분산 협력 필터링에 대한 에이전트 기반 접근 방법
(An Agent-based Approach for Distributed Collaborative Filtering)

김 병 만 † 이 경 †† Adele E. Howe ††† 여 동 규 ††††
(Byeong Man Kim) (Qing Li) (Adele E. Howe) (Dong Gyu Yeo)

요 약 협력 필털링은 그 유용성으로 인해 현재 학문적으로나 상업적으로 널리 사용되고 있지만 확장성 문제, 평가 데이타의 희박성 문제, 초기 평가 문제 등을 안고 있다. 본 논문에서는 이러한 문제들을 일부 해결하기 위해 에이전트 간 협력에 기초한 분산 협력필터링 방법을 제안하였다. 제안 방법에서는 사용자의 평가정보를 에이전트가 지역 데이타베이스에 보관하고 이 정보를 친구들에게만 전파하는 방법을 사용함으로써 사용자 증가에 따른 확장성 문제를 해결하고자 하였다. 그리고 평가 데이타 부족에 따른 추천 질 저하를 줄이기 위해 친구 에이전트의 의견을 반영하는 방법을 사용하였고 새로운 사용자에 대해서도 추천이 가능토록 하기 위해 사용자 프로파일을 이용한 협력필터링 방법을 사용하였다. 실험결과, 본 제안 방법이 확장성뿐만 아니라 데이타 희박성 문제 및 새로운 사용자 문제에도 도움이 됨을 확인할 수 있었다.

키워드 : 정보필터링, 분산협력필터링, 웹에이전트, 친구 네트워크, P2P 네트워크, 확장성 문제, 희박성 문제, 초기 평가 문제

Abstract Due to the usefulness of the collaborative filtering, it has been widely used in both the research and commercial field. However, there are still some challenges for it to be more efficient, especially the scalability problem, the sparsity problem and the cold start problem. In this paper, we address these problems and provide a novel distributed approach based on agents collaboration for the problems. We have tried to solve the scalability problem by making each agent save its users ratings and broadcast them to the users friends so that only friends ratings and his own ratings are kept in an agents local database. To reduce quality degradation of recommendation caused by the lack of rating data, we introduce a method using friends opinions instead of real rating data when they are not available. We also suggest a collaborative filtering algorithm based on user profile to provide new users with recommendation service. Experiments show that our suggested approach is helpful to the new user problem as well as is more scalable than traditional centralized CF filtering systems and alleviate the sparsity problem.

Key words : Information Filtering, Distributed Collaborative Filtering, Web Agent, Friend Network, P2P Network, Scalability Problem, Sparsity Problem, Cold Start Problem

1. 서 론

최근 웹에서 사용 가능한 정보의 양이 기하급수적으로 증가하면서 사용자들이 직면한 문제는 이제 유용한 정보의 부족이 아니라, 개인의 요구에 부합하는 정보를 찾는 일이 되었다. 이러한 문제를 다루기 위해 많은 도구들이 개발되어져 왔는데 그 중의 대표적인 것이 검색 엔진이다. 웹의 초창기에서부터 검색 엔진은 급증하여 왔으나 웹의 엄청난 성장 앞에서는 그 효과가 많이 반감되었다[1].

일반적으로 사용 가능한 검색 엔진에 대한 대안으로는 인터넷 추천 에이전트가 있다. 이들은 사용자들의 취

· 이 논문은 2005년도 한국학술진흥재단의 지원에 의하여 연구되었음 (KRF-2005-013-D00048)
† 종신회원 : 금오공과대학교 컴퓨터공학부 교수
bmkim@kumoh.ac.kr
†† 정 회 원 : 한국정보통신대학교 공학부
liqing@icu.ac.kr
††† 정 회 원 : Colorado State University Computer Science Department professor
howe@cs.colostate.edu
†††† 학생회원 : 금오공과대학교 컴퓨터공학부
sylot@kumoh.ac.kr
논문접수 : 2006년 6월 1일
심사완료 : 2006년 9월 25일

향에 맞는 정보를 찾을 수 있도록 도와주기 위해 사용자 취향에 맞게 자신을 맞추어 나가게 되어 있다. 이러한 에이전트는 일반적으로 사용자 요구를 반영하거나 결과를 여과하는데 필요한 사용자 모델을 학습하여, 사용자의 정보요구를 중재한다. 에이전트들의 적용 분야는 뉴스 필터링, 상거래, 웹 추천, 음악 추천, 영화 추천, 개인 신문 서비스 등 다양한 분야에 걸쳐 있다. 에이전트들은 여러 가지 방법으로 분류할 수 있는데 이 중 하나가 필터링 방법에 따라 분류하는 것이다[2].

정보필터링 방법은 크게 내용기반 방법과 협력 필터링 방법으로 나뉘어 지는데 내용기반 방법은 사용자의 관심사를 표현한 프로파일의 내용과 필터링 대상 항목의 내용을 비교하여 사용자에게 흥미로운 또는 유익한 항목들을 선택하는 방법이다. 이 방법은 텍스트 형태의 항목을 다루는 데 아주 유용한 것으로 알려져 있으며 ACR News, Amalthaea, iInfoFinder, Letizia, NewT, SIFT Netnews, Syskill & Webert, Webmate, WebSail 같은 추천 에이전트들이 이 방법을 사용하고 있다[2]. 그러나 내용기반 필터링은 영화나 음악 등 텍스트 형태가 아닌 항목에 대해서는 자동으로 내용을 분석하기 어려운 관계로 이 방법을 적용하기가 어렵다.

협력 필터링은 타 사용자의 관심사를 예측하는데 동일한 생각을 갖는 사람들의 의견을 이용하는 방법이다. 이 방법은 평가 이력(history) 데이타베이스를 조사하여 대상 사용자와 유사한 관심사를 갖는 사용자들을 찾고 이들이 대상 항목에 대해 어떻게 평가했는지에 대한 정보를 이용하여 대상 항목을 필터링하는 방법이다. Bee-hive, Bellcore Video Recom, GroupLens, Ringo/FireFly, Smart Radio, Tapestry 같은 에이전트들이 이 방법에 바탕을 두어 영화, 음악, 라디오 등을 추천하고 있다[2].

협력 필터링은 내용을 기반으로 하지 않고 단지 항목에 대한 사용자들의 평가를 기반으로 하기 때문에 음악이나 영화 등과 같이 자동으로 내용을 파악하기 힘든 항목에 대해서도 잘 동작하며 프로파일 구성에 신경을 쓸 필요 없이 항목에 대해서 단지 자신의 평가치만 기술해주면 되기 때문에 초보자인 경우에도 별 문제없이 사용할 수 있다. 이러한 특성으로 인해 협력 필터링 방법은 학문적으로나 상업적으로 널리 사용되고 있다. 하지만, 협력 필터링에는 초기 평가 문제, 평가 데이타의 희박성 문제, 확장성 문제 등 자체적으로 해결하기 힘든 문제점들을 안고 있다[2].

협력 필터링이 갖고 있는 문제들은 내용기반 방법에서는 발생하지 않는다. 따라서 최근에 협력 필터링의 문제점들을 해결하여 궁극적으로 좀 더 나은 성능을 얻기 위해 혼합 필터링(Hybrid Filtering) 방법, 즉 내용기반 방법과 협력 필터링 방법을 결합하는 많은 연구들[3-7]이 진행되어 왔다. 본 연구자도 이러한 맥락에서 최근 협력 필터링 방법 내에서 내용기반 방법의 장점을 살릴 수 있는 새로운 방법들[8]을 제안하였고 이를 확률적 모델로 재해석한 방법[9]도 제안하였다. 또한, 이러한 방법을 음악 및 영화 추천시스템에 적용시켜 그 유효성도 입증하였다[10]. 현재 Amazon, Casmir, CDNow, Fab, MovieLens, NewsWeeder, Personal WebWatcher, WebSell, WebWatcher 등 다양한 추천에이전트들이 혼합 필터링 방법을 채택하고 있다[2].

