 기법을 사용한다.

Speedup은 계산 노드 수의 증가에 따른 효율성의 변화를 나타내는 것으로 동일 데이터를 사용하여 노드 수에 변화를 주었을 때 그 결과가 선형관계를 보일 때 효과적이라 할 수 있다. 식(5)는 speedup 계산식으로 T_1은 단일 노드에서 알고리즘 실행 시간을 나타내고, T_p는 p개의 노드에서의 실행시간을 나타낸다.

$$S_p = \frac{T_1}{T_p} \tag{5}$$

F_1 기법은 추천 성능을 평가하기 위해 사용되며 precision과 recall의 조화평균으로 계산 된다. precision은 추천시스템이 생성한 추천 아이템 목록 중 목표고객이 실제로 선택한 아이템의 비율이고, recall은 목표 고객이 실제로 선택한 아이템 중 추천시스템이 추천한 아이템의 비율이다. precision과 recall은 추천의 개수가 증가하면 recall은 증가하지만 precision은 감소하는 상충관계를 가지며 이를 보완하고자 F_1을 사용한다. F_1의 값이 클수록 추천의 적합성이 좋다고 할 수 있다[17]. 식(6)은 F_1을 계산하는 식이다.

$$F_1 = \frac{2 \times recall \times preision}{recall + preision} \tag{6}$$

본 논문에서는 아이템 크기에 따른 speedup 성능 측정을 위해 데이터 셋의 아이템 수를 2000, 5000, 8000, 10000개로 다르게 구성하고 노드 수를 1~5로 변화를 주어 실험을 진행하였다.

그림 7은 speedup 실험 결과를 나타내는 것으로 아이템 수와 노드 수가 증가함에 따라 speedup이 선형적으로 증가함을 보여준다. 특히,아이템 수가 2000으로 데이터 셋이 작을 때보다 데이터 셋이 커질수록 노드 추가에 따른 속도가 더 빨라져 확장성이 향상되는 것을 알 수 있다.

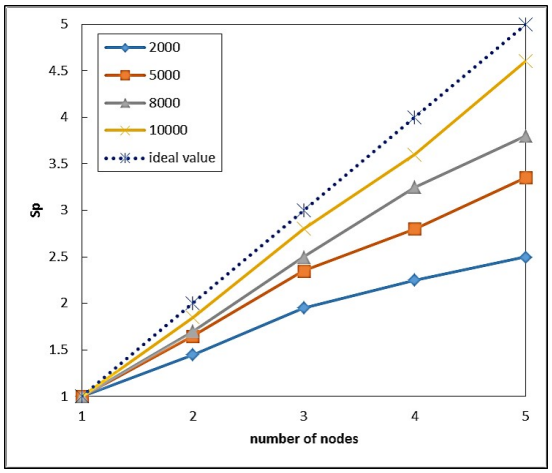

Fig. 7 Result of speedup test

F_1 성능측정을 위해 근접 이웃 수의 변화에 따른 기존 협업필터링 기법들과의 비교 실험으로 진행하였다. 그림 8은 근접 이웃 수 변화에 따른 F_1 실험 결과를 나타낸 것으로 아이템 기반, 사용자 기반 협업필터링 기법과의 비교에서 제안하는 기법의 추천의 적합성이 약 6~22% 향상됨을 보여준다.

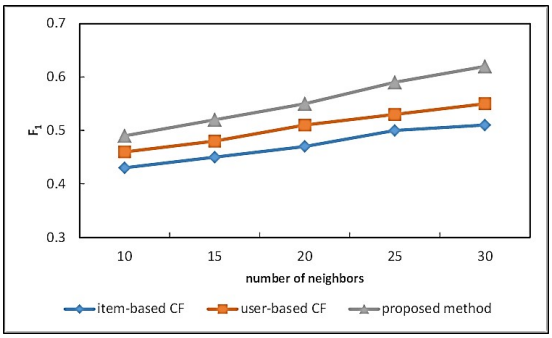

Fig. 8 Result of speedup test

Ⅴ. 결론 및 향후 연구 방향

본 논문에서는 협업 필터링 기반의 추천 시스템의 문제점인 확장성과 추천의 적합성을 향상시키기 위해 빅 데이터 분산 처리 방식과 태그 가중치를 적용한 아이템 기반 추천기법을 제안하였다. 제안 기법에서는 정확한 추천을 위해 아이템에 대한 사용자들의 선호도 변화를 적용하여 데이터를 전처리한 후 4단계의 MapReduce를 거쳐 예측값을 출력하였다. 유사도 계산에서는 태그 가중치를 적용한 계산을 통해 근접이웃을 지정하는 방식을 사용하였다. 제안 기법의 확장성과 적합성 성능을 평가하기 위한 실험에서 제안 기법이 대규모 데이터를 처리 시 확장성이 더 향상되고 기존의 협업 필터링 기법보다 추천의 적합성도 향상된 결과를 보였다.

향후에는 실시간 분석에 더 적합한 방법과 사용자들이 아이템에 남긴 태그를 개수로만 사용하지 않고 그 의미의 연관성을 찾아 아이템 분석에 사용하고 이를 바탕으로 유사아이템을 선정할 수 있도록 연구하고자 한다.

ACKNOWLEDGMENTS

This work was supported by a Research Grant of Pukyong National University(2016 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] M. Papagelisa, D. Plexoosakis, “Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents,” *ACM Transactions on Information Systems, 22*, vol 1. pp.152-166, Sep. 2004.

[4] 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 System*s, vol. 25, no.12, pp.3221-3231, Dec. 2014.

[5] A. Stanescu, S. Nagar, and D. Caragea, “A Hybrid Recommender System : User Profiling from Keywords and Ratings,” *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence(WI) and Intelligenr Agent Technologies(IAT)*, pp. 73-80, Dec.2013.

[6] Z. Zhao, M. Shang, “user-Based Collaborative-Filtering Recommendation Algorithms on Hadoop,” *2010 Third International Conference on Knowledge Discovery and Data Mining*, pp. 478-481, Jan. 2010.

[7] 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.

[8] 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.

[9] F. Lu, L. Hong, and L. Changfeng, “ The Improvement and implementation of distributed item-based collaborative filtering algorithm on Hadoop,” *Proceedings of the 34th Chinese Control Conference*, pp.9078-9083, July 2015.

[10] H. Liang, Y. Xu, Y. Li, R. Nayak, and G. Shaw, “A Hybrid Recommender Systems based on Weighted Tags,” *in 10th SIAM International Conference on Data Mining(SDM* 2010), Apr. 2011.

[11] S. Yun, S. Youn, “A Study on Recommender Technique Applying User Activity and Time Information,” *Journal of Korea Institute of Information and Communication Engineering*, vol. 19, no. 3, pp. 543-551, Mar. 2015.

[12] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, “GroupLens: Applying Collaborative Filtering to Usenet News,” *Communications of the ACM*, vol. 40, no. 3, pp. 77-87, Mar. 1997

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

[14] T. White, *Hadoop: The Definitive Guide*, Sebastopol. CA: O’REILLY, 2010.

[15] K. Han, *Do it: Hadoop with big data*, Seoul: Easys Publishing, 2013.
[16] 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.
[17] S. Ko, "A Recommender Agent using Association Item Trees," *Journal of KIISE:Software and Applications*, vol. 36, no. 4, pp. 298-305, Apr. 2009.

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