DEWS2008 B9-4

多様に変化するユーザの興味を考慮したウェブサイト推薦システム

中本　レン[†]　中島　伸介[†]　宮崎　純[†]　植村　俊亮[††]　加藤　博一[†]

[†] 奈良先端科学技術大学院大学情報科学研究科
[††] 奈良産業大学情報学部情報学科
E-mail: †{reyn-n,shin,miyazaki,kato}@is.naist.jp, ††UemuraShunsuke@nara-su.ac.jp

あらまし　本研究では，リアルタイムに更新されるウェブサイト推薦において，ユーザの興味が多様に変化することを考慮したウェブサイト推薦システムを提案する．ユーザの現在の興味を考慮に入れるため，現在ユーザが参照しているウェブサイトに付与されているタグ情報と，そのユーザの嗜好情報とのマッチングに基づく，タグベースコンテキスト依存型協調フィルタリングを拡張する．被験者実験の結果，我々の提案システムは，従来一般に使われている検索・推薦技術よりも，関連性のより高い情報を提供することができることを確認した．この結果から，ユーザの多種多様な興味に対応するためには，推薦情報提供時のユーザの興味を考慮することが重要であるといえる．
キーワード　協調フィルタリング、タグ情報、情報推薦、情報検索

Live-Updating Website Recommendations Using Reasonable Tag-based Collaborative Filtering

Reyn NAKAMOTO[†], Shinsuke NAKAJIMA[†], Jun MIYAZAKI[†], Shunsuke UEMURA[††], and Hirokazu KATO[†]

[†] Graduate School of Information Science, Nara Institute of Science and Technology
[††] Department of Informatics, Faculty of Informatics, Nara Sangyo University
E-mail: †{reyn-n,shin,miyazaki,kato}@is.naist.jp, ††UemuraShunsuke@nara-su.ac.jp

Abstract　In this paper, we present a tag-based collaborative filtering recommendation method for use with recently popular online social tagging systems. Combining the information provided by tagging systems with the effective recommendation abilities given by collaborative filtering, we present a website recommendation system which provides live-updating personalized recommendations that update to match the user's changing interests as well as the user's bookmarking profile. Based upon user testing, our system provides a higher level of relevant recommendation over other commonly used search and recommendation methods. We describe this system as well as the relevant user testing results and its implication towards use in online social tagging systems.
Key words　collaborative filtering, tagging, recommendation systems, information retrieval

1. Introduction

As the Internet continues to mature and becomes more accessible to the common user, the amount of information available increases exponentially. Accordingly, finding useful and relevant information is becoming progressively difficult. Moreover, a lot of the information available–blogs, various types of reviews, and so forth–are highly subjective and thus, hard to evaluate purely through machine algorithms. Being subjective in nature, one person may absolutely love something while the next may loathe the same–no single authority exists. It is in these cases where people–more so than the current ability of machine algorithms–are greatly effective in evaluating and filtering this information.

For this reason, the idea of tag-based contextual collaborative filtering or TCCF was created and described in [13] and found to be effective in providing explicit recommendations in [12]. This method combines the strengths of both Collaborative Filtering (CF) as well as tagging information from social tagging services to provide effective, personalized recommendations to the user. We now enhance this one step further–to provide implicit recommendations which consider the user's current interest–and provide only recommendations which are related to this.

Indeed, many users often go right ahead and search using their favorite search engine to find whatever website they are looking for. However, there are times when implicit website recommendations are effective. Websites that they never knew existed and maybe never bothered to search for could be discovered and enjoyed by the user. However, these recommendations must be relevant to not only the user's preferences but also the topic the user is currently interested in. Thus, this is the motivation for our system: providing live-updating effective website recommendations for use in online social tagging systems–namely social bookmarking services like del.icio.us [4].

Our method, Reasonable Tag-Based Collaborative Filtering or RCF–does just this. Using the tagging-information from the currently viewed website, we provide effective website recommendations relevant to both the current page and the user's past bookmarking profile. RCF considers the reasons for liking their previously bookmarked websites and also the reasons why they are viewing the current webpage to provide "reasonable" recommendations to the end user.

In comparison to other basic search methods, we tested four methods–including our own–of generating recommendations based upon the user's current page. While each of the other methods provide good results in normal document search–our method was found to be most effective when considering implicit live-updating recommendations. In this paper, we describe these results and its corresponding implications.

2. Related Work

2. 1 Collaborative Filtering Systems

Collaborative Filtering (CF) is the process whereby the community of users is used to sort out relevant or important information from the non-relevant or non-important information. The process is based upon the idea that if users prefer the same item or items, then their preference will be similar for other items liked by the similar users. In other words, a user should enjoy the same items that their similar users like. From a wider perspective, once users have recorded their preferences within the system, subsequent users can benefit from the previous users' knowledge, hence the collaborative aspect of the system.