하지만, 본 연구자가 제안한 방법들을 포함한 대부분의 협력 필터링 방법(혼합 필터링 포함)들은 평가 정보들이 서버에 저장되어 있다는 가정 하에서 동작한다. 그리고 실제 협력 필터링을 채택하는 에이전트들도 이 가정에 따라 사용자의 평가정보를 서버로 전송하고 필요할 경우 서버로부터 추천을 받는 구조로 되어 있다. 이렇게 사용자 평가 정보가 모두 한곳에 집중될 경우는 데이타 유지 및 관리에 대한 부담이 커지며 통신량이 서버 측에 몰리는 병목현상이 발생하기 쉽다. 특히, 서버 고장 시 시스템 전체가 동작하지 않는 치명적인 문제점을 갖고 있다.

최근, P2P(Peer-to-Peer)의 장점, 즉, 확장성으로 인해 많은 프로그램들이 P2P 기반으로 개발되고 있는데, 몇몇 연구자들[11-13]이 기존 협력필터링 방법의 문제점을 P2P 기반으로 해결하고자 하고 있다. 본 논문에서는 이러한 맥락에서 새로운 형태의 분산협력필터링 방법을 제안하였다. 제안 방법에서는 사용자의 평가정보를 에이전트가 지역 데이타베이스에 보관하고 이 정보를 친구들에게만 전파하는 방법을 사용함으로써 사용자 증가에 따른 확장성 문제를 해결하고자 하였다. 그리고 평가 데이타 부족에 따른 추천 질 저하를 줄이기 위해 친구 에이전트의 의견을 반영하는 방법을 사용하였고 새로운 사용자에 대해서도 추천이 가능토록 하기 위해 사용자 프로파일을 이용한 협력필터링 방법을 사용하였다.

다음 장에서는 기존의 분산협력필터링 방법을 소개하고 본 제안방법과 이들과의 차이점에 대해 간략히 기술한다. 3장에서 새로운 분산협력필터링을 지원하기 위해 기존의 에이전트 구조를 어떻게 확장했는지를, 4장에서 확장된 에이전트 구조의 핵심부분인 분산협력필터링 방법에 대해서 알아본다. 5장에서는 제안 분산협력필터링 방법의 효용성을 보이기 위한 모의실험 결과를 소개한다.

2. 관련 연구

지금까지 대부분의 협력필터링에 대한 연구는 추천 질 향상에 초점을 맞추어 왔다. 비록 몇몇 논문에서 확

장성 문제를 다루고 있지만 이들 대부분은 모든 평가정보가 서버에 저장되었다는 가정에 기초한다. 최근 들어 확장성 문제에 대한 새로운 접근 방법으로 분산화 방안에 대한 연구들이 진행되고 있는데 대부분 P2P 네트워크에 기초하고 있다. P2P 기반 파일시스템에서는 분산의 대상이 파일이지만 협력필터링 시스템에서는 사용자의 평가 정보가 분산의 대상이 된다.

P2P 시스템에서 자원(예. 파일, 영화, ...)의 위치를 찾는 방법에는 두 가지가 있다. 하나는 Gnutella에서 사용했던 비구조적 방법(unstructured content location)으로 자원을 찾기 위해 모든 피어(peer)들에게 요청을 전파(flooding)하는 방식이다[14]. 이 방법에서는 먼저 피어가 자원을 찾기 위해 그의 모든 이웃에게 요청 메시지를 보내고 이에 대한 응답으로 요청 메시지를 받은 이웃 피어들은 요청한 자원을 찾거나 임계치 거리만큼 도달할 때까지 메시지를 다시 이웃 피어들에게 전달한다. 이러한 접근 방법은 간단하고 피어들이 자주 시스템에 가입하거나 빠져나가는 동적인 환경 하에서도 견고하게 동작한다는 장점이 있다. 하지만, 피어 수가 늘어날 경우 복잡도가 엄청나게 증가하는 문제점, 즉 확장성 문제를 안고 있다.

자원을 찾는 다른 한 방법은 Plaxton 등[15]에 의해 제안된 DHT(Distributed Hash Table)에 기초한 방법이다. 이 방법에서는 피어들이 자원 요청을 처리하기 위해 체계화된 구조(well-defined structure)로 조직화된다. 이 방법에서는 자원의 저장 위치가 DHT에 의해 결정되기 때문에 피어 수가 늘어나더라도 자원을 찾는 시간은 크게 증가하지 않는다. 반면에 동적인 환경 하에서 피어 가입과 탈퇴에 따른 오버헤드가 크다는 문제점이 있다. 지금까지 제안된 대부분의 분산 협력필터링 방법은 위 두 가지 방법 중 하나에 기초하고 있다.

Han과 그 동료들은 PipeCF[11]라는 분산 협력필터링 방법을 제안하였는데 이 방법에서는 평가 데이타 저장 및 예측 작업이 모두 분산적으로 이루어진다. P2P 오버레이 네트워크 상에서 DHT 방식을 이용하여 구현되었기 때문에 구조적 오버레이 네트워크 상에서는 효과적으로 동작한다. 하지만 동적 환경 하에서의 이 방법의 성능은 알려지지 않았다.

Tveit[13]는 모바일 상거래를 위한 P2P 기반 협력필터링 시스템을 제안하였는데 이 방법은 Gnutella 방법을 따르고 있다. 각 피어들은 모바일 고객과의 인터페이스를 담당하는 소프트웨어 에이전트로 동작한다. 각 피어는 자신의 평가정보만 보관하며 추천이 필요한 시점에서 성향이 비슷한 이웃들을 찾기 위해 보관된 평가정보를 이웃 피어들에게 전달한다. 요청을 받은 이웃 피어들은 보내온 평가정보와 자신이 보관하고 있는 평가정보 사이의 유사성을 계산하여 그 값이 임계치 이상이 되면 자신의 평가정보를 목적 피어(target peer : 처음 요청을 발생시킨 피어)에게 전달한다. 임계치보다 낮을 경우는 TTL(Time To Live) 값을 검사하여 0이 아니면 받은 요청을 자신의 이웃들에게 다시 전달한다. 이 방법은 Gnutella처럼 자원요청을 flooding시키기 때문에 확장성 문제를 갖고 있다.

P2P 기반 외의 분산 협력필터링으로 Tivo 텔레비전[16]에서 사용하는 방법이 있다. 이 방법은 수년간 백만 이상의 시청자들에게 적용되어 온 방법으로 항목기반 협력필터링 방법을 사용하고 있다. 협력필터링 방법은 예측 시 항목간의 유사도를 이용하느냐 사용자간 유사도를 이용하느냐에 따라 항목기반과 사용자기반 협력필터링으로 나누어진다. Tivo에서는 PipeCF 방법과는 달리 평가정보를 분산시켜 저장하는 대신 서버에 저장하며 클라이언트-서버 방식으로 동작한다. 대신에 중앙집중식 협력필터링에서 서버가 담당하던 일의 일부를 클라이언트에게 위임하는 방식을 취하고 있다. 비록 항목간 유사도를 계산하는 옵션이 있지만 여전히 서버에서 항목 간 유사도를 계산하고 이를 요청한 클라이언트에게 전달해야 한다. 이는 여전히 서버 측에 확장성 문제를 그대로 놓아두게 된다.

지금까지 분산 협력필터링의 대표적 3가지 방법에 대해 살펴보았다. 이들 방법과 본 제안 방법의 차이점을 간단히 소개하면 다음과 같다. Tivo 방법과의 주요 차이점은 평가 데이타의 분산화 여부에 있으며 PipeCF와는 평가 데이타의 분산화 방법과 자원 접근 방법에 차이가 있다. PipeCF에서는 데이타가 항목 위주로 분산되어 있는 반면 본 방법에서는 사용자 위주로 분산화 되어 있다. 그리고 PipeCF에서는 DHT 기반 자원 접근 방법을 사용하는 반면 본 방법에서는 Gnutella와 유사한 방식을 사용한다. Tveit 방법과 본 제안방법은 여러 면에서 유사하지만 두 가지 면에서 주요한 차이가 있다. 첫째, 피어가 유지하는 평가 데이타에 차이가 있다. Tveit 방법에서는 각 피어가 한 사용자의 평가정보만 유지하지만 본 방법에서는 한 사용자와 그의 친구들에 대한 평가 정보를 유지한다. 둘째, Tviet 방법은 Gnutella 방식을 그대로 사용하지만 본 제안 방법은 확장성 문제를 해결하기 위해 변형된 방식을 사용한다.

