Personalized Web Content Provider Recommendation through Mining Individual Users’ QoS

Songhua Xu[♮,♯,‡∗]

Hao Jiang[‡]

Francis C.M. Lau[‡]

♮: College of Computer Science and Technology, Zhejiang University, Hangzhou, Zhejiang, 310027, P.R. China

♯: Department of Computer Science, Yale University, New Haven, Connecticut, 06520-8285, USA

‡: Department of Computer Science, The University of Hong Kong, Pokfulam Road Hong Kong S.A.R., P.R. China

ABSTRACT

We propose an optimal web content provider recommendation algorithm based on mining QoS (quality of service) information of the Internet. The QoS refers principally to the network bandwidth and waiting time (for a connection to be established). For contents replicated over multiple sites, our algorithm recommends a list of webpages having the desired content and ranked according to their QoSs for any specific user. The recommendation is generated through a data mining procedure based on known QoSs of connections between pairs of computers. Our user QoS mining procedure incrementally constructs a neural network group for QoS prediction based on clustering over the prediction errors. An accompanying decision tree algorithm is then used to select the most appropriate neural network among the neural network group to predict the QoS for a particular user connection. Based on our proposed recommendation algorithm, we have implemented a user-oriented search engine which can identify similar web content providers and make a ranked recommendation based on the prediction over the QoS experienced by individual users. Experiment results have verified that our QoS-based personal web content provider ranking algorithm can indeed produce a recommendation that improves the QoS experienced by individual users.

Categories and Subject Descriptors

C.2.m [**Computer-Communication Networks**]: Miscellaneous; H.3.5 [**Online Information Services**]: Commercial Services, Web-based Services; H.4.3 [**Information Systems Applications**]: Communications Applications; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems

General Terms

Algorithms, Design, Experimentation, Human Factors, Legal Aspects, Measurement, Performance

[∗]Contact him at A DOT B AT C DOT com in which A = "songhua", B = "xu", and C = "gmail".

Keywords

Web content provider recommendation, personalized recommendation, user QoS mining, optimal connection route selection, neural network group

1. INTRODUCTION

1.1 Motivation

There is a large variation in quality of service (QoS) of WWW access in many areas of the developing world such as China. The web browsing and file downloading experiences in these areas can generally be described as unreliable. Consequently there is a need to offer personal web source recommendation in order to maximize the QoS experienced by the end users. In fact, part of the unreliability experienced by the users is due to political, administrative, and economic reasons, as opposed to technical ones. Typically, ISP companies are not in shortage in in developing regions. In these regions, however, connections between computers subscribing to the same ISP tend to be significantly faster than connections across different ISP networks. This might be due to overly sophisticated configurations and topologies that are imposed on a large number of networks and sub-networks. A computer can belong to multiple sub-networks, each of which provides speedy, reliable connections to certain groups of computers but slow and unstable connections to other groups, depending on whether the groups belong to the same subnet. Therefore, optimal connection route selection can improve the overall QoS experienced by the end user, which unfortunately is difficult, time consuming and not always possible if done manually.

As QoS is so critical for web browsing and many other types of web access, any method that can improve QoS to a noticeable extent has tremendous commercial value. A number of solutions have been proposed and some of which have been put to commercial use. Among these solutions, the most successful large-scale commercial ones make use of the simple idea of automatically opening multiple links to web content providers to do parallel download or access. An example is a software called XunLei (http://www.xunlei.com/), which is the second most popular Chinese software among Chinese users. However, by using such kind of programs, the webpage visit counts of the content providers’ websites would suffer significantly because it is the automated programs that visit the webpages, rather than the end users; consequently, online advertisements in these webpages will lose their intended values. This problem has led to a number of court cases now going on between companies offering such kind of QoS improvement services and web content providers. In this paper, we propose a data mining based method to recommend web content providers that can offer optimal QoS for individual users. Our method takes into consideration the kind of possible legal issues as just described.

1.2 Contribution

In this paper, we propose a personalized web content provider recommender algorithm aiming at improving the QoS of Internet connections as experienced by individual users via mining known user QoS records. QoS here is in terms of network bandwidth and waiting time, the latter of which refers to the elapsed time between the moment a user clicks a button to request a page from the server and the moment the page is actually displayed on his screen. For web browsing, waiting time affects a user's experience more directly than bandwidth; whereas for downloading large files, bandwidth is more important. Our algorithm considers both factors in optimizing the QoS for a particular user connection through on-line mining a collection of QoS data; the results are fed to a training component for prediction of the user's network conditions. Based on the prediction results, personal webpage ranking by QoS is produced. We verify the effectiveness of our algorithm via experiments with both simulated and real networks. To the best of our knowledge, none of the current search engines or web accessing software have addressed this problem.

The rest of this paper is organized as follows. We first survey some related work. And then we give a high-level overview of our algorithm and the rationale behind our design. After that we explain how our algorithm predicts QoS values through a data mining procedure. And then we discuss how our user-oriented web content provider recommendation algorithm is constructed based on the predicted QoS for individual users. Afterwards, we report some experiment results. Finally we conclude the paper.

2. RELATED WORK

Recently, there has been a flurry of research activities on personalized or user aware search engines and algorithms, such as [10, 5]. In our previous work, we have also proposed a personalized solution using user attention time to predict the potential user interest for building a user oriented webpage search engine [12] or making personalized recommendations for documents, images and videos search [11]. In this paper, we present yet another personalized search engine design effort, which is dedicated to optimizing the QoS of individual users' web connection. More generally, our study on maximizing user QoS belongs to the broad category of research on user-oriented optimal route selection. In the following, we survey some most related work on this problem.

Routing is a widely encountered problem in many areas of studies as well as in daily life. Generally speaking, routing is the process to select a suitable path to transfer information or materials from one place in a network to another network location. The Dijkstra's shortest-path algorithm [4] is a famous example of an optimal routing algorithm, focusing on minimizing the traveling path length. There also exists a great number of domain-specific routing algorithms for large networks in reality. For example, David and Stefan [9] proposed a general heuristic based method for the vehicle transportation problem, which relies on a pickup and delivery model and a large neighborhood search process to solve the routing problem. There has also been a rich collection of work on utilizing the QoS of a network connection to optimize the route selection, some of which are learning-based approaches. For example, Boyan and Littman [1] proposed a Q-routing algorithm by embedding a reinforcement learning module into each node in the routing network for minimizing the delivery time of data packages over the switching network. Caro and Dorigo [2] introduced an adaptive learning approach called "AntNet" to adjust routing tables in a network. Their approach have been nicely adopted by many communication networks, especially on mobile agents. Recently, Peshkin and Savova [8] proposed another adaptive routing algorithm based on reinforcement learning, which tries to learn a set of policies on interaction with the network environment. One thing common to all these advanced routing algorithms is that the optimal routing algorithm is supposed to change the network configuration through altering the routing tables in the network or adjusting other network settings. Unlike these approaches, our new web content provider recommendation algorithm directly interacts with the end Internet users; hence it does not need to change the network configuration in any way. Since most users do not have control over or the abiliity to reconfigure the network, our method is thus more affordable and feasible to be adopted in reality.

