Web Site Recommendation Using HTTP Traffic

Ming Jia* jiaming@iastate.edu
Shaozhi Ye† sye@ucdavis.edu
Xing Li‡ xing@cernet.edu.cn
Julie Dickerson* julied@iastate.edu

∗ Dept. of ECE
Iowa State University
Ames, Iowa 50011

† Dept. of CS
University of California, Davis
Davis, CA 95616

‡ Dept. of EE
Tsinghua University
Beijing, China 100084

Abstract

Collaborative Filtering (CF) is widely used in web recommender systems, while most existing CF applications focus on transactions or page views within a single site. In this paper, we build a recommender system prototype, which suggests web sites to users, by collecting browsing events at routers without neither user nor website effort. 100 *million HTTP flows, involving* $11,327$ *websites, are converted to user-site ratings using access frequency as the implicit rating metric. With this rating dataset, we evaluate six CF algorithms including one proposed algorithm based on IP address locality. Our experiments show that the recommendation from K nearest neighbors (*R_{kNN}*) performs the best by* 50% $p@10$ *(precision of top* 10*) and* 53% $p@5$ *(precision of top* 5*). Although the precision is far from ideal, our preliminary results suggest the potential value of such a centralized web site recommender system.*

1. Introduction

Collaborative Filtering (CF) is extensively used in recommender systems, especially for E-commerce web sites, which make predictions based on the similarities among user activities. The strong correlations among user ratings to the resources (items or web pages) within the same site make CF successful in singe site recommender systems. Due to the data collection problem, however, there is little work done on applying CF to coarser granularities, such as web site recommendations, whose precision and scalability in large scale systems has not been carefully investigated.

In this paper, as a proof of concept, we develop a web site recommender system prototype which treats HTTP traffic as user preference to web sites. More specifically, the system works as follows. First an ISP (Internet Service Provider) collects HTTP traffic to feed the recommender system. The system extracts users' browsing traces and maps them to user ratings for web sites. Then the CF algorithm generates recommendations for each user based on his/her own browsing history and its similarity to other users' browsing history. This service can be provided as a web portal for users with opt-in subscription.

The main reason we only provide site grained recommendation is to reduce the data collecting/processing cost. To get page view semantics within an HTTP packet, we have to check the payload besides the TCP header, which is expensive for today's high bandwidth network. Moreover, page views also have more threats on user privacy than the aggregated "site views." We believe that in spite of its simplicity, this prototype system gives us valuable insight for using web traffic to do web page/site recommendation.

Site grained recommendation also has some benefits in terms of recommendation. First, a web site can be viewed as a portal to a collection of web pages, which makes the recommendation results compact and thus easier and more efficient for users to view. Secondly, web sites are more stable than individual web pages. A web search user usually wants a single web page which specifically answers his/her query while our recommender system does not pursue this. Instead, it tries to complement the search systems we have today in such a way that the user can find new resources in a passive fashion (without issuing queries).

After converting 100 million HTTP flows into user-site ratings, we evaluate the following six algorithms: three representative CF algorithms (user based, clustering based, and item based), one proposed CF algorithm based on client IP locality, and two baseline approaches (random and global popularity based). Our experimental results show that the user based R_{kNN} (recommendations from k nearest neighbors) performs the best with 50% $p@10$ (precision of top 10) and 53% $p@5$ (precision of top 5). As an unexpected result, the clustering based algorithm achieves close precision as R_{kNN}, which indicates that our system is able to scale with larger datasets. The recommendation precision from our preliminary experiment without carefully tuning

suggests the practical value of our system.

We believe that this paper makes the following three contributions.

- Provide a prototype for a web site recommender system, which require neither user nor server side involvement for data collection.
- Convert HTTP traffic to user-site rating data, which provides a new resource for web recommender systems.
- Conducted an experimental evaluation of six CF algorithms on web site recommender system.

The rest of this paper is organized as follows. Section 2 briefly reviews CF algorithms. Section 3 describes the data collection and conversion procedures and Section 4 introduces the six CF algorithms evaluated in this paper. Experimental results are shown in Section 5. After exploring the future work and possible solutions for similar systems without ISP involvement in Section 6, we conclude the whole paper with Section 7.

2. Collaborative Filtering

CF has been developed and improved over the past decade to the point where a wide variety of algorithms exist for generating recommendations. The CF algorithms can be roughly divided into two categories: memory-based and model-based.