3. 시스템 구조

본 논문에서 제안하는 분산 협력필터링은 대표적 P2P 시스템인 Gnutella 방법에 기초한다. 앞장에서 언급한 바와 같이 Gnutella 방법은 사용자들이 자주 P2P 네트워크에 가입.탈퇴를 하는 동적인 환경에서 DHT 보다 나은 성능을 보인다. 하지만 자원 요청을 모든 이웃 피

어들에게 전파하기 때문에 피어 수가 늘어날 경우 오버헤드가 큰 문제점이 있다. 본 제안 시스템에서는 이 문제를 해결하기 위해 자기조직화(self-organizing) 프로토콜을 사용한다. 이 프로토콜에서는 피어들 간의 지름길(shortcut)을 제공하기 위해 친구목록(friend list)을 사용한다. 이에 대한 자세한 내용은 4장을 참고하기 바란다. 또한, 앞에서 언급한 시스템들은 실제 환경에서 발생하는 초기평가(cold-start) 문제에 대한 특별한 대책들이 없다. 그러나, 본 제안구조에서는 사용자 프로파일을 이용한 보완책을 제공한다. 이러한 방법은 초기에 정확한 추천을 제공하여 주지는 못하지만 새로운 사용자에게도 시스템이 어느 정도 잘 동작하도록 한다.

본 제안 시스템 구조는 기존의 개인 에이전트 구조에 분산협력필터링을 이용하여 에이전트 간 추천이 가능하도록 확장하는 방법을 선택하였다. 그림 1에서 보는 바와 같이 본 제안 에이전트의 구조는 두 부분으로 구성된다. 첫 번째 부분은 웹 검색 부분이고 다른 하나는 협력 추천 부분이다. 웹 검색 부분에 대한 연구는 많이 진행되어 있기 때문에 이 부분에 대해서는 기존의 연구내용을 그대로 채택하였다. 본 연구에서는 CSU(Colostae Sate University)에서 제안한 SurfAgent[17] 구조를 사용하였다.

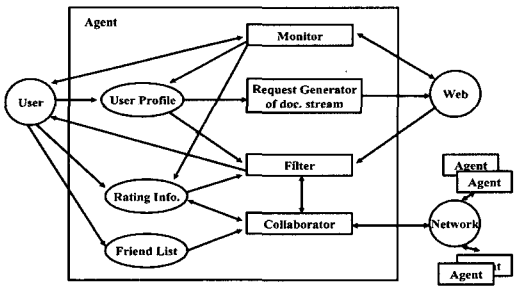


그림 1 시스템 구조

SurfAgent는 사용자의 장기간 관심사를 학습하여 이를 사용자 질의 시 또는 추천 시에 활용한다. 예를 들어, 사용자 철수가 "박중훈 영화"라는 질의를 할 경우, 만약 철수의 프로파일에 액션을 좋아하고 주로 90년대 이후의 영화를 좋아하는 것으로 되어 있으면 질의를 "박중훈 영화 90년 이후 액션" 형태로 질의가 자동 확장되어 웹 검색 엔진에 주어지게 된다. 그리고 검색 결과에 사용자의 피드백을 받을 수 있게 추가한 후 이를 사용자에게 제시한다. 사용자는 검색결과를 보면서 해당 문서에 대한 평가를 내릴 수 있으며 이 때 에이전트는 사용자의 평가에 따라 사용자의 프로파일을 수정하게 된다. 하지만 이러한 형태의 개인 에이전트는 타사용자의 의견을 반영하지 못해서 비슷한 성향을 갖는 사용자의 유용한 정보를 이용치 못하는 경우가 발생한다. 만약, 비슷한 사용자간의 선호도 정보나 항목에 대한 평가정보를 공유할 수 있다면 보다 유용한 정보를 제공할 수 있을 것이다. 예를 들어, 철수와 돌이가 서로 영화에 대한 성향이 비슷하고 돌이는 박중훈 뿐만 아니라 스타일의 비슷한 주명치 영화도 선호한다면 우리는 쉽게 철수도 주명치 영화를 좋아할 것이라고 예측을 할 수 있고 이를 사용자에게 추천할 수 있다. 그림 1에서 빗금친 부분이 바로 이러한 기능을 구현하기 위해 SurfAgent에 추가된 부분이다. 제안된 에이전트 시스템의 간단한 시나리오는 다음과 같다.

i) 각 에이전트는 자신의 사용자에 대한 웹검색에 대한 선호도 프로파일을 유지한다.

ii) 사용자가 원하는 정보를 찾기 위해 질의를 입력하면 시스템은 사용자의 프로파일을 기초로 질의를 재구성하여 서치엔진에 전달한다. 사용자는 검색결과를 보면서 본인이 해당 웹페이지를 좋아하는지 그렇지 않은지를 피드백할 수 있다. 시스템은 사용자의 피드백 정보를 이용하여 사용자 프로파일을 갱신함과 동시에 협력 추천을 위해 사용자의 피드백 정보(평가정보) 자체를 보관한다.

iii) 사용자가 웹페이지에 대해 평가를 내리면 그 정보는 친구목록(friend list)를 통해 친구들에게 전파되고 전달받은 친구는 다시 그 친구들에게 평가정보를 전달한다. 이런 식으로 일정 범위 내의 사용자들에게 평가 정보를 전달한다.

iv) 타 사용자로부터 전달받은 평가정보들을 바탕으로 에이전트는 주기적으로 항목(웹문서, 오디오파일, 비디오파일 등)과 관련된 URL을 사용자에게 추천한다.

장기간 질의 서비스와 질의 수정방법은 [17]에 기술되어 있고 협력추천에 대한 방법은 다음 장에 기술되어 있다. 본 논문에서는 추천 서비스를 제공하기 위해 중앙 서버에 전체 데이타베이스를 관리하는 대신에 자발적 에이전트 환경 하에서 P2P 오버레이에 기초한 분산협력 필터링 방법을 사용하였다.

4. 에이전트기반 분산 협력필터링

중앙 서버가 추천대상 사용자와 유사한 성향을 갖는 이웃들을 찾고 이들이 과거에 내린 평가정보를 이용하여 추천대상에게 추천해 주는 중앙접근 방법과는 달리 본 제안 분산 협력필터링에서는 각 피어들이 업무를 분담하게 된다. 즉, 각 피어들이 자신의 가까운 친구들을 찾고 이들의 정보를 이용하여 추천하게 된다. 이를 위해 각 피어들은 항목 카테고리(예: 영화, 음악, 웹페이지, ...) 별로 사용자 프로파일을 유지한다.

사용자 프로파일은 사용자의 방문 히스토리로부터 묵시적으로 얻거나 아니면 사용자가 명시적으로 선언한 사용자 관심사를 기록한 것으로 여러 개의 선호도 벡터로 표시될 수 있다. 하나의 선호도 벡터는 사용자가 관심을 갖는 하나의 주제를 나타낸다. 본 논문에서는 문제를 단순화시키기 위하여 사용자의 관심사는 하나의 선호도 벡터로 표시된다고 가정한다. 전체 평가데이타의 일부도 사용자 프로파일에 저장된다. 즉 각 피어는 자신의 평가데이타와 누가 자신과 비슷한 성향을 갖고 있는지에 대한 정보를 기록하고 있는 친구목록(friend list)을 사용자 프로파일에 유지한다. 친구목록은 친구의 ID와 친구의 평가데이타로 구성되며 이 목록은 4.1절의 방법에 따라 생성 및 갱신된다.

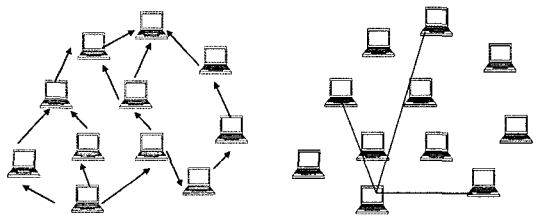
각 피어들이 자신의 평가데이타와 성향이 비슷한 피어들의 평가데이타를 지역데이타베이스에 유지함으로써 동적인 P2P 환경 하에서 다음과 같은 이점을 얻을 수 있다.