The most related work to our study here is the downloading software called XunLei [13] which is extremely popular in China. The software uses a combination of PC-to-PC and PC-to-server downloading strategies to increase the downloading bandwidth. Unlike the peer-to-peer method, it does not require the client end to upload any data. When the user requests for a file, it will return to the user a list of all the available URLs that contain the file in a bandwidth descending order. Then the client program on the user computer will try to connect to several of the URLs at the top of the list. This way of parallel downloading can achieve a significant bandwidth increase. However, this company is entangled in some serious legal disputes because users can now download files directly using XunLei without looking at the original web pages on the web content provider(s). As a result, both the user visit counts and the time a user spent on those websites are greatly reduced, which means reducing significantly the values of the advertisements that appear in these webpages.

In designing our new algorithm, we avoid the above legal problem by returning a ranked list of pages that contain downloading links to user requested materials. Users will have to click on a page in order to initiate the download. This way we protect the business interests of content provider websites. Since users have to manually start the downloading process, we therefore carefully construct our algorithm to make a best possible recommendation on content provider websites to maximize the QoS of the network connection for any particular user. A major difference between our algorithm and XunLei's algorithm is that we model QoS as a function between a particular client and a server and try to make an optimal estimation on the QoS of such connection; whereas Xunlei's algorithm allows the user-side client program connecting to multiple servers to do the parallel downloading without having to optimize the QoS of a particular connection for the user.

3. OVERVIEW OF OUR METHOD

3.1 Client-Server Structure of Our Algorithm

At a high level, our algorithm consists of a server component and a client component. The client end runs a customized web browser which periodically reports to the server the QoS of its current connection. In our experiment, we set the frequency of this report to be four times a minute. At the beginning of a new connection, such QoS will also be reported to the server. The server side is responsible for predicting the QoS for a potential connection between a client computer and a server computer through a data mining process based on all the QoS information collected thus far. These predicted values are then used to produce a user-oriented web content provider recommendation to optimize the QoS experienced by a particular user.

3.2 Predicting QoSs for Individual Users

Each connection between a pair of client computer C_i and server S_j is associated with a certain bandwidth $b_{i,j}$ and a connection time $t_{i,j}$. The connection time is the duration between the moment when C_i requests a certain page from S_j and the moment when the page is actually displayed on C_i. Of course, the size of the webpage in some cases would have an effect on $t_{i,j}$. In reality, however, such variation is usually very small for most of the webpages. We assume therefore $t_{i,j}$ and $b_{i,j}$ are two largely independent factors, which characterize the QoS of a connection.

For simplicity, we assume bandwidth $b_{i,j}$ changes continuously and gently with the passage of time and remains at a certain fixed value for a short period of time. It is also often possible to have access to some properties of C_i and S_j. We denote these properties as $\mathbf{P}(C_i)$ and $\mathbf{P}(S_j)$ respectively. Currently $\mathbf{P}(X)$ includes computer X's IP address, $NetID(X)$, the ID of the sub-network to which X belongs, $SubNet(X)$, the ISP X uses, $ISP(X)$, and a timestamp of this connection, which further consists of the day of the week, $D(X)$, and the hour, $H(X)$. That is, for every computer X, whether it is a client or a server computer, we have:

$$\mathbf{P}(X) \triangleq \big(NetID(X), SubNet(X), ISP(X), D(X), H(X)\big) \tag{1}$$

The basic idea of our method is that we assume there exists a functional relationship ψ between $\mathbf{P}(C_i), \mathbf{P}(S_j)$ and the QoS of the connection between C_i and S_j, namely $b_{i,j}, t_{i,j}$. This can be formally described as:

$$\psi\big(\mathbf{P}(C_i), \mathbf{P}(S_j)\big) \to (t_{i,j}, b_{i,j}). \tag{2}$$

The server accumulates a collection of training records of the form $(\mathbf{P}(C_i), \mathbf{P}(S_j), t_{i,j}, b_{i,j})$ to be used in a data mining process to allow our algorithm to predict the QoS of a specific connection between a pair of computers. The rationale behind our learning based method is that *the more similar the network conditions of two connections are, the closer their QoSs are*. Thus by capturing the QoS of past and present web connections, the learning algorithm would be able to predict the likely QoS of a new connection. We assume that the five properties encoded in $\mathbf{P}(C_i)$ and $\mathbf{P}(S_j)$ provide a good depiction of the network connection condition between the pair of computers C_i and S_j; this is based on an empirical study we conducted, which suggests these five properties provide some most revealing clues on the connection condition.

It is non-trivial to learn function ψ in (2). In this paper, we introduce a data mining procedure which uses neural networks, clustering and decision tree algorithms to derive the best possible form of ψ. Once the function ψ is derived, and when a client needs to choose among multiple servers to connect to for requesting a webpage or file, our algorithm can predict the QoSs of the potential connections between this client and all the servers containing the desired content respectively, based on which an ordered list of content providers is suggested from the point of view of optimal connection QoS for the particular user. In the next section, we examine the details of this learning process to derive the function ψ.

4. PREDICTING QOS FOR INDIVIDUAL USERS VIA NEURAL NETWORK GROUP

4.1 Main Steps

The main steps of our learning based QoS predication procedure are as follows. After the training set is collected (Sec. 3.1), we train a neural network for predicting the functional relationship between the QoS values and the network conditions of the client and server computers. To improve the learning capability, we iteratively apply a clustering algorithm to introduce additional neural networks for refining the accuracy of the captured functional relationship. Given a neural network group, we also introduce a decision tree for identifying a most suitable neural network to predict the QoS values for a particular connection between a client computer C_i and a server S_j. Also using this neural network identification decision tree, we can evaluate the performance of the entire neural network group, which is useful for deciding when to stop the iterative neural network group construction process. The next few subsections examine the details of these steps.

4.1.1 Predicting Individual Users' QoS via a Neural Network

We use a group of neural networks to capture the functional relationship ψ in (2). The training data of user QoS records are classified into multiple categories with one neural network responsible for learning the functional relationship embodied in the training data in a category. How to divide all the training data into multiple categories will be shortly discussed. In this subsection, we are concerned with how to use a neural network to capture the functional relationship ψ for a category of training data.

According to (2), a neural network is expected to predict the QoS values $b_{i,j}, t_{i,j}$ of a certain user connection given the network conditions of the client computer C_i, $\mathbf{P}(C_i)$, and that of the server S_j, $\mathbf{P}(S_j)$. Each neural network we use is a multilayer perceptron with two hidden layers. We use the back-propagation method [6] with 10000 iterations to train each individual neural network. The training process is constructed in an online learning fashion: All the new connections from client computers to servers are added into the training set when users use our search service. Neural networks in our neural network group will be trained periodically on the search engine's server to replace the previous version on the fly. Compared with other more sophisticated data mining algorithms, neural network has an advantage that even though its training process is usually slow, once it finishes training, it can be evaluated very efficiently. This feature fits well in our optimal network route selection application which aims at real-time response but not necessarily fast training as the Internet connection condition is unlikely to change rapidly over a short span of time.

During our training process, we associate recent QoS samples on network connection conditions with larger weights to ensure the resultant neural networks would better reflect the QoS of the recent network connection condition. We use the following weighting function to compute a weight w at time t to be associated with a training record obtained at time $t_{creation}$:

$$w(t, t_{creation}) = \kappa_{mag} - \frac{\kappa_{mag}}{1 + \exp(\kappa_{period}(t_{creation} - t))}. \tag{3}$$

Here κ_{mag} controls the maximum possible magnitude of this weight, and κ_{period} indicates the rate at which the sigmoid shaped function converges, i.e., how much we favor recent QoS training records over the old ones. In all our experiments, we empirically set $\kappa_{mag} = 2$ and $\kappa_{period} = 1$ whose unit is day^{-1}, which achieves an optimal configuration empirically.

