블로그 검색을 위한 태그 기반 피드 포스트 랭킹 알고리즘

한승균[a], 이상진[b], 박종헌[c]

[a] NHN 검색품질Lab
정기도 성남시 분당구 서현동 266-1 퍼스트타워 13층
Tel:+ 82-31-600- 6286, E-mail: jackleg83@nhncorp.com

[b]서울대학교 산업공학과 박사과정
151-742, 서울 관악구 관악로 599
Tel: +82-2-882-0504, E-mail: sjinlee@snu.ac.kr

[c]서울대학교 산업공학과 부교수
151-742, 서울 관악구 관악로 599
Tel: +82-2-880-7174, E-mail: jonghun@snu.ac.kr

Abstract

본 논문은 Web 2.0시대의 새로운 컨텐츠 매체로 각광받고 있는 블로그와 관련하여 태그 기반의 검색 알고리즘을 제안하고자 한다. 최근 블로그 검색과 관련하여 태그 기반의 블로그 검색 서비스가 등장하기 시작했지만, 현재 제공되는 태그 기반의 검색 서비스는 태그의 유무와 컨텐트의 최신성을 주요 기준으로 삼고, 태그와 컨텐트 간의 관련성을 제대로 고려하지 않아 검색 결과가 만족스럽지 못하는 경우가 많다.

따라서 본 논문에서는 태그와 컨텐트와의 관련성을 실수화하고 이를 주요 기준으로 검색 결과의 순위를 결정하는 PTRank 알고리즘을 제안하였다. PTRank 알고리즘에서는 1) 태그가 피드의 제목에 포함되었는지 여부, 2) 태그가 피드의 설명에 나타나는 회수, 3) 태그가 아이템의 제목에 포함되었는지 여부, 4) 태그가 아이템의 설명에 나타나는 횟수, 5) 피드 내에서 태그의 IDF값, 6) 사용자의 검색 행위를 이용해 태그와 컨텐트간의 관련성을 실수화하였다. 실험 결과, PTRank 모델 및 학습 알고리즘이 태그 기반의 피드 검색에서 잘 작동하며 검색에 효과적으로 활용될 수 있다는 것을 알 수 있었다.

Keywords:

블로그, 태그, 검색, 랭킹, 학습, 웹 2.0

서론

최근 Web 2.0 시대로 바뀌면서 소비자들은 직접 자신들이 원하는 새로운 서비스를 만들 수 있게 되었고(web services), 다양한 정보들을 취합해 새로운 정보를 재생산할 수 있게 되었으며(syndication), 보다 빠르게 정보를 퍼뜨릴 수 있는 도구와(blogging), 분산적인 플랫폼(Napster)을 손에 넣게 되었다(O'Reilly, 2005). 일반 사용자들이 자신의 정보, 혹은 컨텐트를 웹에 게시하기 위한 편리한 도구의 하나로 블로그를 사용하기 시작했으며, 블로그 사용량의 증가로 인한 블로그 검색, 특히 태그 기반의 블로그 검색 서비스가 제시되었다. 하지만 현재 제공되는 태그 기반의 검색 서비스는 태그의 유무와 컨텐트의 최신성을 주요 기준으로 삼고, 태그와 컨텐트 간의 관련성을 제대로 고려하지 않아 검색 결과가 만족스럽지 못하는 경우가 많다. 이에 본 연구는 효과적인 블로그 검색을 위하여 피드(feed)를 활용한 새로운 랭킹 알고리즘을 제시하고자 한다. 피드는 특정 웹 사이트의 업데이트 여부를 사용자들에게 알리기 위한 하나의 규약으로 현재 태그와 더불어 그 사용량이 크게 증가되고 있으며(Sifry, 2007), 실제 사용자들이 피드에 추가한 태그 정보를 활용할 수 있다면 의미적인 검색까지도 가능하게 하여 기존의 검색 방식보다 훌륭한 결과를 나타낼 수 있을 것이다.

따라서, 본 연구에서는 이러한 피드에 달려 있는 태그들을 활용한 검색을 통해 피드 검색에 보다 효율적이고 정확한 방식의 랭킹 알고리즘을 구현해보고자 한다.

관련 연구들

최근 태그, 피드와 같은 웹 리소스의 관계를 적절히 모델링할 수 있는 방법 및 해당 리소스에서 키워드 등의 정보를 추출하는 방식에 대한 연구가 활발히 진행 중이다.