1) P2P 환경은 동적이어서 구성 멤버들이 자주 변하게 된다. 이러한 환경 하에서도 현재 P2P 네트워크에 가담하지 않는 친구일지라도 그의 평가데이타를 사용할 수 있다. 왜냐하면 한 피어의 평가 데이타는 해당 피어의 지역데이타베이스뿐만 아니라 친구들의 지역데이타베이스들에도 복사가 되어 있기 때문이다. Pitsilis의 방법[12]에서는 친구의 평가 데이타를 유지하지 않기 때문에 현재 P2P 네트워크에 가담하지 않는 피어들의 평가정보를 사용할 수 없다.
2) 본 방법은 추천 시에 추가의 네트워크 트래픽을 야기하지 않는다. 대신에 평가데이타를 친구에게 전파할 경우와 친구목록을 갱신할 경우 네트워크 트래픽을 야기한다. 하지만, 이 두 경우 친구목록을 바탕으로 이루어지기 때문에 P2P 네트워크 전체에 대해서 전파시킬 필요가 없다. 즉, 친구목록이 P2P 오버레이 네트워크 상에 생성된 지름길 역할을 담당한다.

4.1 친구목록 관리

Gnutella에서는 그림 2(a)에서 보는 바와 같이 아주 단순한 라우팅 알고리즘을 사용한다. 즉, 한 피어에 의해 만들어진 자원 탐색 질의는 이웃 피어들에게 전파가 된다. 질의를 전파 받은 피어는 질의에 맞는 자원을 갖고 있으면 더 이상 질의를 전파시키지 않고 자원 탐색 질의를 발생시킨 피어에게 전달한다. 만약, 해당 자원을 찾지 못했을 경우는 TTL 값을 하나 감소시킨 후 이웃 피어들에게 질의를 다시 전파한다. TTL 값이 0이면 더 이상 질의를 전파하지 않는다.

본 시스템의 추천 질은 친구목록을 유지하는 방법에 상당히 의존한다. 성향이 비슷한 피어들을 친구들로 많이 유지할수록 추천 질은 좋아진다. 본 시스템에서는 피어들이 처음에 친구들을 찾기 위해 Gnutella에서 사용하는 방법과 유사한 방법을 사용한다. 즉, 요청(친구 찾기)이 이웃 피어들에게 전파된다. 요청에는 요청발생 피어의 선호도 벡터와 전파범위를 한정짓는 TTL값이 포함된다. 요청을 받은 피어는 자신의 선호도 벡터와 요청 속에 포함된 선호도 벡터를 비교하여 유사도가 임계치 이상이면 자신의 ID와 평가정보를 요청발생 피어에게 전달한다. 만약 임계치 이하이면 TTL값을 1 감소시킨 후 이웃 피어들에게 전파한다. TTL 값이 0이면 더 이상 전파하지 않는다. 요청발생 피어는 자신이 발생시킨 요청에 대해 여러 피어들로부터 응답을 받게 된다. 이 경우 응답에 포함된 평가정보를 바탕으로 사용자간 유사도를 계산하고 이를 바탕으로 상위 n개의 피어들을 선택하여 친구로 유지한다.



(a) Gnutella (b) Located by friend list

그림 2 라우팅 방법

각 피어들의 친구목록이 구축이 되면 P2P 네트워크 위에 새로운 계층, 즉 지름길층(shortcut layer)가 형성되는데 이를 본 논문에서는 "친구네트워크"라 한다. 피어는 평가 정보를 계속적으로 생성하고 이를 이웃에게 전파하는데 이 때 P2P 네트워크 상의 이웃이 아니라 그림 2(b)와 같이 친구네트워크 상의 이웃, 즉 친구에게 전파한다. 또한 피어는 친구로부터 받은 평가정보를 자신의 지역데이타베이스에 저장하고 차후 예측에 사용한다.

세월이 흐르면 어제의 친구가 오늘의 친구가 되지 않을 수 있다. 마찬가지로 현재 구축된 친구목록이 최근의 친구관계를 제대로 반영하지 못할 수 있다. 제대로 친구관계를 반영하여야만 제대로된 추천을 할 수 있다. 이를 위해 본 논문에서는 각 피어들이 주기적으로 친구관계를 갱신하도록 하고 있다. 이를 위해서 초창기에 친구목록을 생성하듯이 각 피어들이 이웃 피어들에게 요청을 발생시키고 이에 따라 유사한 성향을 갖는 피어들이 응답하는 형태로 동작시킬 수 있다. 하지만, 이 경우 네트워크 트래픽이 증가하는 문제가 있고 친구관계가 갑작스럽게 크게 변하지 않기 때문에 본 논문에서는 새로운 친구를 찾기 위해 친구목록을 이용하는 방법을 사용하였다. 각 피어는 친구들에게 새로운 친구를 추천해 달라고 요청함으로써 친구 탐색 범위를 축소할 수 있다. 하

지만 친구에게만 요청할 경우는 성향이 아주 비슷한 새로운 친구를 찾지 못하는 경우가 발생할 수 있다. 이러한 문제점을 완화시키기 위하여 본 논문에서는 FDTL (Friendship Depth To Live)이라는 패러미터를 도입하였다. 이 패러미터는 P2P 네트워크이 아니라 친구네트워크 상에 정의되었다는 점을 제외하면 TTL과 동일하다.

새로운 친구를 찾기 위해 보내지는 요청에는 요청발생 피어의 평가정보가 포함된다. 이 평가정보를 이용하여 친구가 될 정도(간단히 친구정도)를 계산하게 된다. 계산 방법은 4.2절을 참고하기 바란다. 초기, 즉 친구목록 구성 시에는 평가정보가 없기 때문에 사용자 선호도 벡터가 이 용도로 사용됐었다는 점을 기억하기 바란다. 어떤 피어의 사용자들은 한참 후에도 아무런 평가 정보를 제공하지 않을 수 있다. 이 경우도 초기와 마찬가지이기 때문에 평가정보 대신에 사용자 선호도벡터가 요청메시지에 포함되어 전달된다.

친구로부터 요청메시지를 받은 피어는 요청에 포함된 평가정보와 지역데이타베이스에 있는 친구의 평가정보를 이용하여 친구정도를 계산하고 이 값이 임계치 이상이면 친구 ID와 친구의 평가치를 요청발생 피어에게 전달하게 된다. 이처럼 친구에 의해 추천된 새로운 친구를 본 논문에서는 "잠정적 친구(potential friend)"라 부른다. 요청발생 피어는 친구 또는 친구의 친구(친구의 친구의 친구, ... 등을 포함)들로부터 응답들을 받은 후 현재의 친구와 추천 받은 잠정적 친구들 중에서 성향이 유사한 상위 n 개를 선택하여 새로운 친구목록을 구성한다. 이러한 과정이 동적인 환경 하에서 점진적으로 친구 선택을 개선케 한다.

아래의 시나리오는 친구목록이 구성된 후 새로운 친구 발견과 선택에 대한 예를 보여준다. 각 피어는 N명의 친구를 친구목록에 유지한다고 가정한다.

- 사용자 A가 자신의 선호도를 포함하는 질의 패키지를 친구목록에 있는 친구들에게 전송한다. 이 때 전송 패키지가 전달될 단계를 제한하기 위해 FDTL 값이 패키지에 같이 전송된다. 그림3에서 피어 A의 FDTL 값을 3으로 하면 모든 피어들을 포함할 수 있다.
- 일단 사용자 A의 친구 C, D, E, K, Q, P가 A로부터 질의 패키지를 받으면 이들은 자신의 지역데이타베이스에 있는 친구들의 평가정보와 패키지 내에 포함된 A의 선호도, 즉 평가정보를 비교하여 유사도가 임계치 이상이 되는 친구들을 잠정적 친구로 추천한다. 이 때 잠정적 친구의 ID와 평가정보를 같이 보낸다. 동시에 질의 패키지를 그들의 친구들에게 전달한다.

  사용자 A는 질의 패키지에 대한 응답을 수합하여 자신의 친구목록을 갱신한다.