CF has been proven to work well under certain domains–mainly entertainment domains–such as usenet recommendations [14], movie recommendations [7], product recommendations [1], and so forth. Many CF systems rely upon a matrix of numerical ratings of resources by users [14]. Once enough ratings are in place, a similarity score is calculated between the user and other users. These similarity scores are multiplied by the ratings other users recorded and then averaged. Those resources with an average score above a certain threshold are recommended.

The main draw of CF is that it only attempts to understand if a user like some product or not. However, it does not consider the reasons why a user likes something.

2. 2 Social Tagging Systems

Tagging has been around for sometime, albeit known by other terms such as metadata, categorization, labels, and so forth. Tagging is the process of attaching natural language words as metadata to describe some resource like a movie, photo, book, etc. Tagging vocabulary is usually uncontrolled, whereby the user themselves can decide what word or combination of words are appropriate.

The current main use of tagging is for the purpose of retrieval [11], whereby users can search for a tag and the resources with that tag attached will be returned to the user. The user who added it can use the same tags for later retrieval. For other users, tags serve as a way to discover new resources by searching for whatever tag they are interested in.

In recent years, the advent of Social Tagging Systems have brought tagging back into the limelight. Currently, there are several online social tagging systems that are popular and are the subject of continuing research [11] [9]: they range from website bookmarking such as del.icio.us [4], photo sharing [5], research paper searching [2], to even people rating [3]! All of these sites use tagging for many purposes, but in addition to that, they focus on the social networking aspects of tagging to enhance the experience for end users. However, in their present form, tags are generally used for tag searching; user profile matching and subsequent recommendations are yet to be implemented. As mentioned before, tags provide the clues as to why a user liked something. They are the who, what, when, where, and why of the user's reason for tagging something [9]. Because of this, as well as the similar use of social networking, social tagging systems provide an ideal choice for combination with CF systems.

In terms of analyzing, [15] use a statistical approach towards deriving the emergent semantics of social tagging systems, namely del.icio.us. They use the Expectation Maximization algorithm (EM) algorithm [8] to automatically cluster documents, users, and tags. They explored the optimal number of domain clusters as well as the number of iterative steps to get satisfactory cluster sizes and document classification. Lastly, they explored creating a personalized tag search engine based upon their findings.

2. 3 TCCF Website Recommendation System

TCCF is the combination of traditional CF systems and social tagging systems to allow for personalized recommendation. The essential idea is that CF provides personalization, and tags provide the reasons why users liked something. Unlike traditional CF models which use numeric ratings, our TCCF model uses tags as the indicator of why a user likes something. For example, say we have a website bookmarking system where users can come in and bookmark websites that they enjoy using tags. Normally, the act of bookmarking a website is a strong indicator of whether something is liked. However, the used tags provide the key distinguishing factor from traditional CF systems–the tags attached to the resource can be seen as the reason in which the user likes the resource. Usually, the user will use tags to describe the resource as the user themselves see it, and in most cases it would be the reason why they liked something. Thus, lies the key difference between traditional CF and TCCF: In addition to considering whether or not a user likes a resource, it also attempts to consider why a users likes something by using the information provided by tagging systems.

In the TCCF Website Recommendation System, users bookmark websites they like using tags, and subsequently, they can easily retrieve their bookmarks by just searching by the tags. Once the the user has added enough bookmarks, the first step is finding similar users. This is then followed by calculating a score prediction–the level that the system thinks the user will like something–based upon similar users' bookmarks. Both of these steps consider bookmarking as well as the attached tags when generating recommendation candidates. This method is subsequently improved and will be further explained in section 3..

2.4 Topic-Based Vector Space Model

A type of Topic-based Vector Space Model is described in [10]. This model uses topics instead of terms as the parameters to form vectors of each of the documents within the set. Thus, when comparing documents with each other, each document is expressed in terms of how pertinent they are in each of the topic areas rather than in their individual term vectors. In doing so, you avoid some natural language problems–such as comparing synonyms, etc–that occur when doing a traditional tf-idf vector space model. Additionally, calculation complexity can be reduced depending on the number of parameters you choose. In [10], They suggest such methods as the k-nearest neighbor algorithm to cluster the documents. However, for our needs, k-nearest neighbor is inappropriate as documents and tags used in our system often exist under many domains. We instead adopt an approach similar to [15].

3. Reasonable Tag-Based Collaborative Filtering For Social Tagging Systems

The basis of our recommendation system is a website bookmarking system not unlike del.icio.us. A sample usage pattern would be as shown in figure 1.