Graupmann 등(2005)은 XML문서에서 효율적으로 데이터를 추출하고 이를 검색 알고리즘에 활용할 수 있는 알고리즘을 제안했다. 이들은 이러한 XML의 환경에서 다양한 메타데이터(XML 엘리먼트의 이름,

속성 등)을 효과적으로 분석하고 검색에 활용하기 위한 쿼리 언어와 알고리즘을 제안하고, 이를 구현한 SphereSearch Engine을 제안했다.

XML에 국한시키지 않은 일반적인 웹 문서의 경우, 검색 등을 위한 연구가 활발히 진행되고 있는데, 최근 Yih 등(2006)은 웹 문서에서 올바른 광고 검색 키워드를 추출하는 알고리즘을 제안했다. 이들은 웹 페이지에서 검색 키워드를 추출하기 위해 HTML 및 HTTP 메시지에서 얻을 수 있는 다양한 메타데이터를 사용하고, 이를 수학적으로 모델링하기 위한 방법론을 제안했다.

Wu 등(2006)은 보다 본격적으로 태그를 통해 관련 그룹을 분석하고 이를 제안하는 등의 알고리즘에 대해 제안했다. 이들은 주로 커뮤니티에 초점을 맞추어 태그를 검색에 활용했다. 즉, 어떤 문서, 어떤 사용자가 어떤 식의 태그를 사용했는지 분석해 비슷한 관심사를 가지고 있는 하나의 커뮤니티를 발견하고, 그 커뮤니티에 속한 사용자와 문서를 추천하는 방식이다.

학습 모델을 이용해 일반적인 랭킹 알고리즘을 개발하는 많은 연구들이 진행되어 왔다. PRank(Crammer K. and Singer Y., 2000)는 perceptron을 응용한 랭킹 알고리즘이다. Prank는 임의의 데이터가 주어졌을 때, 가장 적절한 k개의 순서화 분류를 선택하는 알고리즘이다. Freund 등(2003)은 RankBoost를 제안했는데, 이 모델은 boosting 개념을 적용한 랭킹 알고리즘이다. 이 방식은 메타 검색 엔진의 결과나 책, 음반, 영화와 같은 상품 추천 등에 사용될 수 있는 개념이다.

RankNet(Burges, 등 2005)은 neural network를 이용해 랭킹에 대한 확률 함수를 학습했다. 즉, 두 데이터 A, B가 있을 때 A가 B보다 랭킹이 높을 확률에 대한 cost function을 정의하고, 이것을 neural network를 이용해 학습시켰다. 이 과정에서 이들은 gradient descent method를 사용했다.

RankNet에서 사용한 cost function 정의 및 이를 학습시키기 위한 gradient descent 방식은 본 연구의 방향과 매우 흡사하다. 하지만 RankNet과는 달리, 본 연구는 웹, 그 중에서도 갱신 주기가 짧은 피드의 랭킹에 대한 것이기 때문에 일정 시간을 두고 테스트 셋을 이용해 모델을 학습시키는 과정이나 사용자와 상호작용이 불가능한 방법론을 직접 적용하기는 힘들다. 따라서 본 연구에서는 이상에서 알아본 선행 연구들의 방식들의 장점을 취하고, 온라인이라는 환경에 대한 학점을 보완하는 랭킹 알고리즘을 개발해보고자 한다.

PTRank 알고리즘

문제 정의

일반적인 웹 페이지들과는 달리 대화체의 구조가 많으며 문서 내 하이퍼링크의 수가 상대적으로 적은 flat한 구조를 가지는 특성을 지닌 블로그는 일반적인 검색 및 랭킹 알고리즘이 적용되기 적합하지 않은 경우가 많다(Brooks와 Montanez, 2006). 따라서 웹 페이지들 간의 하이퍼링크 구조를 이용해 페이지의 랭크를 결정하는 대표적인 웹 페이지 랭킹 알고리즘 중의 하나인 PageRank(Page 등, 1998)와 같은 알고리즘이 직접 적용되는 데 무리가 있다.

이러한 형식적인 차이뿐만 아니라 성격적인 측면에서도 블로그는 웹 검색과 차이를 보인다. Fujimura 등(2006)의 사용자 연구 결과에 따르면, 블로그 검색이 특정 주제에 대한 내용을 검색하는 데(topic search) 일반적인 웹 검색보다 두 배 이상의 만족도를 보였다. Mishne와 Rijke(2006)가 Blogdigger [1] 의 검색 로그를 분석한 결과에서도 주제에 대한 블로그 포스트를 검색하는 것이 블로그 검색의 중요한 사용자 행위 중 하나라고 밝혔다.