4.2 친구관계 계산

본 논문에서는 항목에 대한 선호성향이 비슷한 사용자를 친구로 취급한다. 따라서 친구목록을 구축하기 위해서는 사용자간의 항목 선호성향 유사관계를 계산할 필요가 있다. 이 목적으로 본 논문에서는 두 가지의 척도를 사용한다. 그 중에 하나는 정보검색이나 문서 마이닝[18] 분야에서 많이 사용되는 코사인유사도이다. 이는 사용자의 평가정보가 유용하지 않을 경우 사용자의 선호도벡터를 이용하여 사용자간의 항목 선호 성향 유사관계를 계산할 때 사용된다. 사용자 k와 l 사이의 코사인유사도는 아래와 같이 정의된다.

$$\mathrm{sim}(k,l)=\mathrm{sim}(D_1,D_2)=\frac{\sum_{i=1}^{m}x_{1i}x_{2i}}{\sqrt{\sum_{j=1}^{m}x_{1j}^2}\sqrt{\sum_{j=1}^{m}x_{2j}^2}} \quad (1)$$

여기서, D_1과 D_2는 사용자 k와 l의 선호도벡터이며, x_{1i}와 x_{2i}는 각각 D_1과 D_2의 i 번째 원소를 의미한다.

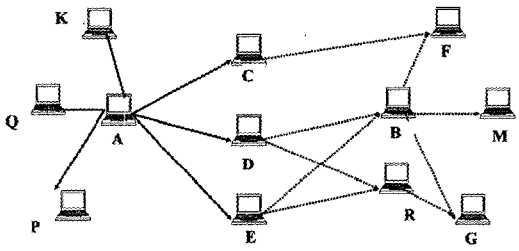


그림 3 친구 네트워크의 예

다른 한 척도는 두 평가 히스토리 사이의 선형관계를 계산하는 피어슨 상관계수[18]이다. 본 논문에서는 주체 피어와 공통된 평가항목이 많은 피어들에게 추가점을 주기 위해 약간 변형된 방법을 사용하였다.

$$\mathrm{sim}(k,l)=\frac{\max(|V_k\cap V_l|,\beta)}{\beta}\times\frac{\sum_{i=1}^{m}(R_{k,i}-\overline{R_k})(R_{l,i}-\overline{R_l})}{\sqrt{\sum_{i=1}^{m}(R_{k,i}-\overline{R_k})^2}\sqrt{\sum_{i=1}^{m}(R_{l,i}-\overline{R_l})^2}} \quad (2)$$

여기서, β는 임계치를, $|V_k\cap V_l|$는 사용자 k와 l이 공통으로 평가한 항목 수를, $sim(k, l)$은 사용자 k와 l 사이의 유사도를 나타내며, m은 사용자 k와 l 모두가 평가한 항목 수를, $R_{k,i}$와 $R_{l,i}$은 사용자 k와 l이 항목 i에 내린 평가치를, $\overline{R_k}$와 $\overline{R_l}$는 사용자 k와 l의 평균 평가치를 의미한다.

4.3 예측 방법

사용자에게 추천서비스를 제공하기 위해서 에이전트는 자신의 지역데이타베이스에 있는 평가정보를 이용하여 사용자가 특정 항목을 좋아할 정도를 예측하여야 한다. 사용자가 항목을 좋아할 정도를 예측하는 방법에는

두 종류가 있다. 하나는 사용자간 유사도에 바탕을 둔 방법이고 다른 하나는 항목간 유사도에 바탕을 둔 방법이다. 첫 번째 것이 사용자기반 협력필터링 방법에 사용되는 것이고 두 번째 것이 항목기반 협력필터링 방법에 사용되는 것이다. [18]의 연구에 의하면 항목기반 협력필터링 방법이 사용자기반 협력필터링 방법보다 추천 질 측면이나 계산 복잡도 측면에서 유리한 것으로 알려져 있다. 하지만 본 논문에서는 예측방법으로 사용자기반의 방법을 사용하였다. 왜냐하면 에이전트(또는 피어)의 지역데이타베이스에는 특정 사용자와 그의 친구들에 대한 평가정보만 기록되어 있기 때문에, 즉 항목 입장에서 보면 부분적인 평가정보만 저장되어 있어서 항목간 유사도를 정확하게 계산할 수 없기 때문이다.

예측은 이웃 사용자들의 평균평가치에 대한 편차들에 대해 가중치평균을 함으로써 이루어진다. 이웃 사용자는 사용자간 유사도를 기반으로 상위 n명의 사용자가 선택된다. 일반적인 예측 식[19]은 아래와 같다.

$$P_{k,i} = \overline{R_k} + \frac{\sum_{l=1}^{n}(R_{l,i} - \overline{R_l}) \times \mathrm{sim}(k,l)}{\sum_{l=1}^{n}|\mathrm{sim}(k,l)|} \quad (3)$$

여기서, $P_{k,i}$는 항목 i에 대한 사용자 k에 대한 예측을 표시한다. n은 사용자 k의 최인접 이웃의 수, $R_{l,i}$는 항목 i에 대한 사용자 l의 평가치, $\overline{R_l}$는 사용자 l의 평균 평가치, $\overline{R_k}$는 사용자 k의 평균 평가치, sim은 사용자 k와 이웃 사용자 l 사이의 유사도로 식 (2)를 의미한다.

고전적인 협력필터링 시스템에서는 새로운 사용자에 대해서는 평가정보가 없는 관계로 위의 예측 식을 적용할 수 없고 이로 인해 추천 서비스를 제공할 수 없다. 하지만 본 시스템에서는 새로운 사용자인 경우에도 위의 식을 사용하여 예측할 수 있다. 대신에, 식 (3)에서의 sim을 식 (2) 대신에 식 (1)로 대치를 하면 된다. 본 시스템에서는 모든 사용자에 대해서 선호도벡터가 존재한다고 가정하기 때문에 새로운 사용자에 대해서도 식 (1)을 사용하여 사용자간 유사도를 계산할 수 있다. 식 (3)을 새로운 사용자에게 적용할 경우 추가로 고려해야 할 부분이 $\overline{R_k}$이다. 새로운 사용자인 경우는 평가 데이타가 없는 관계로 이 값이 0으로 취급이 된다. $\overline{R_k}$는 사용자의 평균 평가치를 의미하는데 새로운 사용자에 대해서는 평가 데이타가 없다는 이유로 0으로 해석한다는 것은 합당치 않다. 따라서, 본 시스템에서는 0 대신에 그의 친구들의 평균 평가치로 해석하였다.

4.4 확장성 및 희박성 문제

많은 응용에서의 성공에도 불구하고 협력필터링 방법은 심각한 두 가지 문제, 즉 확장성 문제와 희박성 문제를 갖고 있다[20]. 최근 협력필터링을 채택하는 시스템들은 증가하는 사용자와 정보의 양으로 인해 수백만의 사용자와 항목들을 다룰 필요가 있다. 협력필터링 알고리즘은 메모리기반 방법과 모델기반 방법으로 나눌 수 있다[21]. 모델기반 방법은 사전에 평가데이타에 대한 모델을 구축하고 이 구축된 모델을 바탕으로 온라인으로 예측하는 방법이다. 따라서, 이러한 접근 방법은 사용자 수가 증가하거나 평가 데이타의 양이 많더라도 온라인 예측시간에는 큰 영향을 주지 않는다. 하지만, 메모리기반인 경우는 온라인으로 데이타베이스를 검색하여 유사한 사용자들을 찾고 이들의 내린 평가치를 종합하여 예측하기 때문에 사용자 수나 항목의 수가 수백만에 달하게 되면 예측 시간이 엄청나게 지연되어 실용성이 떨어진다. 즉, 메모리기반 방법은 확장성 문제를 갖고 있다.

확장성 측면에서 모델기반 방법이 유리함에도 불구하고 본 시스템에서는 4.3에서 기술한 바와 같이 메모리기반 방법을 채택하였다. 다음과 같은 두 가지 이유에서 이러한 결정을 하게 되었다. 첫째, 각 에이전트의 지역데이타베이스에는 부분적인 평가데이타만 있기 때문에 정확한 모델을 구축하기가 어렵다. 둘째 각 에이전트의 지역데이타베이스에는 특정 사용자와 그 친구들에 대한 평가정보만 유지되기 때문에 확장성 문제가 전통적인 협력필터링 방법에 비해 심각하지 않다.