Here, an arrow indicates a user has bookmarked a page and the tags above the arrow indicate the tags used while bookmarking. User A is bookmarking website 1 with the tags 'japanese' and 'dictionary'. Similarly, user B is bookmarking website 2 with the tags 'apple' and 'news'. Based upon this type of social bookmarking system, our system

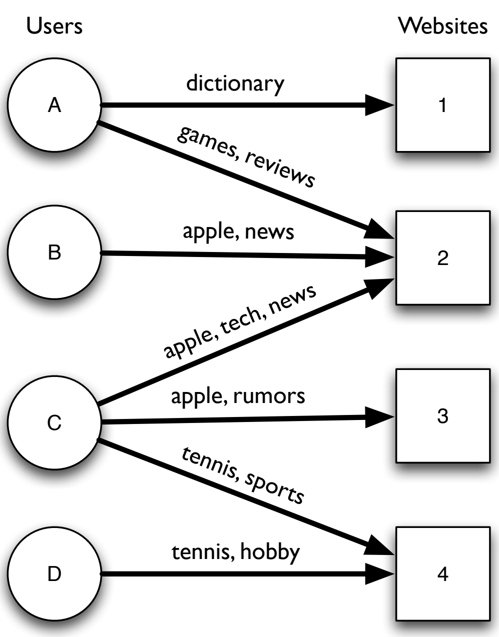


Figure 1 Website Bookmarking System Overview

generates effective recommendations based upon the user's bookmarking profile as well as their currently viewed page. We draw from the idea that a bookmark is an indicator of liking something, and moreover, that the tags are synonymous with the reasons the like it for.

That being said, our recommendation algorithm is heavily based upon comparing website's topic parameters, so first, we explain how we generate topic domain vectors for each website for use in our algorithm.

3.1 Clustering Documents into Topic Domains using EM

One of the drawbacks of TCCF was that it was purely based upon tag vector comparison when determining how similar two webpages were. This tag vector was basically a feature vector with the tags used as the parameters of their respective vectors. For example, in figure 1, website 1's vector would have a value of one for 'japanese' and 'dictionary'. Website 2 would have 'apple', 'news', 'games, 'reviews', and 'tech', with the respective values of two, two, one, one, and one.

This has the inherent problems that are associated with natural language problems. For example, vectors containing two synonyms, such as 'funny' or 'humourous', should be similar; however, when comparing only tags, these synonyms are considered different terms, and thus, their vector similarity will be low. To deal with this, we apply a similar approach to that of [15], by using the EM algorithm to cluster the websites into domains.

Our data consists of 3 months worth of mining of del.icio.us RSS feeds. In total, we have over 100,000 users, 2.5 million webpages, 3.6 million bookmarks, and 870,000 distinct tags. After mining the data, we subsequently stemmed the tags using the well-known Porter stemming algorithm [6] . Then,

D1	mac (0.69)	software (0.68)	osx (0.66)	tool (0.59)	apple (0.58)
D2	program (0.78)	develop (0.56)	refer (0.52)	code (0.34)	software (0.29)
D3	music (0.37)	audio (0.30)	mp3 (0.28)	blog (0.26)	web2.0 (0.23)
D4	photography (0.58)	photo (0.57)	image (0.42)	design (0.41)	art (0.38)
D5	design (0.65)	art (0.49)	graphic (0.45)	inspire (0.38)	refer (0.34)

Figure 2 Probability of term given a topic domain, $P(t|D)$

the number of unique stemmed tags was about 780,000. Over this data, we ran the EM algorithm for 50 iterations over 20,000 of the top bookmarked documents and the top 20,000 used tags and clustered them into 100 domains. The number of domains was chosen after a subjective comparison of the term vector they produced–sufficient enough to separate the documents enough, while avoiding making the topic vectors too sparse. Finding the optimal number of domains is described in [15] and is beyond the scope of our research. That being said, a sample set of domains and their conditional tag probabilities are shown in figure 2. Here, we list five sample domains, and the condition probabilities of the top five terms given that domain.

Following this, website feature vectors based on these topic domain clusters were created for each document as shown in equation 1 for webpage k. We will from now refer to these as 'topic domain vectors' or 'domain vectors' and abbreviate it as DV.

$$DV_k = (dw_{1,k}, dw_{2,k}, dw_{3,k}, ...dw_{100,k}) \tag{1}$$

Here, $dw_{j,k}$ is the topic domain weight of website k for topic domain j. In other words, it is how strongly the document belongs in a given topic. This is calculated by summing the product of the conditional probability and tag term weight for all tags attached to website k. This is calculated as shown in equation 2.

$$dw_{j,k} = \sum_{i=0}^{n} P(t_i|D_j) \cdot tf_{i,k} \cdot idf_i \tag{2}$$

In this case, n is the number of distinct tags attached to website k by all users. $P(t_i|D_j)$ is the conditional probability of tag term i (shown as t_i) given domain j (shown as D_j). $tf_{i,k} \cdot idf_i$ is as calculated by standard tf-idf, where $tf_{i,k}$ is the number of times any user bookmarked a website k with tag i divided by the total number of times that website is bookmarked. Similar the idf_i is determined by the total number of websites in the entire set divided by the number of websites bookmarked with tag i. So, instead of the website vector being described in terms of tag term weights, a website's domain vector parameters are now its topic domain weight value, i.e. how much the document belongs to that particular topic domain. An example calculation is shown in figure 3.