즉, 태그는 사람들이 직접 웹 리소스와 관계 있는 키워드를 지정하기 때문에, 검색 서비스에서 1) 웹 리소스의 인덱싱 과정이 필요 없으며, 2) 의미 있는 키워드를 추출할 수 있다는 장점이 있다. 하지만 현재 제공되고 있는 태그 기반 검색 서비스들은 단순한 키워드 매칭에 그쳐 이러한 태그의 장점을 충분히 살리지 못하고 있다. 또한 피드가 가지고 있는 정보 전달의 신속성을 나타내기 위해 검색 결과의 랭킹을 포스트가 작성된 시간을 기준으로 결정한다. 이러한 방식은, 1) 태그 검색 역시 과거의 검색 서비스에서 키워드와 웹 컨텐츠의 관계가 키워드와 태그로 변한 것뿐으로, 2) 정보의 신속성만 강조되어 의미 있는 순위 결정이 불가능하다.

따라서 본 연구에서는 현재 제공되고 있는 태그 기반 서비스를 개선하기 위한 알고리즘(PTRank 알고리즘)을 제안하고자 한다. 이 알고리즘은 태그와 피드 아이템의 관계성을 구하는 과정에 초점을 맞추고, 이 관계성을 기준으로 피드 아이템의 검색 순위를 결정할 것이다.

1) 시스템은 RSS 2.0 Specification에 정의되어 있는 피드 및 아이템 정보를 가지고 있다.
2) 아이템들은 하나 이상의 태그를 가지고 있다.
3) 사용자가 입력한 임의의 키워드에 대해, 시스템은 그 키워드를 태그로서 가지고 있는 아이템들의 리스트를 사용자에게 제공한다.
4) 아이템들의 리스트는 PTRank 알고리즘에 의해 정의된다.

PTRank 알고리즘

시스템에는 n개의 아이템($i_k(k=1...n)$)이 존재하며, 이 아이템들은 각각 자신의 태그를 가지고 있다($t_{k,j}(k=1...n)$). 사용자가 이 시스템에 임의의 키워드(*keyword*)를 입력하면, 키워드를 태그로서 갖는 아이템들의 리스트를 정렬해 반환한다.

[1] http://blogdigger.com, RSS 및 블로그 검색 서비스

첫 번째 단계에서, 사용자가 입력한 키워드를 태그로 가지고 있는 아이템들을 모두 얻어온다. 현재 태그 검색을 제공하고 있는 대부분의 서비스들은 이 단계에서 얻어진 아이템들(LI)을 최신성(recency)만을 기반으로 랭킹을 결정한다. 하지만 PTRank 알고리즘에서는 최신성보다는 사용자가 입력한 키워드와 아이템의 관계성(relevancy)이 더 높다고 판단되는 아이템의 순위를 높게 책정한다. 즉, 이 알고리즘에서는 다음과 같은 함수가 존재한다.

$$\hat{r}_{k,j} = R(i_k, t_{k,j}) \quad (1)$$

이 함수는 아이템과 태그를 입력 값으로 받아, 그 둘의 관계성을 실수로 계산한다. PTRank Algorithm은 이 관계성을 우선하여 아이템의 랭킹을 결정한다. 즉, $\hat{r}_{k,j}$ 값이 클수록 태그에 대한 아이템의 랭킹이 높다. 단, 한 태그에 대해서 둘 이상의 아이템의 관계성이 동일한 경우엔, 아이템의 최신성을 기준으로 랭킹을 부여한다.

Input	*keyword*
Step	1. Retrieve all items ($LI = \{i_k\}$), where $\exists j$, s.t. $t_{k,j} = keyword$ 2. Order items ($i_k \in LI$) by $\hat{r}_{k,j}$, and recency. A. if $\hat{r}_{a,j_a} > \hat{r}_{b,j_b}$ i. i_a is higher than i_b. B. if $\hat{r}_{a,j_a} = \hat{r}_{b,j_b}$ i. i_a is higher than i_b, if i_a is newer than i_b. ii. i_a and i_b are same rank, if i_a and i_b are published at same time. iii. i_a is lower than i_b, if i_a is older than i_b C. if $\hat{r}_{a,j_a} < \hat{r}_{b,j_b}$ i. i_a is lower than i_b Get a ordered item list from step 3, LI_o.
Output	Ordered item list, LI_o

그림 1 - PTRank Algorithm

관계성 함수 정의

PTRank 알고리즘의 핵심은, 태그와 아이템의 관계성을 계산하는 함수 $R(i_k, t_{k,j})$ 이다. PTRank 알고리즘은 관계성 함수를 기준으로 랭킹을 부여하기 때문이다.