The memory-based algorithms select the top K similar users for an active user with the entire training dataset, then combine those rating together to generate recommendations. Notable examples include vector similarity [4], Pearson correlation [7], and the extended generalized vector-space model [10]. Memory-based algorithms suffer from data sparsity and scalability problems [12].

The model-based algorithms first develop a model of user ratings based on the training dataset, and then use the model to predict user preference to the items in testing set. Well-known model-based algorithms include Bayesian network [2], clustering-based model [11], rule-based approach [9], and item-based approach [8]. Although model-based algorithms often require a lot of time to train the model, the training process can be done offline, and the online recommendations for active users are fast. The item-based method, which performs best when the number of items is much smaller than the number of users, achieves great success in E-commerce web sites, such as Amazon.com[5].

3. Data Collecting and Processing

3.1. Raw Data

The data set used in this paper is collected from a major PoP (point-of-presence) of China Education and Research Network (CERNET), which has 21 million users and 38 PoPs. During a 24-hour (April 2nd, 2006) period, we collected 100 million HTTP flows. A flow is an undirected sequence of TCP packets all sharing the same 5-tuple of source IP, destination IP, source port, destination port, and IP protocol (HTTP here). A flow is considered to be *finished* when one of the following happens: timeout (inactive), TCP reset, or exceeding a fixed time interval (active).

Shown as Fig. 1, the distribution of site-aggregated flows follows power-law, which is consistent with previous observations [6].

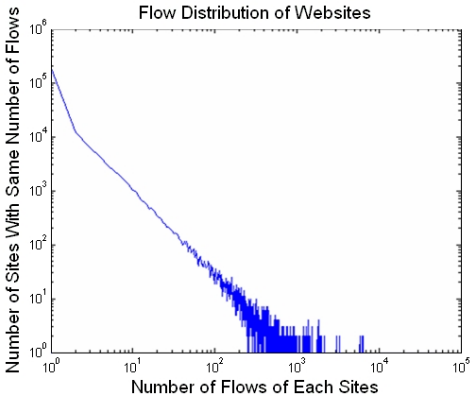

Figure 1. Website Flows: Power Law Distribution

Since the web traffic via port 80 consists of the majority of entire web traffic, in this paper we only consider HTTP traffic via port 80 at the server side.

3.2. Impact of Packet Sampling

Due to the large amount of network traffic across the router, the packets used to construct flow records are randomly sampled at the ratio of 1% (default setting of Cisco Netflow), i.e., one packet out of every 100 packets are used. Fortunately the actual loss rate caused by sampling is far less than 99%.

1. Since most flows contain multiple packets, *the loss rate of flows is much less than that of packets*. For example, Duffield et al. [3] shows that for the HTTP traffic the loss of flow records is 90% with 1% random sampling. Thus we get about 10% flows.

2. When a user visits a webpage having multiple components, e.g., icons, pictures, and videos, modern browsers usually automatically open several other ports (at the client side) to download these components in parallel. Then accessing one web page may result in several flows with the same source and destination IPs, but different client ports. Hence missing some of these flow records may not lose the corresponding page view event. This phenomenon implies that *the loss rate of page views is less than that of flows*. Assuming accessing one web page generates n HTTP flows on average, the portion of *page views* we capture is $1 - 0.9^n$.

3. Moreover, a user may browse several web pages on the same web site during a visit, thus *the loss rate of site visiting records is less than that of page views*. Again, assuming the average page views per visit is p, the portion of *site visiting events* we capture is $1 - 0.9^{np}$.

For example, when $n = 3$ and $p = 2$, the captured site visiting events is 46.9%. Thus it is reasonable to believe that although only 1% packets are sampled, a large portion of site accessing information can be preserved. To verify the real sampling rate, we can conduct similar experiments as [3], which is out of the scope of this paper.

Due to the sampling, the recorded flows are biased by the flows with more packets, and also biased by the visit with more page views. We believe that this kind of bias results in more stable recommendation performance.

3.3. Web User and Site Identification

Given an HTTP flow, we treat the host with port 80 as the web server, and the other host as the client. Usually web sites have static IP addresses or at least their IP addresses are unlikely to change within 24 hours. Thus it is safe to use IP addresses to identify web sites. Because of the IP assignment policy of CERNET [1], most DHCP users get a lease for at least 24 hours, in other words, most users have the same IP addresses for at least 24 hours, which is also the time range of our data collection. Hence it is also reasonable to identify a user with an IP address.