In this case, apple.com has three tags attached by all users: 'apple', 'mac', and 'software'. Using the conditional probabilities shown in figure 2, the calculations of domain 1 and 2 are shown.

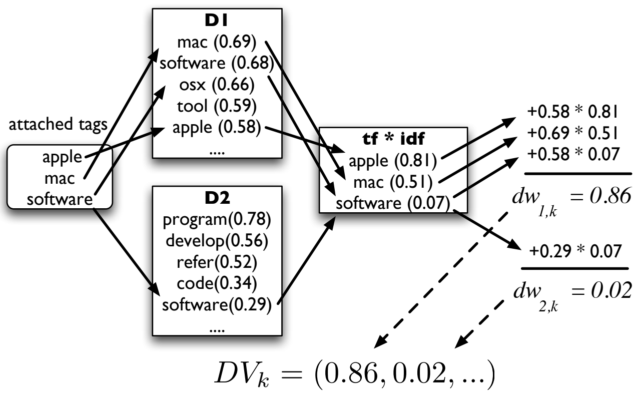


Figure 3 Calculation of k = apple.com's website topic domain vector

Similarly, a bookmark topic domain vector created from user A bookmarking website k, $DV_{A \to k}$, would be calculated in the same fashion as shown in equation 3.

$$DV_{A \to k} = (dw_{1,A \to k}, dw_{2,A \to k}, dw_{3,A \to k}, ...dw_{100,A \to k}) \tag{3}$$

The only exception is that the domain tag weight only considers the tags that user A used on website k as shown in equation 4.

$$dw_{j,A \to k} = \sum_{i=0}^{n} P(t_i|D_j) \cdot idf_{i,A \to k} \tag{4}$$

Since a user can only apply atmost one of the same tag term to a bookmark, only the idf is used here.

3.2 RCF Recommendations

We now describe our Reasonable Tag-based Collaborative Filtering (RCF) algorithm. The process to generate recommendations follows these steps:

(1) Finding similar users.

(2) Finding recommendation candidates.

(3) Providing live-updating recommendations based upon the user's current interest.

We now describe these steps.

3.2.1 Finding Similar Users

We start off by finding similar users. Like normal collaborative filtering, it is based upon commonly liked items–in this case, commonly bookmarked websites. However, it differs in that we also consider the used tags–which we assume to be the reason why a user liked a resource. For example, from our previously shown system in figure 1, say we want to find the similar users of user B. It would be calculated as shown in figure 4.

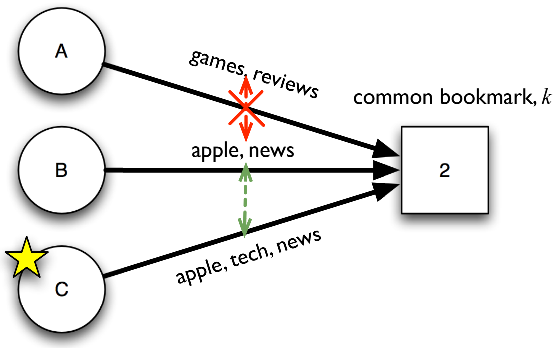


Figure 4 Example User Similarity Calculation

Since user B and C have bookmarked the same website with similar tags–indicated by the dotted arrow–their similarity score is higher. In this case, C becomes a similar user, indicated by the star. On the other hand, users B and A have bookmarked the same site, but they do not use similar tags–indicated by the crossed-out dotted arrow–and thus their similarity score is lower.

User similarity for a user A and a user B is calculated as shown in equation 5.

$$sim_{rcf}(A, B) = \alpha \cdot \frac{1}{n} \sum_{1}^{n} \{sim(DV_{A \to k}, DV_{B \to k})\} + (1 - \alpha) \cdot log_2(1 + n) \quad (5)$$

Here, user similarity is the average of the cosine of A and B's domain vectors for each commonly bookmarked website, k. In other words, if they bookmarked the same websites with similar tags, we assume that they liked the website for the same reasons, and thus, they are similar users. The first half of the equation gives value to bookmarks liked for the same reason. The second half of the equation gives value to the existence of common bookmarks regardless of the reasons for liking it.