4.1.2 *Iteratively Constructing a Neural Network Group via Clustering the Training Errors*

Initially all the training data are assigned to one category, which will be learned from by a neural network. This neural network becomes the initial neural network group. All the training data will then be subdivided into multiple categories according to an error measurement on the performance of the trained neural network group. In this process, new neural networks will be introduced iteratively into the group, each responsible for a newly identified category of training data.

The purpose of introducing multiple neural networks to capture the functional relationship of (2) is to augment the learning capability of our learning device because the relationship is likely to be beyond the learnability of a single neural network. Our practice of classifying all the training data into multiple clusters and designating a neural network for learning a cluster of data amounts to using a collection of local functions of much simpler forms to approximate a much more complex function globally. The benefit is that the local functional relationship that each neural network is supposed to capture behaves more homogeneously, which significantly eases the machine learning process and improves the learning accuracy.

4.2 Deriving Training Errors for a Category of Training Data

Once every neural network in the neural network group is trained, we can apply the group to derive the errors of each training datum. For simplicity, we assume it functions as a black box for the time being. For each training datum $\big(\mathbf{P}(C_i), \mathbf{P}(S_j), t_{i,j}, b_{i,j}\big)$, we derive its training error and denote it as $e_{i,j}$, which is a scalar value.

4.3 Subdividing the Training Data Categories via Clustering the Training Errors

Having all the training errors derived, we first discretize all the errors using a frequency based discretization method. Specifically, we discretize the value range of all the errors corresponding to a category of training data into κ levels, denoted as $L_1, L_2, \cdots, L_\kappa$ respectively. The value of κ is optimally determined, which will shortly be explained. After the above error discretization process, each error value $e_{i,j}$ is associated with an integer between 1 and κ, representing which error level L_x $(x = 1, \cdots, \kappa)$ it falls into. We then use the k nearest neighbor algorithm to cluster the training data according to their respective discretized training error levels. The number of clusters specified for the clustering process is taken to be the number of discrete error levels, κ, determined in the above frequency based discretization process. Notice that for each category of training data, one run of the above error discretization and clustering process will be executed. Thus training data belonging to different categories will not participate in the same clustering process. The optimal value of κ is also separately determined for each category of training data.

4.4 Subdividing the Neural Network Group

After the above clustering process is applied, we get a new training record categorization schema where each category of training data before the clustering process might be subdivided into several smaller categories. For any training data category that either has changed its constituting training data or is newly formed, we train a new neural network to represent the local functional relationship embodied in the training data in the category. It is noted that by our algorithm design, each category of training data will be used to derive a neural network. Thus the above subdivision over the training data guides us to subdivide our neural network group.

4.5 Determining the Optimal Value for κ

To determine an optimal value for κ for each category of training data, our algorithm searches the range of integer values between 1 and 5. We tentatively assign each integer value as the κ value and train a new neural network group. We then call the error measurement module to select a best value for κ which achieves the highest learning accuracy. To avoid overfitting, we employ the tenfold cross-validation method during the neural network group performance evaluation process. The κ value is independently determined in this way when subdividing each category of training data.

4.6 Stop Condition

A stop condition is needed for the above iterative neural network group construction process: Each time after we execute a round of neural network group subdivision process, we evaluate the performance of the resultant neural network group. We compare the performance of this new group with that of the immediately preceding group. Our iterative neural network group construction process will terminate if the relative performance gain in terms of the learning accuracy improves by less than 2% after carrying out the current round of neural network group subdivision process.

4.7 Evaluating Performance of a Neural Network Group via a Decision Tree

Evaluating the performance of a trained neural network group might look trivial at first glance as one might suggest to simply test a piece of test datum through running the datum's corresponding neural network. Unfortunately, this simple way cannot do it—recall earlier when we subdivide a neural network into some smaller networks through clustering, we are relying on the prediction errors to perform the clustering. However, in the testing case, we cannot peek at the groundtruth QoS values to derive the prediction errors.

To address this problem, we propose a decision tree based algorithm to locate a most suitable neural network in the neural network group for predicting QoS for a particular connection between C_i and S_j. Once again this is a learning problem. This time, each record of the training samples is in the form of $(\mathbf{P}(C_i), \mathbf{P}(S_j), NN\_ID)$ where $\mathbf{P}(C_i)$, $\mathbf{P}(S_j)$ are the inputs to ψ in (2) and NN_ID is the index number of the neural network responsible for capturing the relationship between QoS and the network condition for the connection between C_i and S_j. Recall NN_ID is assigned when we derive a neural network from a category of training data. With these training samples prepared, we can train a decision tree for predicting the index number of the neural network most suitable for predicting the QoS of the connection between C_i and S_j.

One of the most important fields in $\mathbf{P}(X)$ probably is computer X's $NetID$, which we assume to be its IP address in our current experiment. However, the space of all the possible IP addresses is huge, which makes equal discretization infeasible for training any decision tree of a reasonable size. We therefore follow an entropy-based discretization method to discretize the ranges of the whole IP addresses into 1000 categories according to the IP addresses in all the training records, i.e., the collection of all the IP addresses of C_i's and S_j's in the training set. For the rest of the components in $\mathbf{P}(C_i), \mathbf{P}(S_j)$, i.e., $SubNet, ISP, D, H$, they are acquired as discrete values. So there is no extra discretization needed for them.

Once the discretization process is finished, we can utilize the classical C4.5 decision tree algorithm to derive a decision tree for identifying neural networks.

Given such an indexing decision tree, for each test record in the form of $(\mathbf{P}(C_i), \mathbf{P}(S_j), t_{i,j}, b_{i,j})$, we first run the tree on the fields of $\mathbf{P}(C_i), \mathbf{P}(S_j)$ to get the neural network indexing number NN_ID for locating a particular neural network in the neural network group. And then we apply the identified neural network to the fields of $\mathbf{P}(C_i), \mathbf{P}(S_j)$ to estimate the corresponding connection waiting time $t'_{i,j}$ and bandwidth $b'_{i,j}$. By comparing $t'_{i,j}, b'_{i,j}$ against the groundtruth $t_{i,j}, b_{i,j}$, we can derive the error on the test record as:

$$e_{i,j} \triangleq |t_{i,j} - t'_{i,j}| + \kappa_{bandwidth}|b_{i,j} - b'_{i,j}|, \qquad (4)$$

where $\kappa_{bandwidth}$ is a coordinating parameter whose typical value in our experiment is set as 500. Finally, summing the errors on all the testing records, we can derive the overall error on the testing set, which is considered the overall performance measurement for the neural network group under evaluation.