Yih 등(2006)은 웹 페이지에서 광고와 관련된 키워드를 추출하기 위해서 웹 페이지를 나타내는 메타 데이터들을 사용했다. 이들은 키워드의 대문자 여부(capitalization), 다른 문서와의 연결에 사용되었는지 여부(hypertext), HTML meta 태그에 사용되었는지의 여부, HTML *title* 태그에 사용되었는지의 여부, URL 및 TF(term frequency), DF(document frequency) 값, 문서에서의 위치, 키워드의 길이, 사람들 웹 문서를 찾기 위해 사용한 키워드(query log)를 복합적으로 조합해 웹 페이지와 관계 있는 광고를 제안하기 위한 키워드를 추측했다. 본 연구에서는 아이템과 태그의 관계성을 설명할 때도 이 방법론을 적용하여, 피드에 기술되어 있는 데이터 및 사용자들 간의 피드백을 중심으로 관계성 함수가 정의하였다. 본 연구에서 관계성 함수 정의에 사용한 데이터들은 다음과 같다.

Channel Title($CT_{k,j}$)과 Channel Description($CD_{k,j}$)은 태그가 아이템이 속한 채널의 메타 데이터인 제목과 설명에 포함되어 있는지 여부를 뜻한다. 채널은 아이템들을 모두 포함하는 일종의 컨테이너 개념으로, 모든 아이템들의 상위 개념이라고 볼 수 있다. 즉, 채널의 메타 데이터는 그 채널에 속한 모든 아이템들의 공통적인 특징을 나타낸다고 볼 수 있고, 태그와 아이템의 관계성을 결정하는 데 도움이 될 수 있다. $CT_{k,j}$ 는 boolean형 데이터로 태그가 채널의 제목에 속해 있으면 1, 그렇지 않으면 0을 가지며, $CD_{k,j}$ 는 태그가 채널의 설명에서 몇 번 나왔는지 계산한다.

Item Title($IT_{k,j}$)과 Item Description($ID_{k,j}$)은 각각 $CT_{k,j}$, $CD_{k,j}$ 와 같은 성격을 가진 값들이다. 태그가 아이템의 제목과 내용에 포함되어 있는지 여부로 측정된다. $IT_{k,j}$ 는 태그가 아이템의 제목에 포함되어 있으면 1, 그렇지 않으면 0을 가지며, $ID_{k,j}$ 는 태그가 아이템의 내용에 몇 번 나왔는지 계산한다.

IDF($IDF_{k,j}$)는 $tf \times idf$ 에서 사용하는 inverse document frequency 값을 뜻한다. 하나의 태그가 그 문서의 특징을 얼마나 잘 표현하고 있는가를 나타내기 위한 항(term)으로, 일반적인 용어를 추출할 경우에는 일반적인 용어들(and, the, I 등과 같이 자주 쓰이는 단어들)을 모두 고려해야 하므로 시스템 내의 전체 문서를 대상으로 값을 구한다. 하지만 태그는 사용자가 아이템을 '잘 나타낸다고 생각하는' 키워드를 직접 지정한 것이기 때문에, 모든 문서를 대상으로 하지 않고, 아이템이 속해 있는 채널 안에서 정의한다. 즉, $IDF_{k,j}$ 는 다음과 같이 정의된다.

$$IDF_{k,j} = \log \frac{N_k}{n_{k,j}} \quad (2)$$

N_k = i_k가 속한 채널의 전체 아이템 수

$n_{k,j}$ = i_k가 속한 채널의 아이템들 중, t_j를 가지고 있는 아이템 수

마지막으로 query log($QL_{k,j}$)는 Yih 등(2006)이 사용한 메타 데이터와 비슷하다. Query log 값의 목적은 사용자들이 어떤 키워드에 관심이 있으며, 그 키워드와 관련된 아이템이 무엇인지 암묵적인(implicit) 피드백을 받은 결과를 반영하기 위함이다. 사용자들이 임의의 태그로 검색한 결과로 보여진 아이템들 중 한 아이템을 선택한 경우, 그 아이템의 태그에 대한 선택