희박성 문제는 많은 사용자들은 극히 일부 항목에 대해서만 평가를 하기 때문에 비록 새로운 사용자가 아니더라도 유사한 성향을 갖는 이웃사용자를 찾기가 곤란한 경우가 자주 발생한다. 이러한 문제를 희박성 문제라 하며 협력필터링의 적용성과 추천 질을 떨어뜨리는 주요한 요인으로 작용한다. 이러한 문제를 완화시키기 위하여 여러 방법[21-23]들이 제안되었는데 이러한 방법 모두 모든 평가데이타가 서버에 저장되어 있는 가정 하에 동작한다. 본 시스템에서는 에이전트의 지역데이타베이스에 친구들에 대한 평가정보만 유지되기 때문에 이러한 방법들을 그대로 적용하기가 어렵다. 그래서 본 시스템에서는 친구들의 의견, 즉 실제 평가 데이타가 아니라 친구들이 자신의 지역데이타베이스를 이용하여 예측한 값을 이용하는 방법을 사용하였다. 예를 들어, 사용자 *U1*의 지역데이타베이스에 그림 4와 같은 평가정보가 저장되어 있고 에이전트가 사용자 *U1*이 항목 *I3*를 좋아할 정도를 예측하고자 한다고 가정하자. 지역데이타베이스에 저장된 다른 어떤 사용자도 사용자 *U1*과 공통으로 평가한 항목이 존재하지 않기 때문에 순수한 협력필터링은 어떠한 예측도 할 수 없다. 그러나, 본 제안방법에서는 친구 *U2, U3, U4*에게 항목 *I2, I4, I7*를 좋아할 정도를 예측해 달라고 요청하고 요청 받은 친구들은 자신들의 지

역데이타베이스에 있는 평가정보를 이용하여 본인들이 요청 받은 항목을 좋아할 정도를 4.3절의 수식을 이용하여 계산하고 이 값을 요청자에게 전달한다. 요청자는 그의 친구들로부터 예측치들을 받은 후 그 값을 이용하여 본인이 *I3*를 좋아할 정도를 계산한다.

	I1	I2	I3	I4	I5	I6	I7	I8
U1		3		5			4	
U2	1		3			5		
U3			4					5
U4	2				3			

그림 4 피어 A의 평가정보

본 방법에서는 부분 평가데이타만 에이전트의 지역데이타베이스에 유지하기 때문에 중앙집중식 방법에서는 발생하지 않는 특수한 상황을 고려해야만 한다. 예를 들어, 그림 5에서 보는 바와 같이 사용자 *U1*의 친구 누구도 항목 *I3*에 대한 평가데이타를 갖고 있지 않다. 그래서 항목 *I3*가 실제로는 새로운 항목이 아님에도 불구하고(사용자 *U5, U6, U7, U8*이 항목 *I3*에 대한 평가치를 가고 있음) 사용자 *U1*의 입장에서는 마치 새로운 항목처럼 취급이 된다. 순수 협력필터링 방법에서는 새 항목에 대해서도 평가데이타의 부재로 제대로 된 추천을 할 수 없다. 하지만 본 방법에서는 이러한 상황에서도 앞의 경우와 마찬가지로 친구의 의견을 물어 사용자 *U1*이 항목 *I3*를 좋아할 정도를 예측할 수 있다. 즉, 그림 5에서 사용자 *U1*의 친구 *U2, U3, U4*는 각각 자신들의 지역데이타베이스에 있는 평가데이타를 이용하여 본인들이 항목 *I3*를 좋아할 정도를 예측할 수 있고 사용자 *U1*는 이들 예측치를 이용하여 본인이 *I3*를 좋아할 정도를 예측할 수 있다.

친구에게 의견을 물을 시, 친구 역시 평가 데이타 부족으로 의견을 제시하지 못하는 경우가 발생할 수 있다. 이러한 경우 요청받은 친구는 자신의 친구들에게 의견을 물어 그 의견을 바탕으로 자신의 의견을 친구에게 제시할 수 있으나 이러한 방법은 과다한 통신을 야기시킬 수 있다. 그래서 본 논문에서는 자신의 지역데이타베이스를 기초로 해서 의견을 제시하지 못할 경우에는 친구의 의견을 다시 구하지 않고 의견 없음으로 답하는 방식을 택하였다. 이러한 방법은 구현이 간단하고 통신량을 줄일 순 있지만 친구목록에 없다는 이유로 유용한 피어의 의견을 활용하지 못할 수 있다. 다행히 이러한 문제점은 주기적인 친구 목록 갱신을 통하여 어느 정도 해결할 수 있다.

5. 성능 평가

본 제안 방법의 성능을 평가하기 위해 DERC(Digital Equipment Research Center)에서 수집한 Each-Movie 데이타를 이용하여 다양한 실험을 하였다. Each-Movie 데이타에는 1,623 개의 영화 항목이 있으며 61,265 명의 사용자가 평가한 2,800,000 개 이상의 평가 데이타가 있다. 평가는 사용자가 직접 입력한 것이며 정수형이다. Each-Movie 데이타에 있는 평가치들은 사용자들이 명시적으로 입력한 값이며 0과 5사이의 값을 갖는다. 이 데이타는 본 연구진이 알고 있는 바로는 협력필터링을 위한 최대 규모의 공개 테스팅 데이타이다. 실험의 속도를 위하여 전체 평가데이타를 이용하지 않고 부분 평가데이타를 이용하였다.

5.1 평가 척도

예측 질에 대한 평가 척도로는 MAE(Mean Absolute Error)를 사용하였다. MAE는 실제 사용자 평가에 대하여 예측치를 비교함으로써 추천 시스템의 정확도를 평가하는 방법으로 현재 널리 사용되고 있다. MAE는 모든 테스트 대상에 대해서 평가치와 예측치 간의 오류를 구하고 이 오류의 절대값을 합한 후 테스트 대상의 수로 나눠줌으로써 얻을 수 있다. MAE가 낮을수록 예측의 정확도는 높아지게 된다.

$$MAE = \frac{\sum_{i=1}^{n} |p_i - q_i|}{n}$$

여기서, n은 평가 대상의 수를, p_i는 대상 i에 대한 예측치를, q_i는 대상 i에 대한 실 평가치를 나타낸다.

User U1

	I1	I2	I3	I4
U1		3		3
U2	5			4
U3	4	2		4
U4		2		3

User U2

	I1	I2	I3	I4
U1		3		3
U2	5			4
U5	4	4	3	4
U6	3	5	4	5

User U3

	I1	I2	I3	I4
U1		3		3
U3	4	2		4
U7	3	5	2	1
U8	5	2	4	5

그림 5 친구의 친구의 정보를 이용한 추천

5.2 친구목록에 유지하는 친구의 수

Breese[20]가 지적한 바와 같이 추천 성능은 예측에 관여하는 이웃의 수에 의존한다. 마찬가지로 본 제안방법의 추천성능도 예측에 관여하는 친구의 수에 많은 영향을 받는다. 친구의 수가 적을 경우는 추천에 이용할 평가데이타가 부족하고 반대로 친구의 수가 너무 많을 경우는 추천 시에 잡음을 발생시켜 추천 질을 떨어뜨리게 된다. 본 저자는 이전에 평가데이타가 서버에 모두 저장된 상황에서 내용기반 필터링 방법을 협력필터링 방법 내에서 흡수/통합한 방법[8]을 제안하고 이의 성능을 Each-Movie데이타를 사용하여 평가하였다. 이 때 성향의 비슷한 이웃의 수가 30일 경우 성능이 좋았다. 따라서, 본 실험에서는 친구의 수에 따른 성능 평가를 하지 않고 대신에 친구의 수를 30으로 고정하여 이 후의 다른 실험들을 수행하였다.

5.3 모의실험

실험 환경을 구축하기 위해 먼저 Each-Movie데이타에서 최소한 20개의 항목에 대해서 평가한 1,000명의 사용자를 선택하고 그들 간의 유사도를 계산한 후 각 사용자의 가장 유사한 사용자 30명을 파악하여 기록하였다. 그리고 나서 시스템의 초기 상태를 시뮬레이션하기 위해 각 사용자의 친구들을 0~10명 무작위로 선택하였다. 시스템 초기 상태에서는 성향이 비슷한 사용자가 친구로 선택될 가능성이 상대적으로 줄어들기 때문에 이렇게 무작위로 선택하였으며 또한 제대로 친구를 파악하기 힘들기 때문에 친구의 수를 30으로 하지 않고 0에서 10명 사이로 선택하였다.