5. PERSONALIZED WEB CONTENT PROVIDER RECOMMENDATION BASED ON PREDICTION OF USER QOS

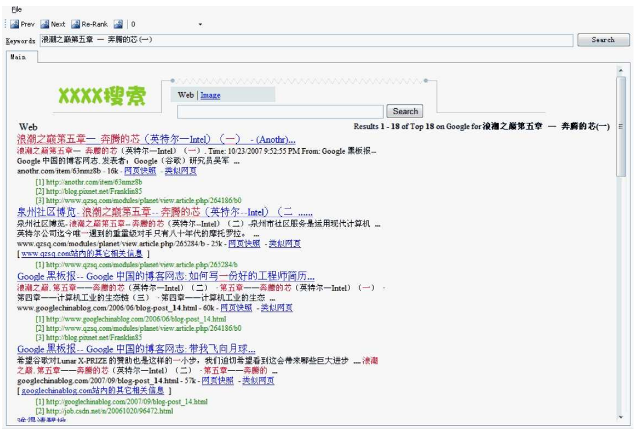
Based on the above prediction of individual users' QoS, we design our personalized web content provider recommendation algorithm for optimizing QoSs experienced by the users.

5.1 Detecting Duplicate Copies of Web Contents

Among all the candidate webpages and online documents to be searched from, our algorithm first evaluates pairwise content similarities. For webpages, it is through detecting their document similarities using the document content similarity estimation algorithm proposed in [7]. If the similarity is above a prespecified threshold (0.95), we assume these two webpages contain duplicate copies of the same content. For pages that contain a download link, we will compare the download files pointed to by these links. We use random bit comparison to determine whether these links refer to the same file. This is possible since most of the file servers support downloading segments of a file at an arbitrarily requested position inside the file. We randomly picked 10 segments of the file of size 1K and compare their contents bit by bit. If two files are found to have the same content in these 10 random segments, we assume they are duplicate copies of the same content.

5.2 Personalized Web Content Provider Recommendation to Optimize User QoS

Each time when a user submits a search query, we first call the traditional Google Pagerank algorithm to return an ordered list of related webpages, which are denoted as $\mathbf{W} \triangleq \{w_1, w_2, \cdots, w_n\}$ where w_i is the i-th webpage in the ordered list. And then using the algorithm discussed in Sec. 5.1, we group webpages containing the same content together, resulting in a number of content groups of webpages $\mathbf{G} \triangleq g_1, g_2, \cdots, g_m$, which have the properties: 1) $\forall w_i \in \mathbf{W}, \exists j,$ s.t. $w_i \in g_j$; 2) $w_i \in g_j \Rightarrow w_i \notin g_t$ $(t = 1, 2, \cdots, j-1, j+1, \cdots, m)$; 3) $\forall w_i \in g_t, w_j \in g_t$ where $g_t \in \mathbf{G}$, w_i and w_j are duplicate copies of the same content. The rank of a webpage content group is the highest rank of its constituent webpage, i.e., $rank(g_t) \triangleq \min\{i | w_i \in g_t\}$. Because of the three properties above regarding the webpage grouping operation, it is



浪潮之巅第五章 — 奔腾的芯 (一)

Web

Results 1 - 18 of Top 18 on Google for 浪潮之巅第五章 — 奔腾的芯(一)

浪潮之巅第五章— 奔腾的芯（英特尔—Intel）（一） - (Anothr)...
浪潮之巅第五章— 奔腾的芯（英特尔—Intel）（一）. Time: 10/23/2007 9:52:55 PM From: Google 黑板报--
Google 中国的博客网志. 发表者：Google（谷歌）研究员吴军 ...
anothr.com/item/63nmz8b - 16k - 网页快照 - 类似网页
[1] http://anothr.com/item/63nmz8b
[2] http://blog.pixnet.net/Franklin85
[3] http://www.qzsq.com/modules/planet/view.article.php/264186/b0

泉州社区博览- 浪潮之巅第五章-- 奔腾的芯（英特尔--Intel）（二 ......
泉州社区博览-浪潮之巅第五章-- 奔腾的芯（英特尔--Intel）（二）-泉州市社区服务是运用现代计算机 ...
英特尔公司迄今唯一遇到的重量级对手只有八十年代的摩托罗拉。 ...
www.qzsq.com/modules/planet/view.article.php/265284/b - 25k - 网页快照 - 类似网页
[www.qzsq.com站内的其它相关信息]
[1] http://www.qzsq.com/modules/planet/view.article.php/265284/b

Google 黑板报-- Google 中国的博客网志: 如何写一份好的工程师简历...
浪潮之巅. 第五章——奔腾的芯（英特尔—Intel）（二） · 第五章——奔腾的芯（英特尔—Intel）（一） ·
第四章——计算机工业的生态链（三） · 第四章——计算机工业的生态 ...
www.googlechinablog.com/2006/06/blog-post_14.html - 60k - 网页快照 - 类似网页
[1] http://www.googlechinablog.com/2006/06/blog-post_14.html
[2] http://www.qzsq.com/modules/planet/view.article.php/264186/b0
[3] http://blog.pixnet.net/Franklin85

Google 黑板报-- Google 中国的博客网志: 带我飞向月球...
希望谷歌对Lunar X-PRIZE 的赞助也是这样的一小步，我们迫切希望看到这会带来哪些巨大进步 .... 浪潮之巅. 第五章——奔腾的芯（英特尔—Intel）（二） · 第五章——奔腾的 ...
googlechinablog.com/2007/09/blog-post_14.html - 57k - 网页快照 - 类似网页
[googlechinablog.com站内的其它相关信息]
[1] http://googlechinablog.com/2007/09/blog-post_14.html
[2] http://job.csdn.net/n/20061020/96472.html

Figure 1: A snapshot of our customized search engine. Pages containing the same contents are listed under the same item and ordered according to the QoS of user access as predicted by our algorithm.

not possible that two webpage content groups will have the same rank value.

For each of the webpages or online files that fall into the same webpage content group, we run our QoS prediction algorithm as explained in Sec. 4 to estimate the bandwidth b_i and waiting time t_i for the webpage as if it were to be viewed or downloaded by this user's computer from its resident server computer. We then rank these webpages or files according to the objective function $O_i = t_i + \kappa_{bandwidth} b_i$, where $\kappa_{bandwidth}$ is the balancing parameter used in (4). We organize all these documents in a descending order according to the values of the above objective function.

In a nutshell, our new search engine returns to the end user an ordered list of web content groups according to the group ranks $rank(g_t)$ calculated above. Within each group, all the webpages or online documents carry the same content and are listed in a descending order according to the values of the objective function O_i computed above. A snapshot of an example of search results by our user oriented webpage ranking algorithm is shown in Figure 1, which is displayed in the client end browser. In our customized client browser, by default, only the top three resources are displayed, or ten items will be listed if the user prefers a more comprehensive displaying format.

6. EXPERIMENT RESULTS

To verify the effectiveness of our algorithm, we conducted several experiments both in simulated network environments and with the real WWW in China. The advantage of simulation experiments is that we can thoroughly test the behaviors of our algorithms with different network configurations and conditions. We might however miss some subtle aspects of the real WWW, which could affect the algorithm's behavior in an unexpected way. The advantage of doing experiments with the real WWW is that all the data are first-hand, most realistic, practical and indicative of the actual performance. Therefore we conduct experiments in both settings.