However, several web sites may share the same IP address, e.g., virtual hosts. On the other hand, a large web site may use multiple web servers for load balance purpose. Similarly, one IP address may be shared by several users, e.g., via a proxy, and a user may also occupy several IP addresses. These factors combined together result in a complicated impact to our recommender system.

- Multiple sites share one IP: A *mixed* web site fails the assumption that all the accesses to one site are correlated, thus hurts the recommendation precision.

- One site has multiple IPs: A site is broken into several smaller sites, and the accesses to the same site are treated as accesses to different sites. On one hand, we lose the correlations among these accesses, which hurts our precision. On the other hand, since these smaller sites are correlated, they boost the overall similarity within the whole site pool, where we choose candidates for recommendation, thus improve the precision.

- Multiple users share one IP: Its effect is similar to the case where multiple sites share one IP.

- One user has multiple IPs: Its effect is similar to the case where one site has multiple IPs.

We eliminate Case 2 by downloading and comparing the homepages corresponding to each IP. A popular reason for Case 4 is that a user has different IPs at work and home. Considering the different tasks performed with each IP, Case 4 is not so harmful compared to other cases. Both Case 1 and 3 hurt the precision of our system, thus the results presented in the following sections can be further improved if these two cases are factored out.

3.4. Converting Flows to Ratings

We consider the flow between client c and server s as the implicit rating of s given by c. A straightforward rating method is binary rating, which represents whether or not a client visits a server within the observed time window. Specifically, let $R(c, s)$ denote the binary rating from client c to server s, then we have:

$$R(c,s) = \begin{cases} 1 & \text{if exists at least one flow between } c \text{ and } s \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

This is the simplest method to count the user preference to web sites. A zero rating does not necessarily mean that the client dislikes the corresponding web site. In fact it is possible that the user is not aware of this site. In this sense, binary rating is similar to unary rating [4].

The observations in both [6] and Fig. 1 indicate that the rating among users or web sites follows the power-law. In fact, for most individual users, the distribution of their flows among web sites also follows power-law. Since users have strong preference to the web sites they visit, quantitative ratings (e.g., setting the number of flows or packets as the value of the rating) may be better than the binary rating. Here we choose the simplest binary rating to present the preliminary results. A more comprehensive study is underway.

[1] To our knowledge, most other ISPs have similar policies.

4. Collaborative Filtering Algorithms

In this section, we describe the six CF algorithms evaluated in this paper, random, global popularity, R_{kNN} [4], clustering based CF [7], item-based CF [8], and nearest IP.

4.1. Baselines

The first baseline algorithm is random recommendation, which just randomly selects N web sites, and recommends them to the user. It shows the similarity of the whole site pool, i.e., if most sites are similar, the results from random recommendation will be good.

The second baseline, global popularity recommendation, recommends N web sites having top N popularities (number of visitors) among all the users. This method is widely used when no particular user preference is known, e.g., best seller, most viewed items.

4.2. R_{kNN}

R_{kNN} selects the K (30 in our experiment) neighbors (users) who are most similar to the active user c and then recommends N web sites having top N popularities (number of visitors) among these K neighbors.

We use the basic vector cosine similarity, i.e., the number of sites visited by both users divided by the number of sites visited by each of them, to select neighbors.

4.3. Clustering-based Algorithm

Clustering-based algorithm first clusters the users into groups. Given an active user c, the algorithm finds the group $g(c)$ which c belongs to, and recommends N web sites having top N popularities among $g(c)$. This method is similar to R_{kNN} but does not need to compute neighbors for each user on line and is, thus, faster and more scalable.

We use k-means to group users, where k is set to 60 based on our experience.

4.4. Item-based Algorithm

The item-based method [8] first calculates the similarity between every pair of items (web sites), and then ranks web site s according to the number of users who have visited the web sites similar to s. The overall rating is a summation weighted by the cosine similarity between two sites and scaled by the overall similarity s has with other sites.

Since storing all the similarities of item pairs requires $O(N^2)$ space for N items, Sarwar et al. [8] utilize an approximate method by only considering a small portion of the items with highest similarities for each item. We also use this method and get the same trend as [8]. We keep top 20 similar sites for each site in the following experiments.