먼저 100명의 사용자를 선택하여 그들에게 그들 친구들에게 새로운 친구를 찾아 달라는 요청을 보내도록 했다. 요청을 받은 친구들은 자신의 지역데이타베이스에 있는 평가정보를 이용하여 친구를 추천함으로써 선택된 사용자들이 친구 30명을 찾도록 해준다. 각 런(run)마다 이렇게 새롭게 구축된 친구목록을 이용하여 사용자가 항목들을 좋아할 정도를 예측하였다. 그림 6에서 보는 바와 같이 예측 질이 초기에 급격히 증가하다가 25런 후에는 거의 증가하지 않는 것을 알 수 있다. 친구네트워크가 정적이면, 즉 네트워크에 새로이 가입된 친구나 탈퇴한 친구가 없으면 일정 시간이 지난 후 더 이상 친구목록에 아무런 변화가 없게 된다. 이러한 상태를 "포화(saturated)" 상태라 하며 친구네트워크가 최적이 상태에 도달한 것이다.

중앙집중형 CF 방법과의 성능비교를 위하여 초기에 저장해둔 30명의 친구들을 이용하여 예측한 결과와 앞의 실험에서 친구네트워크가 포화된 상태에서 예측한 결과를 비교하여 보았다. 그림 7에서 보는 바와 같이 예측 정확도(막대 그래프) 면에서는 거의 차이가 없는 반면 커버리지(coverage) 면에서는 본 방법이 중앙집중형 CF 방법에 비해 월등히 좋음을 알 수 있다. 커버리지는 테스트 데이타의 항목 평가치 중 몇 개나 예측이 가능한가를 나타내는 것으로 예측에 사용되는 평가 데이타가 희박할 경우 이 값이 일반적으로 떨어진다. 이처럼 커버리지가 좋은 이유는 4.4 절에 기술한 희박성 문제 해결 방법에 기인한 것으로 보인다.

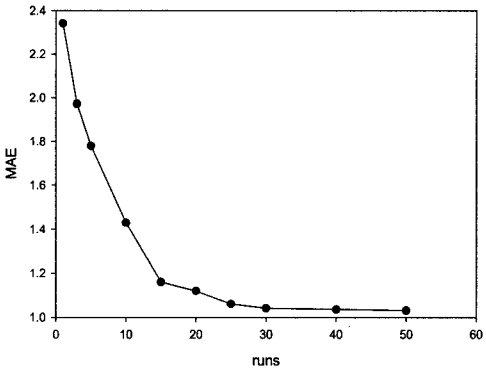


그림 6 런에 따른 성능평가

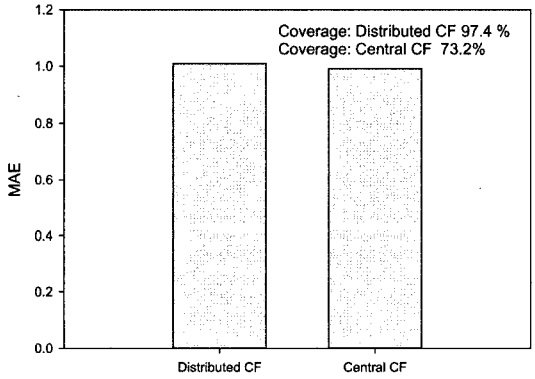


그림 7 분산 CF와 중앙집중식 CF와의 성능 비교

FDTL 값도 추천 성능에 직접적인 영향을 미친다. FDTL에 제한이 없으면 첫 번째 런에서 거의 모든 피어에게 요청을 전파시키게 된다. 이 경우, FDTL은 TTL과 유사하게 동작하여 네트워크 트래픽을 과도하게 발생시켜 러시아워를 야기시킬 수 있다. 반대로 FDTL이 너무 작으면 유사한 성향의 친구를 찾는데 제한이 가해진다. 그림 8에서 보듯이 FDTL이 4인 경우 만족할 만한 성능을 보임을 알 수 있다.

동적인 환경을 시뮬레이션하기 위해, 먼저 1000명의 사용자의 평가데이타의 80%만 이용하여 네트워크 포화 상태를 만든 후 나머지 평가데이타를 추가한 경우의 성능을 테스트하여 보았다. 사용자의 평가정보는 시간이 지나면서 계속 변경되게 된다. 이 실험은 이러한 경우

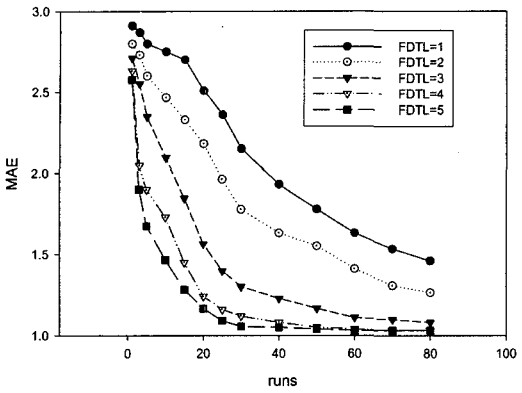


그림 8 FDTL에 따른 성능평가

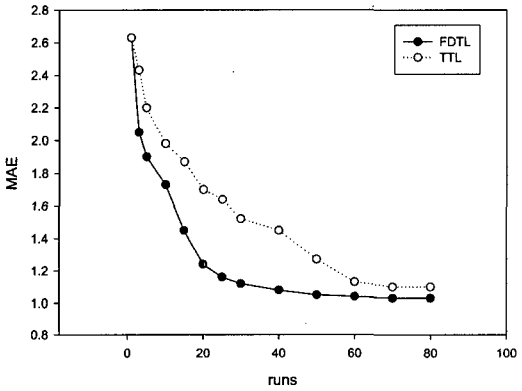


그림 9 FDTL과 TTL에 따른 성능 비교

시스템이 얼마나 빨리 변경된 환경에 적응하는지를 보기 위함이다. 그림 9의 FDTL로 표시된 그래프가 그 결과를 보여 주는데 본 제안 방법이 사용자 평가정보의 변경에 신속하게 반응함을 알 수 있다.

친구목록 갱신 방법의 유용성을 보이기 위해 P2P 네트워크 상에서 TTL를 이용한 경우와 친구네트워크 상에서 FDTL를 이용한 경우의 성능을 비교하였다. 각 피어가 1~50 사이의 이웃 (neighbor)을 무작위로 선택하게 함으로써 P2P 네트워크를 구성하였고 TTL 값은 FDTL 값과 같은 4를 사용하였다. 정확한 모의실험을 위해서는 P2P 네트워크 구성 시 좀 더 정교한 방법을 사용하여야 하나 이는 본 연구의 범위를 벗어나며 FDTL의 유용성을 보이기에는 간단한 방법도 충분하다고 판단하여 이와 같은 방법을 사용하였다. 그림 9에서 보듯이 FDTL을 사용할 경우가 TTL을 사용할 경우 보다 성능이 좋음을 알 수 있다.

고전적인 협력필터링 방법에서는 새로운 사용자에 대해서는 아무런 평가정보가 없기 때문에 성향이 비슷한 이웃들을 찾을 수가 없다. 하지만, 본 방법에서는 사용

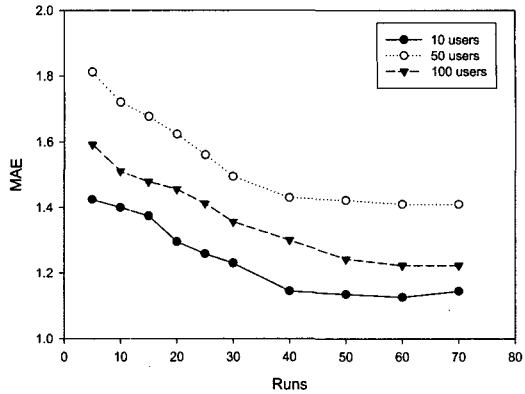


그림 10 cold start 문제에 대한 성능

자마다 선호도벡터가 유지되기 때문에 새로운 사용자의 친구들도 이 정보를 이용하여 찾을 수가 있고 이로 인해 추천도 가능해지게 된다. 새로운 사용자에 대한 본 방법의 성능을 테스트하기 위해서는 먼저 객관적인 방법으로 구성된 사용자 선호도벡터가 필요하다. 본 실험에서는 [24]에서 사용했던 방법, 즉 사용자 평가정보로부터 역으로 사용자 선호도벡터를 구성하는 방법을 적용하였다. 그림 10에서 보는 바와 같이 사용자 선호도벡터 덕분에 사용자는 계속적으로 친구목록을 갱신할 수 있고 이로 인해 예측 질을 개선할 수 있음을 알 수 있다. 그림 10에서 1000명의 사용자 중에서 무작위로 10, 50, 100명을 선택하여 이들의 평가정보를 없앰으로써 마치 새로운 사용자인 것처럼 만들어 실험하였고 이러한 과정을 10번 반복 수행하여 그들의 평균치를 취하였다.