6.1 Simulation Experiments

6.1.1 Basic Environment Setup

We build a virtual network environment simulating various network conditions of the WWW, with a special focus on emulating the sit-

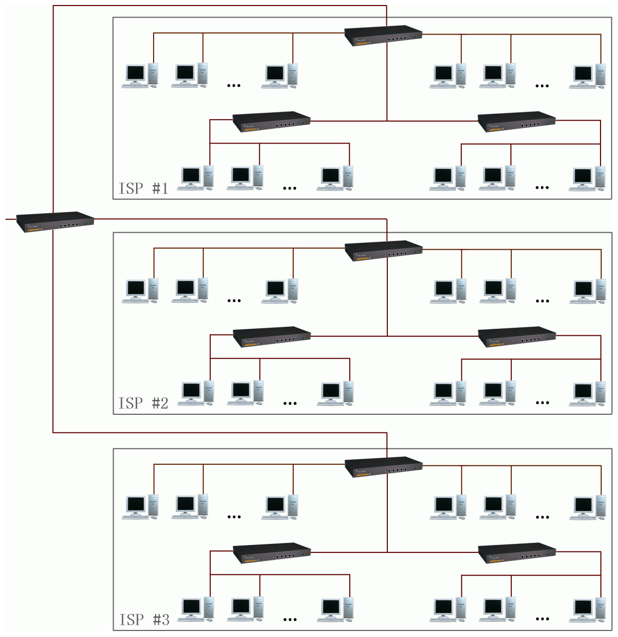


Figure 2: A simplified view of the virtual network for the simulation experiment.

uation of the WWW in Internet-developing regions where there is wide variation in QoS experienced by the users and hence our algorithm would be most useful. In this virtual environment, we introduce several virtual ISPs representing those major ISP companies in the region. Each of them possesses more than 10000 computers. Inside the network supported by each ISP, the network configuration is organized in a tree-shaped structure through a number of routers. The leaves of a tree represent the computers. A simplified view of the virtual network configuration used in our simulation experiment is shown in Figure 2.

Each computer in this network can be either a client or a potential web server or both. When an HTTP request for computer B is initiated by computer A, the request will travel from A to B through several routers. Every router imposes a latency in delivering the request. In our simulation experiments, the latencies for routers at a deeper level of the ISP tree are set to be higher than those at an upper level, and the cable bandwidths at a deeper level are set to be smaller than those at an upper level. Routing between computers belonging to different ISPs will have a much higher latency than between computers belonging to the same ISP. The waiting time for a computer to access another computer is set to be roughly constant if they are supported by the same ISP. However, these constants are different for different ISPs. These assumptions are all based on our observations and experiences with the WWW in China, by far the biggest Internet-developing country with a huge user population. The latency settings are acquired from pinging between computers which may or may not be supported by the same ISP in the real Internet. The settings over the servers in our virtual network also emulate the behaviors and properties of a few popular web sites together with many small ones in China; all the traffic data have come from one of the most famous web information companies—Alexa Internet Inc. (http://www.alexa.com/).

Once the virtual network is set up, we can test our algorithm performance with different user access patterns and network conditions in a controlled manner. In each round of the experiments, we first generate the user access patterns. To do this, we randomly pick pairs of clients and servers in the virtual network. We also randomly choose sizes of files to be transmitted in each connection. The routing path chosen for each connection is computed using Dijkstra's shortest path algorithm where each edge in the ISP service trees has a unit weight. After a package finishes its traveling from the requesting client to the destination computer, we record the time consumed and use twice this time plus the data transferring time under the current bandwidth as the total downloading time for this online browsing or downloading task. Moreover, in our simulation experiment, router latency at each routing operation is perturbed by a Gaussian noise. We repeat the above process multiple times to get a collection of training records, whose number is typically on the order of several 10Ks on average. The connection distribution on the servers is assigned according to the web server traffic data from Alexa Internet Inc.

6.1.2 *Our Algorithm's QoS Prediction Capability*

In Figure 3, we analyze the QoS prediction capability of our algorithm on both waiting time and bandwidth when different numbers of neural networks are used in our QoS predicting neural network group. We use the standard ten-folded cross validation technique for measuring the performance of our algorithm. We examined three types of network configurations here. One is the special case in which the routing time between a pair of computers is set according to the difference between their IP addresses, which is a toy case we purposely cook up for comparison purpose. The second is a more realistic case representing situations with the real WWW where only computers directly under the same router will have their IP address being consecutive. The third case extends the second, which offers a most realistic approximation to WWW conditions in China, where the IP addresses and web-access latency settings obey the actual distribution of two major ISPs in China, i.e., China Telecom and China Netcom together with CERNET (China Education and Research Network) which serves the colleges and education institutes. Statistics shown in the table reveal that: 1) our QoS prediction algorithm can satisfyingly capture the QoS conditions of Internet with over 10K user QoS records as the training data; 2) for a network configuration approaching the real situation of the WWW, having a group of neural networks is a necessary move; 3) if we consider a predication error of 20 ms to be tolerable, the rough range of the number of neural networks needed is below 50, which is a reasonable number, feasible for today's PCs for evaluating the neural network group in real-time, thus guaranteeing the real-time execution of our algorithm; 4) having 10000 records of user QoS data for training is sufficient for our simulated network environment.

6.1.3 *Performance Gain using Our Algorithm*

Given our QoS prediction neural network group, personalized web content provider recommendation is produced by our algorithm to assist individual users' webpage browsing or file downloading. We measure the performance gain after using our algorithm as the percentage of time saved to finish the webpage browsing or file downloading tasks if following our algorithm's recommendation. We construct the following setup for experimentation. In our virtual network environment, we create 100000 documents in 2000 groups distributed over the servers. All the documents in the same group are multiple copies of the same document. For simplicity, each query to the search engine returns just an ordered list of the documents in the same group, which are ordered by our personalized web content provider recommendation algorithm. Given such a

Records \ NN #	1	5	10	50	100
10000	43.1ms	21.0ms	17.7ms	7.5ms	5.6ms
	22.4Kb	11.9Kb	9.9Kb	4.6Kb	2.7Kb
50000	47.2ms	19.3ms	14.5ms	6.7ms	4.5ms
	29.5Kb	10.1Kb	9.1Kb	4.3Kb	2.8Kb
100000	107.3ms	18.6ms	12.6ms	6.5ms	4.2ms
	55.7Kb	10.1Kb	7.7Kb	4.6Kb	2.4Kb

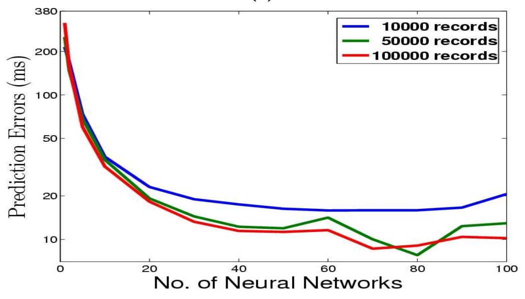
(a)

Records \ NN #	1	5	10	50	100
10000	334.2ms	68.3ms	44.3ms	24.6ms	22.9ms
	152.1Kb	38.0Kb	25.4Kb	14.2Kb	12.4Kb
50000	317.6ms	63.6ms	41.7ms	21.1ms	19.0ms
	173.0Kb	34.8Kb	19.6Kb	11.1Kb	9.9Kb
100000	373.3ms	63.0ms	40.4ms	19.8ms	17.3ms
	211.6Kb	27.4Kb	18.8Kb	12.5Kb	9.3Kb

(b)