4.5. Nearest IP Algorithm

Although ISPs assign IP addresses randomly for DHCP users, usually two users with short geographic distance also get close IP addresses (in terms of shared prefix length). For example, people in the same building with the same ISP may share the same subnet. Here we try to evaluate whether IP locality can be used as a strong tie in CF. Our proposed nearest IP (NIP) algorithm simply chooses K users with nearest IPs (longest shared IP prefixes) as neighbors for the active user. The recommendation step is then the same as that of R_{kNN}. We set the number of neighbors to 30 for fair comparison with R_{kNN}.

5. Experiment

5.1. Evaluation Metric

There are many metrics used in previous recommender systems. Given a certain algorithm, the user task and the data set determine the proper metrics.

Our goal is to provide the user with a list of web sites which he/she may be interested in. Considering usability, this list can not be too long. Thus the Mean Absolute Error (MAE) does not make much sense here, because an algorithm, which is good at predicting irrelevant items, but poor at recommending relevant items, may also achieve low MAE. We choose the top N precision metric [1] while we believe that similar conclusions can be attained through other metrics, such as precision combined with recall and $F1$ in [8].

Without user surveys, it is hard to evaluate whether a user is indeed satisfied with or interested in a recommendation result given by our system. To avoid the cost for large number of user studies (in our case, $17,000$ users), however, some heuristics can be applied to infer the actual user satisfaction. Here, we assume that if a user visits a site, he/she is interested in it. More specifically, for each user c, the algorithm recommends N web sites, which have not been visited by c in the training set. If c visits L of them in the testing set, the top N precision is calculated as $P = L/N$. In other words, we use the training set for prediction and the testing set for evaluation.

5.2. Training/Testing Datasets Split

After removing the clients and servers which appear only once, the rating dataset contains totally $493,908$ binary ratings, involving $17,697$ clients and $11,327$ sites, with a sparsity level [8] of 99.75%. We split this data set into a training set and a testing set with a size (number of ratings) ratio $5:8$. As a comparison, in [8], the ratio is $8:2$. When splitting the ratings, we keep the flow ordering such that any

flow in the training set has an earlier time stamp than any flow in the testing set.

5.3. Results

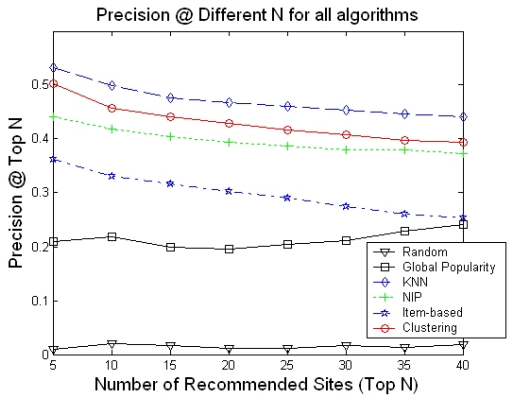

Figure 2. Comparison of Six CF Algorithms

The results of all six CF algorithms are shown as Fig. 2. The X-axis represents the number of recommendations the algorithm gives, and the Y-axis represents the average top N precision. The findings from our experiments are summarized as follows.

- The R_{kNN} algorithm performs the best with 53% $p@5$ and 50% $p@10$. Since the rating distribution follows power-law, the high precision also verifies the statement in [4] that "systems where there is an exponential popularity curve may be able to find agreement among people in the dense sub-region and use that agreement to recommend in the sparse space."

- With different N, the precision order remains the same: R_{kNN} > Clustering > NIP > Item-based > Popularity $\gg$ Random.

- All other algorithms perform better than the baselines. This result indicates that using per-user history can greatly improve the recommendation precision.

- The clustering-based algorithm ranks the second with precision close to R_{kNN}. Clustering algorithms which consider the similarity among users may get better results than the simple k-mean we use here. This unexpected result shows that our system is able to deal with larger datasets by clustering when keeping relative high precision.

- NIP algorithm takes the third place, which suggests that IP locality does represent user similarity to some degree.

- The precision of random recommendation is barely above zero, which shows that the overall similarity within our data set is low.

- Except the popularity algorithm, the precision of other methods decreases with increasing N. Item-based method drops to the same level as popularity method when N reaches 40. This fact indicates that the users do have stronger preference to some sites, otherwise the precision would not drop.