빈도수(QueryFrequency$_{k,j}$)가 하나씩 증가한다. 즉, QueryFrequency$_{k,j}$ 는 $t_{k,j}$ 로 i_k 를 검색한 횟수를 뜻한다. 이 때, $QL_{k,j}$ 는 다음과 같이 정의된다.

$$QL_{k,j} = \log(1 + QueryFrequency_{k,j}) \quad (3)$$

이상과 같이 정의된 변수들을 이용해 관계성 함수를 정의할 수 있다. 이 때 관계성 함수는 지속적인 사용자들의 피드백과 검색 행위(behavior)를 통해 학습될 수 있어야 한다. 본 연구에서 다음과 같이 neural network의 perceptron을 이용하여 관계성 함수를 다음과 같이 정의하였다.

$$R_p(i_k,t_{k,j}) = w_{CT}CT_{k,j} + w_{CD}CD_{k,j} + w_{IT}IT_{k,j} + w_{ID}ID_{k,j} + w_{IDF}IDF_{k,j} + w_{QL}QL_{k,j} \quad (4)$$

Perceptron을 이용해 관계성 함수를 정의하면 식이 간단해 구현이 쉽고 계산 시간이 단축되며, 가중치 값을 통해 어떤 항목(term)이 중요한지 직관적으로 알 수 있다는 장점이 있다. 하지만 1차식을 사용하기 때문에 함수의 표현력에 한계가 있어 실제 값을 예측하는데 문제를 안고 있다. 따라서, 본 연구에서는 이러한 문제를 해결하기 위해 multilayer network를 이용하였으며 이 경우 input vector와 output vector의 관계식은 다음과 같이 정의된다.

$$z_k = f(\sum_{j=1}^{n_H} w_{kj} f(\sum_{i=1}^{d} w_{ji}x_i + w_{j0}) + w_{k0}) \quad (5)$$

위의 관계성 함수에서 w_{ba} 는 input node a에서 hidden node b로의 가중치이고, w_b 는 hidden node b에서 output node로의 가중치이다. 또한 n_H 는 hidden node의 개수를 뜻한다. 본 실험에서 hidden node는 10개를 사용했으며 각각의 node에서 $f_1(x)=x$, $f_2(x)=\tanh(x)$를 사용했다.

관계성 함수 학습 알고리즘

피드 데이터는 시간이 지남에 따라 지속적으로 생성된다. 따라서 본 연구에서는 사용자가 시스템에서 피드백을 줄 때마다 사용자의 피드백을 학습해 새로운 함수를 생성할 수 있는 학습 알고리즘을 사용하였으며, 관계성 함수 학습의 성능 평가 기준(criteria)은 SE(Squared Error)를 이용하였다.

관계성 함수를 통해 얻어진 관계도를 $\hat{r}_{k,j} = R(i_k,t_{k,j})$ 라 하고, 실제 관계도를 $r_{k,j}$ 라 하면, SE 함수 $SE(\vec{w})$, ($\vec{w}$ 는 가중치들을 벡터로 나타낸 것이다.)를 다음과 같이 정의한다.

$$SE(\vec{w}) = \frac{1}{2n}\sum_{k\in TrainingSet}(r_{k,j} - \hat{r}_{k,j})^2 \quad (6)$$

이 SE 함수를 최소화 하는 것을 목적으로 하는 학습 알고리즘을 정의한다. $R_p(i_k,t_{k,j})$ 를 위한 학습 알고리즘은 다음과 같다.

Input	Trained Weights($\vec{w}(t)$), $TrainingSet = \{k = 1..t \mid (i_k, t_{k,j}, r_{k,j})\}$, And learning rate($\alpha_t$)
Step	1. Update weights. $c \in \{CT, CD, IT, ID, IDF, QL\}$ A. $w_c(t+1) = w_c(t) + \Delta w_c(t)$, $\Delta w_c(t) = -\alpha_t \frac{\partial SE(\vec{w})}{\partial w_c}$ i. $\frac{\partial SE(\vec{w})}{\partial w_c} = -\frac{1}{n}\sum_{k=1}^{t} x_{k,j}(r_{k,j} - \bar{r}_{k,j})$ 2. Update relevancies of all data with updated weights. 3. Calculate squared error, $SE_t(\vec{w}) = \frac{1}{2n}\sum_{k\in TrainingSet}(r_{k,j} - \hat{r}_{k,j})^2$ 4. Update learning rate based on the error function. A. if $SE_{t-1}(\vec{w}) \geq SE_t(\vec{w})$ then $\alpha_{t+1} = 2 \times \alpha_t$ Otherwise, $\alpha_{t+1} = \frac{1}{4} \times \alpha_t$
Output	Trained Weights, $\vec{w}(t+1)$

그림 2 - Learning Algorithm for $R_p(i_k,t_{k,j})$

우선 사용자로부터 검색 결과에 대한 피드백을 받는다. 이 피드백에는 아이템, 검색에 사용된 태그, 그리고 그 둘의 실제 관계성 값으로 이루어진다. 이 피드백이 사용자로부터 얻어질 때마다 training set에 포함된다.