Records \ NN #	1	5	10	50	100
10000	214.7ms	73.5ms	37.3ms	16.3ms	20.6ms
	112.4Kb	43.3Kb	17.7Kb	10.5Kb	10.6Kb
50000	257.8ms	70.8ms	36.6ms	10.0ms	11.8ms
	132.2Kb	38.1Kb	19.3Kb	4.8Kb	7.9Kb
100000	314.6ms	60.0ms	32.0ms	11.3ms	10.2ms
	210.3Kb	34.2Kb	19.3Kb	5.4Kb	5.9Kb

(c)



(d)

Figure 3: Statistics for the training accuracy on predicting waiting time and bandwidth using different sizes of neural network groups. The columns in (a)–(c) stand for the different numbers of neural networks in a neural network group, the rows correspond to different numbers of training records used in the learning process. Each cell in the table is the average absolute difference between the predicted connection time/bandwidth and the known connection time/bandwidth carried in the groundtruth data. The unit is in milliseconds/kilobits per second(Kbps). (a) is the special experiment setting where NetIDs are assigned to computers in the virtual network, which approximately reflect the routing time/waiting time between pairs of computers. In this setting, the difficulty of predicting connection time and bandwidth via data mining is significantly reduced. (b) reports results of an experiment where NetID is randomly assigned for computers in the network with the only constraint that computers directly under the same router have their NetIDs being consecutive. (c) gives results for an experiment where the IP address distribution and networking configurations are purposely set up to approximate the actual situation of the WWW in China. (d) shows the visual analysis over the training accuracy of our algorithm for predicting the QoS in the experiment reported at (c). Here we analyze the performance of our algorithm on a finer granularity scale with the increment step for the number of neural networks used being 1.

recommendation list, we assume the probability that a virtual user would click on the i-th top ranking result in the ordered list is $\frac{3}{4^i}$. We also randomly choose 10% of the virtual servers to be out of function to simulate link failure and network unstability.

After setting up the simulation scenario, we can then evaluate our algorithm performance in terms of the overall waiting time and bandwidth improvement by comparing between the cases when a user randomly connects to a website on the Internet providing the contents he wants versus links to the website upon the recommendation of our algorithm (which optimizes the QoS he would experience). In this simulation experiment, a group of one hundred virtual users with one hundred randomly generated client computer properties, i.e., $\mathbf{P}(X)$'s, are simulated to perform a task to download an online file of size 100M. We calculate the average time needed by these virtual users to finish the three web access tasks, which is indicative of both the waiting time and the bandwidth of their connections.

Figure 4 shows partial statistics of the experiment results. In each table cell of (a)–(f), we report the average time (in milliseconds) for these one hundred users to complete the tasks for one hundred times. The rows of these tables show the numbers of neural networks in the neural network group used by our algorithm. The first row is the case when no recommendation from our algorithm is available, which corresponds to the situation that the users try to connect to one of the websites containing the materials they need at random without any extra clue on network QoS conditions. This case is very similar to the real situation experienced by an unexperienced WWW user. We also calculate and plot the percentage of time saved in executing the web browsing or downloading tasks with the help of the recommendation from our algorithm. This experiment shows when the number of neural networks employed is above 10, significant time savings can be achieved. Through this experiment, we also show users can benefit in terms of the WWW QoS they experience by following the personalized recommendations on web content providers suggested by our algorithm.

6.2 Experiments on WWW in China

We also conducted experiments with the real WWW in China in order to have a more realistic measurement over the performance gain using our algorithm. The training set consists of around 15000 records of individual user's QoS data collected from web accessing logs generated by university students when surfing the Internet using our customized web browser for over 1 week. The experiment design is as follows: We selected 100 web servers supported by three major ISPs in China. We also randomly selected 10 copies of 500 webpages, whose average size is 10.6K, 10 copies of 500 files, whose average size is 3.49M, and 10 copies of 20 popular online game distribution files, whose average size is 784M, all scattered across these servers. We then invited 20 guest users (ten conducting experiments at home and ten in their offices) to browse ten randomly selected webpages among the 500 webpages, download ten files randomly selected from the 500 online files as well as to download one file randomly selected from the 20 game distribution files using our customized browser. The purpose of having some experiments conducted at home web access condition and others conducted at office condition is to see how effectively our algorithm can help the users in gaining better web access QoS under different conditions. All the users were asked to conduct the browsing and downloading tasks twice—the first time, we disabled our webpage ranking algorithm and the users would have to figure out which source to choose to browse or download from; the second time, we recommended the websites to visit using our webpage ranking algorithm. We compared the total amount of time they consumed in carrying out these Internet access tasks, which is recorded by our client end customized web browser. These time data are shown in Figure 5.(a)–(d). These statistics reveal that users can roughly save 30% to 70% of their time when performing a web browsing or online file downloading task, which proves the benefits brought by our recommendation algorithm.

6.3 Comparison with XunLei

We also compared the effectiveness of using our algorithm versus existent commercial downloading assistance software. Here we choose to compare with the software of XunLei [13] because it is by far the most popular downloading tool in China, which can achieve the best downloading efficiency improvement. To make a fair comparison with XunLei, we disabled the parallel downloading option in XunLei because this is the feature which caused many legal disputes. XunLei's algorithm [3] simply sorts the candidate web sites according to their most recent QoS records known to the server. Two major differences exist between their algorithm and ours: 1) their algorithm does not predict the QoS of unvisited websites; 2) their algorithm does not associate QoS with any user information; i.e., QoS records generated by user connections in Beijing are mixed with those in small towns, and hence their QoS optimization is not personalized. In comparison, our algorithm addresses both issues via a data mining approach.

We conducted two types of control experiments: One with the real WWW in China and the other under the simulated network environment. The results of the comparison experiments with the WWW in China are also reported in Figure 5.(a)–(c). (d) compares the performance gain in terms of percentage of time saved using XunLei and our algorithm respectively. Statistic characters including the minimum, maximum, mean, first decile (10th percentile), first quartile (25th percentile), median (50th percentile), third quartile (75th percentile), ninth decile (90th percentile) are plotted. In the simulated environment, we implemented the data downloading method proposed by XunLei [3] and repeated the simulation experiment designed studied in Figure 4. We show the experiment results in Figure 5.(e)–(g), from which a noticeable extra saving of downloading time can be clearly observed using our algorithm versus XunLei. Comparing between the experiment results obtained with the WWW in China in reality and the simulated environment, we found even though under both situations our algorithm outperforms XunLei's method consistently, the performance gain with WWW in China is not as significant as in the simulated network environment. We assume this is because XunLei's servers possess a huge amount of access log data across the country, which makes up for its simplistic site selection method, especially for transfer of large files. We thus assume if our algorithm is under massive deployment or has access to a large scale user QoS training record set, the performance gain achieved with the WWW in China using our algorithm would be much more noticeable.

7. CONCLUSION

In this paper, we present a personalized web content provider recommendation algorithm based on the prediction of individual users' QoS. Our algorithm features a data mining procedure, which incrementally constructs a neural network group for QoS prediction based on the clustering over prediction errors. A decision tree algorithm is also introduced to accompany the neural network group for selecting the most appropriate neural network among the group to predict QoS for a particular user connection. Experiment results with both the real WWW in China and simulated network environments have verified the effectiveness and benefits of our new algorithm.