Also, n is the number of bookmarks that user A and user B have in common. Lastly, α is the weight given to topic domain vector comparison, which for our experiments was set to 0.9. Additionally, $sim(DV_{A \to k}, DV_{B \to k})$ is the cosine of user A's bookmark domain vector on website k and user B's bookmark domain vector on a website k as shown in equation 6.

$$sim(DV_{A \to k}, DV_{B \to k}) = \frac{DV_{A \to k} \cdot DV_{A \to k}}{|DV_{A \to k}||DV_{B \to k}|} \quad (6)$$

As can be seen, our algorithm tries to make sure that the users have like the same resource for the same reason in addition to having only the same website liked. This is useful for mainly larger sites, such as Yahoo! or Slashdot which tend to cover many topics and thus, users may like those sites for differing reasons.

3. 2. 2 Finding Recommendation Candidates

Next, we try to find recommendation candidates for a user through score prediction calculation, i.e. predicting how much a user will like a resource. Again, similar to traditional collaborative filtering, we attempt to find websites that the similar users like, except now we also additionally match the topic domain that the original user and the similar user have a common interest on. The thinking being that users often have multiple interests, and therefore we can only recommend websites that match the topic that the two users were similar on.

For example, in the system shown in figure 1, we try to find recommendation candidates for user B who has a similar user C as shown in figure 5. They both liked website 2 for the reason that the website is about apple technology news. Thus, we try to find websites that match this topic. In this case, similar user C has two other bookmarks. Since only website 3's domain vector matches the commonly bookmarked website 2's domain vector–indicated by the dotted arrow–it has a high score prediction, and thus, becomes a recommendation candidate (denoted by a star). On the other hand, website 4 does not match–marked by the crossed-out dotted line–and thus, its score prediction is lower.

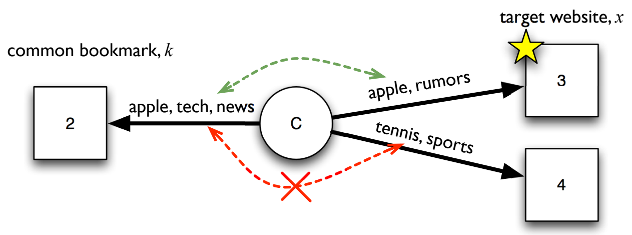


Figure 5 Example score prediction calculation

The score prediction algorithm is shown in equation 7. Here, we are finding the score prediction for a target website x for a user A. We first take all bookmarks from all similar users S_k with a user similarity above a certain user similarity threshold–in our case, 0.75. We then compare each of those bookmark domain vector $DV_{S_k \to x}$ with the bookmark domain vector on the commonly bookmarked website k, $DV_{S_k \to k}$. We average these scores from all users and then those webpages with scores above a certain threshold–0.50 in our tests–become recommendation candidates. Additionally, we associate this recommendation to the commonly bookmarked webpage k whose domain vector generated the score. This is used in the final score calculation described in following section.

Also, α is the weight given to the domain vector comparison, and in this case, it is 0.90. Additionally, while the left side attempts to match the topic, the right side of the equation is to give value to the mere existence of a similar user's bookmark regardless of the matching tags.

$$score_{pred}(A,x) = \alpha \cdot \frac{\sum_{k=1}^{n}\{sim_{rcf}(A,S_k) \cdot sim(DV_{S_k \to k}, DV_{S_k \to x})\}}{\sum_{k=1}^{n}\{sim_{rcf}(A,S_k)\}} + (1-\alpha) \cdot log_2(1 + \sum_{k=1}^{n}\{sim_{rcf}(A,S_k)\} \qquad (7)$$

3.2.3 Providing Live-Updating Recommendations Based Upon User's Current Interest

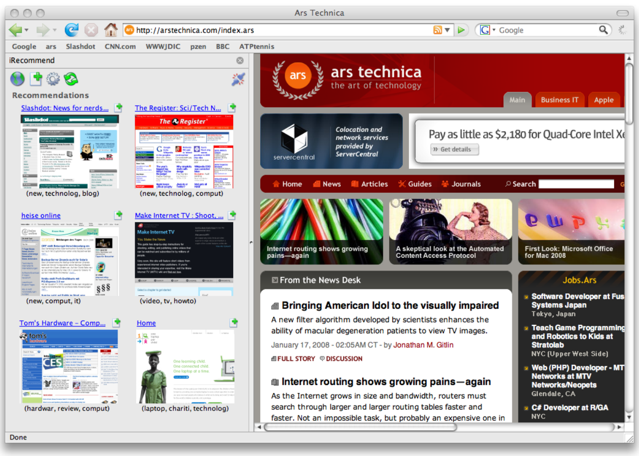
The last step in the process of generating recommendations is determining which of the recommendation candidates match the current interest. In our case, we assume the user's current interest to be the topic of the website they are presently viewing. So again from our example system, we want to provide recommendations for user B as shown in figure 6.