- As N increases, the precision decreasing rate slows down. A possible explanation is that there is much less difference of the user preference among the sites ranked between 20 and 40. The skewed user preference distribution suggests a shorter recommendation list.

The ordering of algorithm precision from our experiment is different from [8], which claims that the item-based algorithm performs better than R_{kNN}. The possible reasons are as follows:

- *Rating metric*: Our experiment uses binary rating while the work in [8] uses a quantitative rating (integer values from 1 to 5). Binary rating results in a bias to the sites with high overall ratings, which also explains why the precision of item-based algorithm in our experiment is close to the popularity algorithm.

- *Accuracy metric*: The accuracy metric used in [8] is MAE.

- *Dataset*: In [8], movie ratings from MovieLens recommendation system [2] is used. The movies can be easily classified into several categories, and each movie usually only belongs to one or two categories. In our dataset, however, the web pages in one site may belong to multiple categories. Such ambiguity in our dataset increases the error rate for CF algorithms, which may hurt different algorithms non-uniformly.

6. Future Work

Currently we are conducting more comprehensive studies with our system, including:

- *Weighted user rating*: Binary rating loses the site visiting frequency thus makes the similarity between two users inaccurate. A weighted rating, which considers the number of flows or data during one site visit, will help to select better neighbors (in terms of similarity) thus get higher recommendation precisions.

[2] http://movielens.umn.edu

- *Better clustering algorithm*: As mentioned before, a better clustering algorithm which takes into account the distribution of users may greatly outperform the basic k-means clustering.

- *User studies*: We are trying to conduct some user studies to further evaluate the recommendation performance on an experimental service with opt-in subscription.

- *HTTP header*: In this paper, we only check the TCP headers for flow information due to performance and privacy concerns. With sampling and anonymization, we may be able to use the HTTP headers. By using HTTP headers, especially the requested URLs and referrers [3], we can correlate user events better and thus improve the user rating and recommendation precision.

To get cross-site HTTP traffic, there are also solutions besides cooperating with ISPs.

- Link exchange has been widely used for web site propagation, which can be easily used for exchanging user accesses among web sites. Google AdSense [4] actually uses a similar approach.

- DNS and CDN (Content Delivery Network) systems also track the browsing events to some extent, thus are able to provide the data for centralized cross-site recommender systems.

- OpenID [5] keeps a unique ID for each user on multiple web sites. A similar idea can be applied for tracing user events across sites.

- A browser toolbar similar to that from Alexa [6] is also an option.

7. Conclusion

In this paper, we propose a web site recommender system with centralized HTTP traffic collection, which does not require any effort from the user or server side. We evaluate six CF algorithms including one proposed algorithm based on IP address locality. R_{kNN} performs the best, which achieves 53% $p@5$ and 50% $p@10$. The clustering-based algorithm gets close precision as R_{kNN}, which means our system has the potential to scale with larger datasets. Despite the simplified settings in this prototype and the far from ideal 50% recommendation precision, our preliminary results indicate the promising practical value of our system.

With the discussion of our undergoing research and possible solutions for similar systems without ISP support, we hope that our work encourages wide deployment of web site recommender systems.

[3] A referrer is a field in an HTTP header which indicates the page leading the user to the requested URL.

[4] http://www.google.com/adsense

[5] http://openid.net/

[6] http://www.alexa.com

References

[1] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: using social and content-based information in recommendation. In *AAAI'98/IAAI'98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 714–720, 1998.

[2] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[3] N. Duffield, C. Lund, and M. Thorup. Properties and prediction of flow statistics from sampled packet streams. In *IMW'02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement*, pages 159–171, 2002.

[4] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.

[5] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 07(1):76–80, 2003.

[6] M. Meiss, F. Menczer, and A. Vespignani. On the lack of typical behavior in the global web traffic network. In *WWW'05: Proceedings of the 14th international conference on World Wide Web*, pages 510–518, 2005.

[7] P. Resnick and H. R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, 1997.

[8] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW'01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.

[9] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *EC'00: Proceedings of the 2nd ACM conference on Electronic commerce*, pages 158–167, 2000.

[10] I. Soboroff and C. Nicholas. Collaborative filtering and the generalized vector space model (poster session). In *SIGIR'00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 351–353, 2000.

[11] L. Ungar and D. Foster. Clustering methods for collaborative filtering. In *AAAI Workshop on Recommendation Systems*, 1998.

[12] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In *SIGIR'05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 114–121, 2005.