Training set이 업데이트될 때마다, 반복적으로(iteratively) 가중치를 개선한다. 이 과정에서, 우리는 error function인 $SE(\vec{w})$ 의 값을 최소화시키는 것이 목적이므로, gradient descent 방법을 사용한다. 이 방법을 사용해 가중치를 업데이트하는 식은 다음과 같다.

$$w_c(t+1) = w_c(t) + \Delta w_c(t), \quad \Delta w_c(t) = -\alpha_t \frac{\partial J(\vec{w})}{\partial w_c} \quad (7)$$

일반적으로 Gradient descent 방식에서는 learning rate, $\alpha_t(>0)$ 를 업데이트를 해줘야 하는데, 본 연구에서는 Lapidus 등(1961)이 다른 방식에 비해 빠르게 수렴한다고 제안한 least-squares procedure에서 사용되는 방식을 적용시켰다. 즉, 업데이트 된 후의 오차 값이 전보다 작아지면, 우리가 원하는 방향으로 이동하는 것이라 판단하고 learning rate를 2배 증가시킨다($\alpha_{t+1} = 2 \times \alpha_t$). 그렇지 않은 경우에는 learning rate를 1/4로 감소시킨다($\alpha_{t+1} = \frac{1}{4} \times \alpha_t$). $R_m(i_k,t_{k,j})$ 은 기본적인 learning algorithm step은 $R_p(i_k,t_{k,j})$와 동일하나, backpropagation algorithm을 사용한다.

실험 및 결과 분석

도메인 선정

블로그에서 생성되는 정보의 유용성을 많은 검색 서비스들이 인지하면서, 블로그 검색 서비스를 개시했다. 국내에서는 네이버[2], 다음[3] 등이 이미 블로그 검색 서비스를 시작했으며, 국외에서는 구글과 같은 상위 검색 엔진에서 블로그 검색 서비스[4]를 시작했다. 이러한 블로그 검색 서비스들 중에서 가장 많은 시장을 점유하고 있는 서비스가 Technorati이다. 각종 온라인 통계치를 제공하는 HitWise의 연구원인 Prescott(2007)에 따르면, Technorati의 2007년 3월 미국 인터넷 전체 시장 점유율은 0.0046%로 1월부터 계속 구글의 블로그 검색 서비스를 따돌렸다. 따라서, 본 연구의 실험은 현재 블로그 서비스들 가운데 가장 신임할 수 있고 많은 양의 데이터를 축적하고 있는 Technorati의 데이터를 사용하였으며, 실제 데이터 수집은 Technorati에서 제공하는 API[5]를 이용하였다.

시스템 프로트타입 및 검증실험

본 연구에서 제안한 PTRank 알고리즘은 사용자들의 피드백을 통해 지속적으로 포스트와 태그의 관계성을 학습해간다. 따라서 실제로 사용자들이 피드의 검색 결과를 보고 피드백을 해 줌으로써 가중치를 반복적으로 학습시켜야 하기 때문에, 본 연구에서는 수집된 데이터들 중 임의의 피드를 선택해 테스트 셋을 만들고 피실험자들에게 임의의 랭킹을 보여 주어 쿼리 로그 및 랭킹에 대한 피드백을 받는 방식을 통하여 알고리즘의 성능을 분석하였다.

사용자가 시스템 프로토타입에 접속하면, 시스템은 트레이닝 셋에 있는 태그 중 임의의 하나의 태그(Keyword)를 선택하고, 그 태그를 가지고 있는 포스트 중 임의의 두 개의 포스트를 사용자에게 보여주고, 사용자로부터 피드백을 받는다. 검증 실험에서 피실험자들은 동일한 태그로 검색된 10개의 포스트를 가지고 있는 두 개의 리스트를 보고 키워드와 관련 있는 포스트를 체크하도록 요청되었다.

아래 그림에서 나타난 바와 같이 피실험자들에게 임의의 키워드(위 그림의 경우 'shopping')에 대한 검색 결과 리스트 2개가 주어진다.