Our user-oriented webpage ranking algorithm can be applied to optimize QoS of user WWW experiences, which can be useful in a number of applications. For example at present in China and many other developing countries, computer games are extremely popular, generating billions of dollars of revenue each year, surpassing that of the film industry. However, due to the limitations with the WWW in these regions, most online game manufacturers have to use CD or DVD retail disks to release their games or updates, which is very inconvenient for the users and may also shun business opportunities. Using our algorithm, a better QoS can be experienced by WWW users in China, which can help overcome this problem. In general, provision of better QoS for Internet users will make more people realize the importance and take advantage of the benefits of new web technologies, which in turn could benefit the development of the WWW in the country and elsewhere.

8. ACKNOWLEDGEMENT

This work has several patents pending.

9. REFERENCES

[1] J. A. Boyan and M. L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in Neural Information Processing Systems*, volume 6, pages 671–678, 1994.

[2] G. D. Caro and M. Dorigo. Antnet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.

[3] X. L. N. T. Company. A method & system for downloading data. *China Patent: 200710084661.7*, 2007.

[4] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[5] Z. Dou, R. Song, and J.-R. Wen. A large-scale evaluation and analysis of personalized search strategies. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 581–590, 2007.

[6] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, second edition, 2001.

[7] Microsoft. Shingleprinting code for estimating document similarity. Source code package, 2004. http://research.microsoft.com/research/downloads/.

[8] L. Peshkin and V. Savova. Reinforcement learning for adaptive routing. *ArXiv Computer Science*, 2007.

[9] D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, 2005.

[10] J. Pitkow, H. Schütze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel. Personalized search. *Commun. ACM*, 45(9):50–55, 2002.

[11] S. Xu, H. Jiang, and F. C. Lau. Personalized online document, image and video recommendation via commodity eye-tracking. In *RecSys '08: Proceedings of the 2nd ACM Conference on Recommender Systems*, pages 83–90, 2008.

[12] S. Xu, Y. Zhu, H. Jiang, and F. C. M. Lau. A user-oriented webpage ranking algorithm based on user attention time. In *AAAI '08: Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1255–1260, 2008.

[13] XunLei. Software, http://www.xunlei.com/, Xun Lei Network Technology Company, Ltd., Shenzhen, China.

NN #	web (10K)	data (1M)	data (100M)
0	285.3	23,472	2,290,176
1	233.7	17,205	1,662,668
5	117.8	10,445	846,298
10	97.0	8,098	785,530
50	87.1	6,957	695,418
100	88.3	6,329	718,152

(a) simulation experiment for regions with poor web access conditions (number of training records = 10000)

NN #	web (10K)	data (1M)	data (100M)
0	296.6	23,631	2,394,101
1	238.8	18,149	1,941,615
5	118.6	8,791	945,567
10	77.1	6,940	825,965
50	76.3	5,334	631,479
100	73.3	5,404	598,086

(b) simulation experiment for regions with poor web access conditions (number of training records = 50000)

NN #	web (10K)	data (1M)	data (100M)
0	269.8	22,255	2,250,904
1	195.6	17,381	1,609,396
5	80.1	7,077	841,838
10	54.8	4,874	567,228
50	41.5	4,015	347,065
100	41.5	3,387	338,572

(c) simulation experiment for regions with poor web access conditions (number of training records = 100000)

NN #	web (10K)	data (1M)	data (100M)
0	135.8	3,808	486,680
1	133.1	4,116	476,460
5	123.1	3,610	435,579
10	124.5	3,397	346,516
50	102.5	3,046	305,148
100	98.9	2,871	279,206

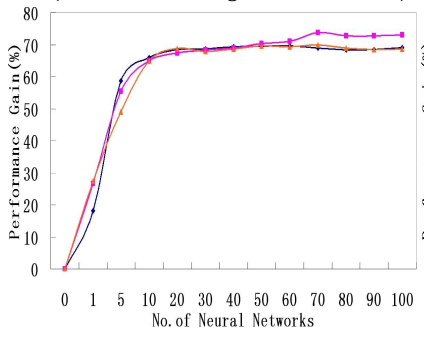
(d) simulation experiment for regions with good web access conditions (number of training records = 10000)

NN #	web (10K)	data (1M)	data (100M)
0	140.2	4,597	369,387
1	142.4	4,951	361,999
5	125.9	4,275	340,575
10	108.7	3,480	271,869
50	106.1	3,273	253,768
100	103.5	3,167	249,336

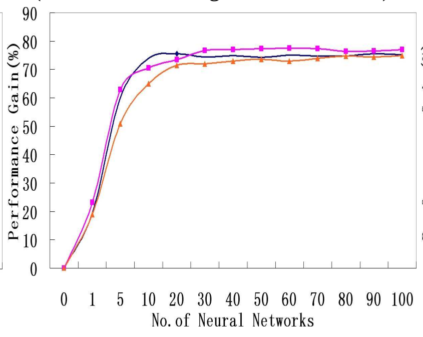
(e) simulation experiment for regions with good web access conditions (number of training records = 50000)

NN #	web (10K)	data (1M)	data (100M)
0	175.8	8,012	494,828
1	180.9	7,403	510,168
5	152.2	6,794	422,088
10	114.8	3,332	220,693
50	101.8	2,660	196,942
100	97.2	2,732	191,004

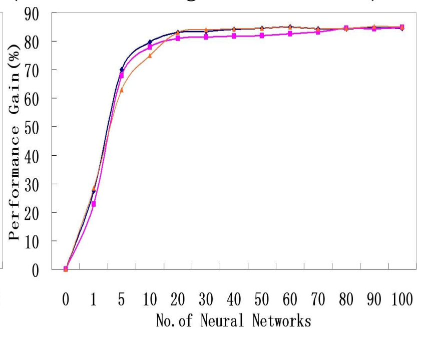
(f) simulation experiment for regions with good web access conditions (number of training records = 100000)



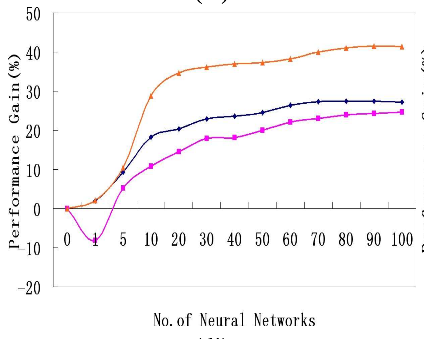
(a')



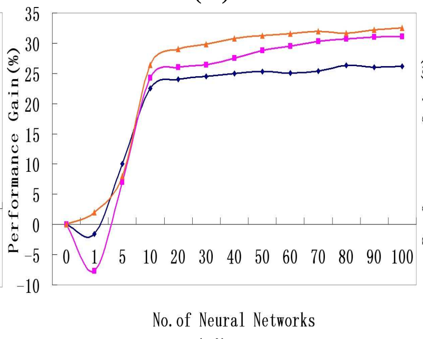
(b')



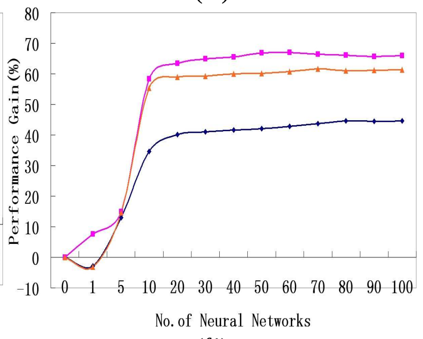
(c')