Here, B is looking at a site which has the tags 'apple', 'mac', and 'software' attached to it by bookmarks from all users. B also has two recommendation candidates, website 3 and 6 (6 is an arbitary webpage not shown in the example system from figure 1) which were generated from section 3.2.2. Website 3 has been associated with commonly bookmarked website 2 and website 6 has been with website 5. We then compare the user's currently viewed webpage's domain vector to all of B's bookmarks. If we find one that is above a certain threshold, then we take the recommendation candidates that are attached to it and calculate its final score as shown in equation 8. All websites above a certain threshold are then recommended to the end user. So in this case, since B bookmark domain vector matches the current website's domain vector, its attached website, 3, is used for final score calculation. If its score is high enough, it is recommended. B also has an arbitrary recommendation candidate, website 6. However, since B's bookmark on website 5 does not match the currently viewed website, it is not recommended at this time. If the current website were to change to something about sports or tennis, website 6 would be recommended if its final score was high enough.

In equation 8, user A is viewing webpage cur, k is the a commonly bookmarked website, and x is a recommendation candidate. Lastly, β is the weight given to the part of the equation which determines how similar the recommendation candidate should be to the current page. For our experiments, this was set to 1. All resultant final scores higher than a threshold of 0.50 are shown to the user in the descending order of the score.

3.3 Live-Updating Website Recommendation

Our system was created to provide live-updating website recommendations that automatically update based on the page that they are currently viewing. Thus, it was designed to be viewable along during the user's normal browsing activity. It is currently implemented as a Firefox sidebar plugin. The recommendation interface is show in figure 7. Here, the generated recommendations are shown in the sidebar. If they feel the recommendations are useful, they can click on it and the browser will be redirected to the recommended website.


Ars Technica
http://arstechnica.com/index.ars
Google
Google ars Slashdot CNN.com WWWJDIC pzen BBC ATPtennis
iRecommend
Recommendations
Slashdot: News for nerds...
(new, technolog, blog)
The Register: Sci/Tech N...
(new, technolog, comput)
heise online
(new, comput, it)
Make Internet TV : Shoot, ...
(video, tv, howto)
Tom's Hardware – Comp...
(hardwar, review, comput)
Home
(laptop, chariti, technolog)
ars technica
the art of technology
Main
Business IT
Apple
Colocation and network services provided by ServerCentral
servercentral
Pay as little as $2,180 for Quad-Core Intel Xe
Get details
Home
News
Articles
Guides
Journals
Search
Internet routing shows growing pains—again
A skeptical look at the Automated Content Access Protocol
First Look: Microsoft Office for Mac 2008
From the News Desk
Jobs.Ars
Software Developer at Fus Systems Japan
Tokyo, Japan
Teach Game Programming and Robotics to Kids at Stratolab
NYC (Upper West Side)
Web (PHP) Developer - MT Networks at MTV Networks/Neopets
Glendale, CA
C# Developer at R/GA
NYC
FULL STORY
DISCUSSION
Done

Bringing American Idol to the visually impaired

A new filter algorithm developed by scientists enhances the ability of macular degeneration patients to view TV images.

January 17, 2008 - 02:05AM CT - by Jonathan M. Gitlin

Internet routing shows growing pains—again

As the Internet grows in size and bandwidth, routers must search through larger and larger routing tables faster and faster. Not an impossible task, but probably an expensive one in

Figure 7 System Interface

4. Experiment

We tested our algorithm against other popular search methods to gauge the effectiveness of the algorithm as well as observe how users react to generated recommendations. We tested our method as described in the previous section as well as three other different methods:

- **topic** - This method takes the top five domains from the currently viewed webpage and then pulls all the bookmarks from the database with high values for those domains. All website above a cosine value of 0.75 are then ranked by the same cosine value and shown to the user
- **tag-bookmark-count (tbc)** - This method takes the top three stemmed tags from the currently viewed webpage and then pulls all bookmarks from the database with these tags. The results are then ranked by the number of bookmarks in descending order.
- **cf-topic** - All bookmarks of all users with common bookmarks with the current user were stored as recommendation candidates. Then we take the currently viewed website's domain vector and those stored bookmarks' domain vectors and calculate the cosine. They are then ranked by cosine value. This has similarity to our method, but it does not consider tagging when calculating the score predictions of the recommendation candidates

4.1 Procedure

Users were asked to create a profile of twenty or more bookmarks with whatever tags they felt like. After this, the system would generate recommendation candidates as described in section 3.2. After this, they followed this recommendation evaluation procedure:

(1) Users would select fifteen or more webpages and browse them.