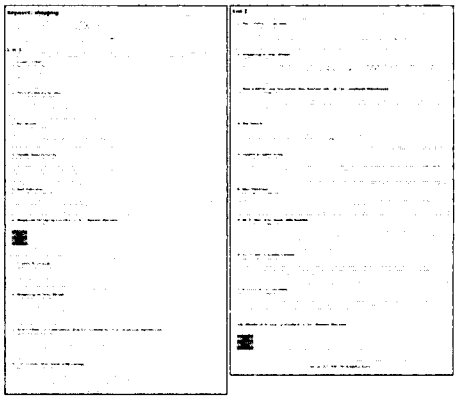

그림 3 – 검증 실험에 사용된 사용자 인터페이스

피실험자들은 이 리스트들 중에서 키워드와 관련이 있다고 판단되는 검색 결과를 체크하는 형식으로 피드백을 줄 수 있다.

결과 분석

PTRank를 적용한 프로토타입은 Ruby on Rails[6]를 이용해 구현되어 피실험자들에게 공개되었다. 학습 실험 과정 동안 약 1,000여 개의 피드백이 발생했다. 이를 이용해 $R_p(i_k,t_{k,j})$ 와 $R_m(i_k,t_{k,j})$ 의 가중치 학습이 진행되었다. PTRank 알고리즘과 Technorati의 검색 방식을 비교한 유효성 검증 실험 결과를 분석하기 위해 $top@n$ 값을 정의했다. $top@n$ 은 순위가 $1 \sim n$ 인 검색 결과에서 사용자가 만족할 만한 검색 결과가 얼마나 포함되어있는가를 나타낸다. $top@n$ 은 다음과 같다.

$$top@n = \frac{nsn}{n \times nf} \tag{8}$$

위 식에서 nf 는 피드백을 받은 수, nsn 은 사용자가 순위 $1 \sim n$ 의 검색 결과 중 만족할 만한 검색 결과가 포함된 개수(the number of satisfaction in ranking n)를 뜻한다.

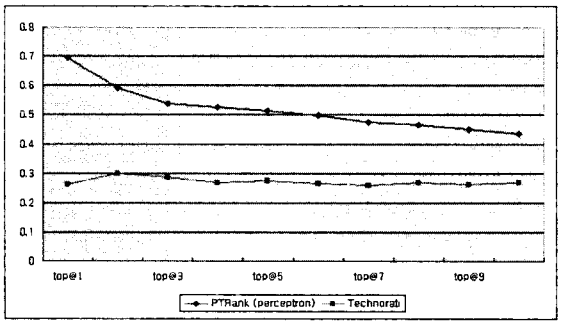

그림 4 – PTRank와 Technorati의 검색 결과 비교

피실험자들이 PTRank와 Technorati의 검색 결과에 대한 만족도를 $top@n$ 으로 구한 값은 위의

[2] 네이버 블로그 검색, http://blogsearch.naver.com/
[3] 다음, http://www.daum.net
[4] Google Blog Search, http://blogsearch.google.com/
[5] *Technorati API Documentation*, http://technorati.com/developers/api/
[6] Ruby on Rails, http://www.rubyonrails.org/

그림에서 알 수 있듯이 PTRank 알고리즘의 검색 결과과 더 좋은 만족도를 보이고 있다.

Technorati에서 제공하는 검색 결과에 대한 *top@n* 값의 경우, n에 관계 없이 약 0.25 ~ 0.3의 값을 나타내고 있다. 이것은 Technorati의 태그 검색 결과가 사용자들에게 썩 좋은 만족도를 주지 못하고 있으며, 검색 결과의 순위에 큰 의미가 없다고 판단된다. 즉, 사용자는 Technorati에서 제공하는 검색 결과에서 원하는 정보를 찾기 위해 리스트의 모든 정보를 확인해야 한다. 반면 PTRank 알고리즘이 적용된 리스트의 경우, *top@1* 값이 약 0.7 정도로 Technorati의 *top@1* 값보다 약 2.5배 높으며, 대부분의 경우 만족스러운 결과를 보여주고 있다. 또한 n이 커질수록 *top@n* 값이 점차 감소하는 형태를 보이고 있어 사용자가 원하는 검색 결과는 높은 순위에 있을 가능성이 높다는 것을 반영하고 있다. 즉, 높은 순위에 있는 검색 결과에 사람들이 만족하고 있으며, 순위가 낮아질수록 검색 결과의 만족도가 떨어지는 형태를 보인다. 또한 가장 만족도가 낮은 *top@10* 의 경우에도 Technorati의 리스트보다 높은 만족도를 제공하고 있다.