6. 결 론

본 논문에서는 개인 에이전트에 협력 추천기능을 지원하기 위한 분산 접근방법, 즉 에이전트가 자기조직화 프로토콜에 따라 사용자의 평가정보를 친구들에게 전파시킴으로써 협력 추천이 가능해지도록 하는 방법을 제안하였다. 제안 방법은 지역데이타베이스에 자신의 평가데이타와 친구들로부터 전파 받은 평가데이타만 유지하기 때문에 기존의 중앙집중형 접근방법보다 확장성이 좋다. 다행히도, 실험 결과, 일부의 평가데이타만 유지함에도 불구하고 본 방법의 예측 질이 중앙집중형 방법의 것과 별 차이가 없음을 알 수 있었다. 또한, 본 방법이 중앙집중형 방법에 비해 커버리지 면에서 월등히 좋음도 확인할 수 있었다. 이는 평가데이타가 충분치 않을 경우 친구의 의견을 사용하는 기능에 기인한 것으로 보인다.

사용자의 평가데이타는 개인의 프라이버시와 관련된 데이타이지만 본 방법에서는 악의적 사용자에게 이러한

정보가 그대로 노출이 될 수 있다. 앞으로 이 부분에 대한 추가의 연구가 필요하다. 또한, 본 방법은 사용자 수에 대해서는 별 문제가 되지 않지만 항목의 수에 대해서는 확장성이 부족하다. 보다 실용적이 되기 위해서는 이 부분에 대한 처리가 필요하다.

참 고 문 헌

[1] Lawrence S. and Giles, CL, Accessibility of Information on the Web, *Nature*, Vol. 400, 1999.

[2] Montaner M., Lopez, B., de la Rosa, J.Ll, A Taxonomy of Recommender Agents on the Internet, *Artificial Intelligence Review*, Vol. 19, 2003.

[3] Claypool, M., Gokhale, A., Mirana, T., Murnikv, P., Netes D. and Sartin, M., Combing Content-Based and Collaborative Filters in an Online Newspaper, In *Proc. of Workshop on Recommender Systems - Implementation and Evaluation*, 1999.

[4] Wasfi, A. M. A., Collecting User Access Patterns for Building user Profiles and Collaborative Filtering, In *Proc. of Int. Conf. on Intelligent User Interfaces*, 1999.

[5] Balabanovic, M. and Shoham, Y., Fab : Content-based collaborative recommendation, *CACM*, Vol.40, No.3, 1997.

[6] Good, N., Schafer, J. B., Konstan, J. A., Borchers, A., Sarwar, B., Herlocker, J. and Riedl, J., Combininig Collaborative Filtering with Personal Agents for Better Recommendations, In *Proc. of the AAAI-99*, 1999.

[7] Popescul, A., Ungar, L. H., Pennock, D. M. and Lawrence, S., Probabilistic Models for Unified Collaborative and Content-Based Recommendation in Sparse-Data Environments, In *Proc. of Conf. on UAI*, 2001.

[8] Kim, B.M., Li, Q., Park, C. S., and Kim, S., A New Approach for Combining Content-based and Collaborative Filters, *Journal of Intelligent Information Systems*, Vol. 27, No. 1, 2006.

[9] Li, Q., Kim, B. M., and S. H. Myaeng, Clustering for Probabilistic Model Estimation for CF, the 14'th International World Wide Web Conference (WWW 2005), 2005.

[10] Li, Q., Kim, B. M., Guan, D. H. and Oh, D., A Collaborative Music Recommender based on Audio Features, In *Proc. Of ACM SIGIR-04*, 2004.

[11] Han, P., Xie, B., Yang F., and Shen, R., A Novel Distributed Collaborative Filtering Algorithm and Its Implementation on P2P Overlay Network, In *Proc. of PAKDD 2004*, 2004.

[12] Pitsilis, C. and Marshall, L., A Proposal for Trust-enabled P2P Recommendation Systems, *Technical Report Series CS-TR-910*, University of Newcastle upon Tyne, 2005.

[13] Tveit, A., Peer-to-peer based Recommendations for Mobile Commerce, In *Proc. of the First International Mobile Commerce Workshop*, 2001.

[14] Chawathe, Y., Ratnasamy, S. and Breslau, L., Making Gutella-like P2P Systems Scalable, In *Proc. of ACM SIGCOMM03*, 2003.

[15] Plaxton, C. G., Rajaraman, R. and Richa, A. W., Accessing Nearby Copies of Replicated Objects in a Distributed Environment, In *Proc. of Ninth annual ACM symposium on Parallel algorithms and architectures*, 1997.

[16] Ali, K. and Wijnand, V.-S., TiVo: Making Show Recommendations Using a Distributed Collaborative Filtering Architecture, In *Proc. of KDD04*, 2004.

[17] Somlo, G., and Howe. A. E., Using Web Helper Agent Profiles in Query Generation, In *Proc. of AAMAS 2003*, 2003.

[18] Sarwar, B., Karypis, G., Konstan, J. and Riedl, J., Item-based Collaborative Filtering Recommendation Algorithms, In *Proc. of WWW Conference*, 2001.

[19] Resnick, P., Iacovou, N., Suchak, M., Bergstorm, P. and Riedl, J., GroupLens: An Open Architecture for Collaborative Filtering of Netnews, In *Proc. of ACM Conf. on Computer-Supported Cooperative Work*, 1994.

[20] Breese, J. S., Heckerman, D. and Kardie, C., Empirical Analysis of Predictive Algorithms for Collaborative Filtering, In *Proc. of UAI 98*, 1998.

[21] Hofmann, H., Latent Semantic Models for Collaborative Filtering, *ACM transactions on Information Systems*, Vol. 22, No. 1, 2004.

[22] Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J., Incremental SVD-based Algorithms for Highly Scalable Recommender Systems, In *Proc. of the 5th International Conference in Computers and Information Technology*, 2002.

[23] Xue, G.-R., Lin, C., Yang, Q., Xi, W., Zeng, H.-J., Yu, Y. and Chen, Z., Scalable collaborative filtering using cluster-based smoothing, In *Proc. of ACM SIGIR05*, 2005.

[24] Li, Q. and Kim, B. M., Constructing User Profiles for Collaborative Recommender System, *Lecture Notes in Computer Science (LNCS)*, Vol. 3007, 2004.

김 병 만

1987년 서울대학교 컴퓨터공학과(학사) 1989년 한국과학기술원 전산학과(석사) 1992년 한국과학기술원 전산학과(박사) 1992년~현재 금오공과대학교 교수. 1998년~1999년 미국 UC, Irvine 대학 방문교수. 2005년~2006년 미국 콜로라도 주립대학 방문교수. 관심분야 인공지능, 정보검색, 정보보안

이 경

1999년 하얼빈 공정대학교 기계공학과(공학사). 2001년 하얼빈 공정대학교 기계공학과(공학석사). 2005년 금오공과대학교 컴퓨터공학과(공학박사). 2005년~2006년 한국정보통신대학교(ICU) 박사학위 후 연수. 2006년~현재 Arizona State University 박사학위 후 연수. 관심분야는 정보검색, 정보필터링, 추천시스템, 인공지능

Adele E. Howe

1983년 Computer Science and Engineering, University of Pennsylvania (B.S.E.) 1987년 Computer and Information Science, University of Massachusetts (M.S.). 1993년 Computer and Information Science, University of Massachusetts (Ph. D.). 1992년~현재 Computer Science Department, Colorado State University 교수. 1996년~현재까지 Knowledge Engineering Review 에디터로 활동하고 있으며 AAAI-2007 Co-Chair로도 활동 중. 관심분야는 인공지능, Planning

여 동 규

1999년 금오공과대학교 컴퓨터공학과 학사. 2001년 금오공과대학교 컴퓨터공학과 공학석사. 2004년 금오공과대학교 컴퓨터공학과 박사수료. 관심분야는 정보보호, 인터넷프로토콜, 무선이동, 네트워크, 추천시스템