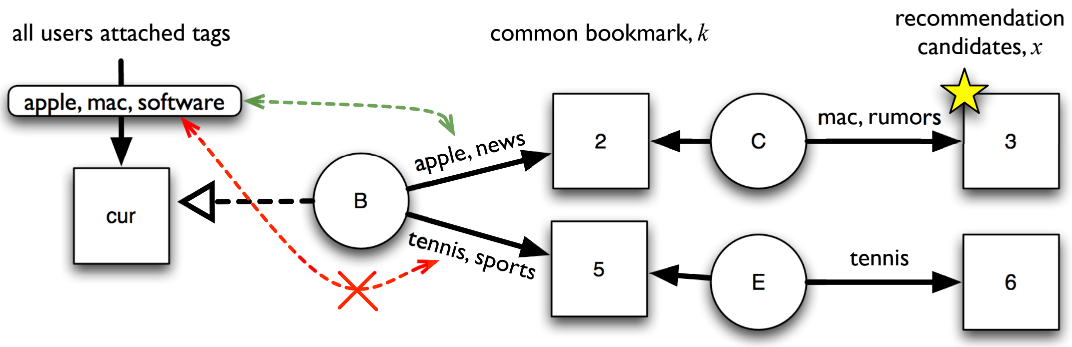


Figure 6 Example Final Score Calculation

$$score_{final}(A, cur, x) = sim(DV_{cur}, DV_{A \to k}) \cdot score_{pred}(DV, x) \cdot sim(DV_{cur}, DV_{A \to x})^{\beta} \tag{8}$$

(2) The system would randomly select one of the above described algorithms and generate up to six recommendations from it.

(3) Users would then look at the website's screenshot thumbnail, top tags, and title. If they thought the page was interesting and related to the page they were browsing, they would click on it.

(4) If they did click on a recommendation, they were asked to evaluate the webpage as either 'Good', 'Fair', or 'Bad' in terms of how interesting the page was and also whether the website was related to the page that they were currently viewing.

4.2 Testing Results

Here are the results of our experiment.

4.2.1 How effective are the recommendations?

Next we examine how effective each method was for recommendation. Figure 8 shows the precision of each of the recommendation methods.

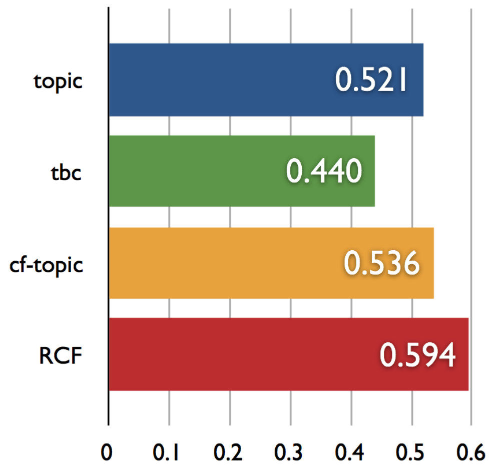


Figure 8 Precision (Good / Total)

Precision is defined as the number of 'Good' ratings over the total number of ratings. Here, each method's precision is shown. As can be seen here our method RCF, provides the highest level among all the recommendation methods. It is followed by cf-topic, topic, and then finally tbc.

Additionally, we also take a look at the average ratings for the top scores per recommendation request as shown in figure 9.

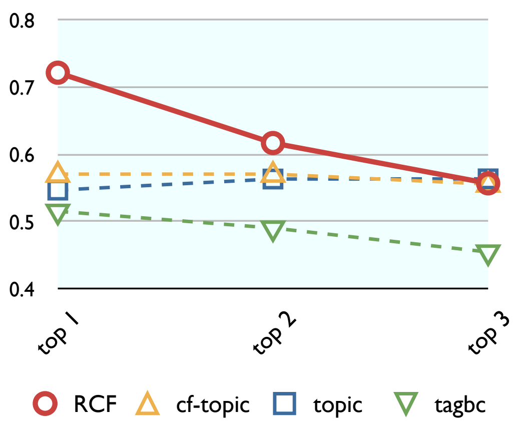


Figure 9 Average Score of the Top Results per Request

Here, we have the average rating the top listed result, followed by the top two, and finally top three results per recommendation request. The ratings were given the following values: 'Good' = 1, 'Fair' = 0.5, and 'Poor' = 0. Again, RCF beats all the other methods and more so when considering the top ranked result per request.