결론

본 연구에서는 태그 기반의 검색 서비스에서 사용될 수 있는 태그와 피드의 관계성을 이용해 검색 랭킹을 구현한 PTRank 알고리즘에 대해 연구했다. PTRank 알고리즘은 태그와 아이템의 관계성을 실수화 한 관계성 함수를 perceptron과 multilayer network를 이용해 정의했으며, 이를 이용해 주어진 태그와 관련된 아이템들의 랭킹을 결정했다.

관계성 함수는 태그와 아이템의 관계를 정의할 수 있다고 판단할 수 있는 6개의 파라미터를 사용했다. 이들은 1) 아이템이 속한 피드의 제목, 2) 아이템이 속한 피드의 설명, 3) 아이템의 제목, 4) 아이템의 설명, 5) 피드 내에서 태그의 idf, 그리고 6) 태그와 아이템의 query frequency의 log 값이다.

본 연구에서 제안한 PTRank 모델과 학습 알고리즘을 적용해 태그 기반 피드 검색 시스템의 프로토타입을 구현해 실제로 얼마나 사용자들의 니즈를 충족시켜줄 수 있는지 검증했다. 실험 결과 PTRank 모델 및 학습 알고리즘이 태그 기반의 피드 검색에서 잘 작동하며 검색에 효과적으로 활용될 수 있다는 것을 보였다.

Acknowledgments

이 논문은 2007년도 정부(과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임(No. R01-2007-000-11167-0)

참고문헌

[1] Baeza-Yates R. and Ribeiro-Neto B. (1999). *Modern Information Retrieval*, Addison Wesley.

[2] Brooks C.H., Montanez N. (2006), "Improved Annotation of the Blogosphere via Autotagging and Hierarchical Clustering", *WWW2006*, Edinburgh, Scotland.

[3] Burges C., Shaked T., Renshaw E., Lazier A., Deeds M., Hamilton N., and Hullender G. (2005), "Learning to Rank using Gradient Descent", *The 22nd International Conferences on Machine Learning*, Bonn, Germany.

[4] Crammer K., and Singer Y. (20002), Pranking with ranking, *Advances in Neural Information Processing Systems*, Vol. 14.

[5] Freund Y., Iyer R., Schapire R., and Singer Y. (2003), "An Efficient Boosting Algorithm for Combining Preferences", *Journal of Machine Learning Research*, Vol. 4, pp.933-969.

[6] Fujimura K., Toda H., Inoue T., et al. (2006), "BLOGRANGER – A Multi-faceted Blog Search Engine", *WWW2006*, Edinburgh, Scotland.

[7] Graupmann J., Schenkel R., and Weikum G. (2005) "The SphereSearch Engine for Unified Ranked Retrieval of Hetrogeneous XML and Web Documents", *31th VLDB Conferences*, Trondheim, Norway.

[8] Kitayama D., Sumiya K. (2006), "A Blog Search Method using News Video Scene Order", *12th Multi-Media Modelling Conference*, Beijing, China.

[9] Lapidus L., Shapiro E., Shapiro S., et al. (1961), "Optimization of Process Performance", *AICHE Journal*, Vol. 7, Issue 2, pp.288-294.

[10] Mishne G., Rijke M. (2006), "A Study of Blog Search", *28th European Conference on Information Retrieval*, London, UK.

[11] Nielsen//NetRatings (September 20, 2005), RSS Users Visit Three Times as many News Web Sites as Non-users, http://www.nielsen-netratings.com/pr/pr_050920.

[12] O'Reilly T. (2005), What is Web 2.0, http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html.

[13] Page L., Brin S., Motwani R., and Winograd T. (1998), "The PageRank citation ranking: Brining order to the Web", *Technical Report*, Stanford University, Stanford, CA.

[14] Sifry D. (April 5, 2007), The State of the Live Web, April 2007, http://www.sifry.com/alerts/archives/000493.html.

[15] Wu H., Zubair M., and Maly K. (2006), "Harvesting Social Knowledge from Folksonomies", *HT'06*, Odense, Denmark.

[16] Yih W.T., Goodman J., and Carvalho V.R. (2006), "*Finding Advertising Keywords on Web Pages*", *WWW2006*, Edinburgh, Scotland.