4.3 Results Discussion

Looking at the precision data, our RCF method provides the most effective recommendations to the user in terms of the precision of the recommendation results. This is due to that RCF attempts to understand the reasons involved when a user bookmarks a website. Additionally, considering that users have multiple interests, it is important to understand what they are currently interested in by using their presently viewed webpage as a guide. In contrast to this, the other methods do not do this.

Topic searching is in effect similar to what RCF does in the final score step. It considers the current page to some extent. However, since topic search does not consider the user bookmarking preferences, it most likely causes the score to lower.

For tbc, it's strength lies in the fact that it takes website popularity into consideration as its ranking method orders by bookmark count. However, again, since this too does not consider the user's bookmarking profile, it is rendered less effective than others.

Lastly, cf-topic is a rather basic implementation of collaborative filtering; but being so, it does consider the user's profile to some degree. However, it does not effectively understand the reasons why a user liked something as it does not consider the tagging information. Being so, it probably causes its score to be lower. However, as can be seen in the results, it is more effective than the methods that do not.

Regarding the top results' values, we can see that our method provides better recommendations for the top ranking results. This is important as it requires less effort on behalf of the user in finding relevant results. Additionally, in recommendation systems, there is simply a limit on how many recommendations you can provide to the end user. Thus, it is important to lower the number of recommendations, while at the same time, maintain a high level of precision. Anything less would reduce the usefulness of a recommendation system.

5. Conclusions and Future Work

We have described our Reasonable Tag-Based Collaborative Filtering method for use in social tagging systems. Additionally, we have described its place in our live-updating website recommendation system which provides recommendations based upon both the reasons they used for bookmarks and their current interest. Through our user testing, we have shown that live-updating implicit recommendation is possible through the use of a website's tags. Our method was more effective than other tested methods, and additionally provided higher results for the top results per recommendation request.

We believe it is an effective method that can be employed in all sorts of online tagging systems. Social tagging systems and CF rely heavily on the people using this systems, so combining them–such as done in our method–seems like the next logical step.

In the future, we plan to further refine the recommendation method. One issue is when a currently viewed webpage is not in the database. Since our algorithm is based solely upon bookmarking data, if the website is not in the database, the system cannot make recommendations. This is especially true for websites with continuously updating webpages such as newsites or blogs. A basic fallback method based on websites contents as well as other standard search methods could be implemented.

Additionally, we are evaluating what the optimal threshold is for our system–one that finds the balance between giving good recommendations, versus giving enough number of recommendations for the user to choose from.

Lastly, we also intend to add relevance feedback into our method. Currently, it is based solely on bookmarks. However, it would make sense to incorporate both positive or negative feedback of the recommendations generated so that we can provide better recommendations to the user.

Additionally, we plan to add recommendations grouping, where as the results are group by topic. This way, the user can then have a better idea of which aspect of the website they are viewing as well as be able to choose what topic that they wish to see more of.

We also plan to do wider testing to gather data in more uncontrolled environments in order to determine users' actual usage patterns. Many users are usually not confident opening pages that they do not understand how it relates to them. Thus, we must determine how to get over this barrier and at the same time provide effective recommendations.

Acknowledgments

This research was partly supported by MEXT (Grant-in-Aid for Scientific Research on Priority Areas #19024058).

References

[1] Amazon.com. http://www.amazon.com/.

[2] Citeulike. http://www.citeulike.org/.

[3] Consumating.com. http://www.consumating.com/.

[4] del.icio.us. http://del.icio.us/.

[5] Flickr. http://www.flickr.com/.

[6] http://www.tartarus.org/ martin/porterstemmer/.

[7] movielens. http://www.movielens.umn.edu/.

[8] A. P. Dempster, N. M. Laird, and D. B. Rubin. *Maximum likelihood from Incomplete Data via the EM Algorithm*, volume B 39. J. Royal Stat. Society, 1977.

[9] S. Golder and B. Huberman. The structure of collaborative tagging systems.

[10] D. K. J. Becker. Topic-based vector space model. In *Business Information Systems*, 2003.

[11] C. Marlow, M. Naarman, D. Boyd, and M. Davis. Position paper, tagging, taxonomy, flickr, article, toread. 2006.

[12] R. Nakamoto, S. Nakajima, J. Miyazaki, and H. K. S. Uemura. Evaluation of tag-based contextual collaborative filtering effectiveness in website recommendation. Technical Report 131, IEICE, 2007.

[13] R. Nakamoto, S. Nakajima, J. Miyazaki, and S. Uemura. Tag-based contextual collaborative filtering. In *18th IEICE Data Engineering Workshop*, 2007.

[14] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM.

[15] X. Wu, L. Zhang, and Y. Yu. Exploring social annotations for the semantic web. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 417–426, New York, NY, USA, 2006. ACM.