(d')



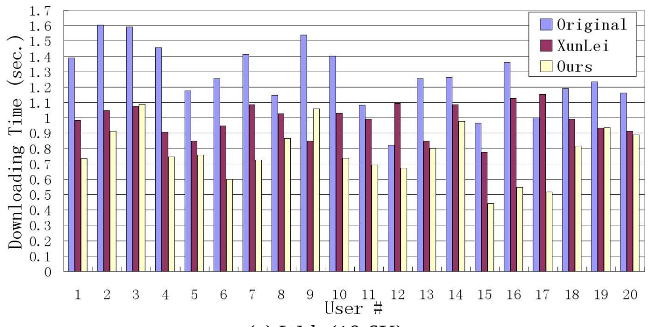
(e')



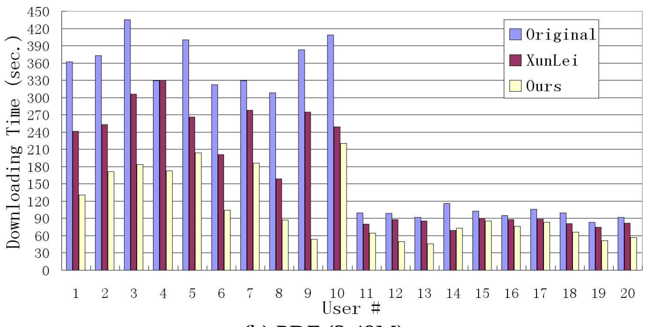
(f')

◆ web(10K) ■ data(1M) ▲ data(100M)

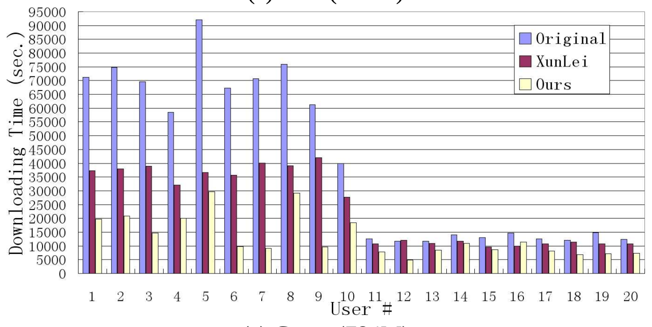
Figure 4: Examine the prediction capability of our neural network group with different group sizes, i.e. with different numbers of neural networks in the group. We report the average time in milliseconds consumed by a group of 100 virtual users in performing the three web browsing and downloading tasks when our web content provider recommendation algorithm is supported by different sizes of QoS prediction neural network groups. (a)–(c) are conducted in a virtual network environment where the latency between routers supported by different ISPs is set to be 200ms. Such a virtual network is set up to simulate poor web access conditions, e.g., those in the rural regions. (a)–(c) give the results when our algorithm has access to different sizes of training sets. (d)–(f) are conducted in a virtual network which simulates good web access conditions, e.g., cities. The main difference is that the latency between routers supported by different ISPs is set to be 50ms. (d)–(f) give the results when our algorithm has access to different sizes of training sets. In these tables, we analyze the prediction capability of our neural network group when it has 0, 1, 5, 10, 50 and 100 neural networks where the case of 0 neural network corresponds to the situation when our recommendation algorithm is turned off and the virtual user has to randomly choose a web source to download the files. A more refined analysis is given in the subfigures at the bottom in which we plot the performance gain in terms of the percentage of time saved to finish the web data downloading tasks when using our algorithm. We examine the performance gains when our recommendation algorithm is supported by different sizes of neural network groups. In these subfigures, (a') corresponds to (a); (b') corresponds to (b); and so on.



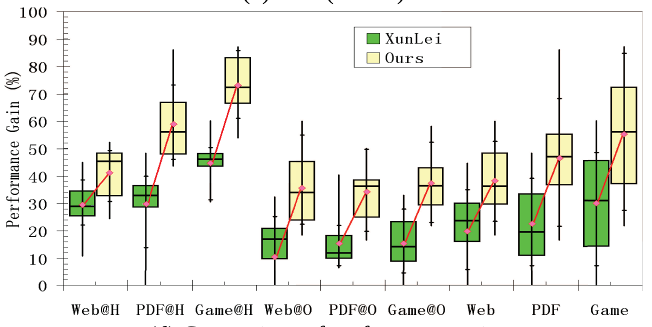
(a) Web (10.6K)



(b) PDF (3.49M)



(c) Game (784M)



(d) Comparison of performance gain

#	web (10K)	data (1M)	data (100M)
original	285.3	23,472	2,290,176
XunLei	254.4	19,603	1,951,160
Ours	88.3	6,329	718,152

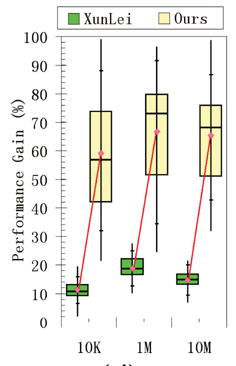
(e) Downloading time under the simulated network with 10000 records of user QoS data

#	web (10K)	data (1M)	data (100M)
original	296.6	23,631	2,394,101
XunLei	253.6	18,858	1,855,428
Ours	73.3	5,404	590,086

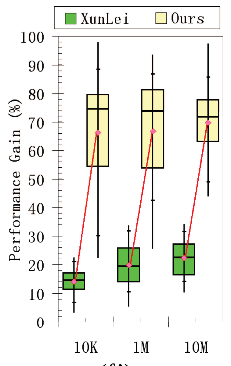
(f) Downloading time under the simulated network with 50000 records of user QoS data

#	web (10K)	data (1M)	data (100M)
original	269.8	22,255	2,250,904
XunLei	204.8	15,710	1,482,256
Ours	41.5	3,387	338,572

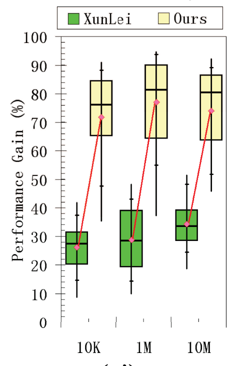
(g) Downloading time under the simulated network with 100000 records of user QoS data



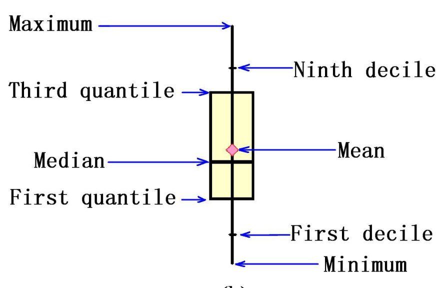
(e')
Statistic characteristics for (e)



(f')
Statistic characteristics for (f)



(g')
Statistic characteristics for (g)



(h)
Statistic characteristics in a boxplot element

Figure 5: Comparison between the results of experiments of downloading webpages, PDF files and game distribution files without using any downloading software ("original"), using XunLei ("XunLei") and using our algorithm ("Ours") respectively. (a)–(c) show the results of experiments conducted on the WWW in China by a group of twenty users. In (d), we compare the performance gain in terms of the percentage of time saved to finish the above downloading tasks after using XunLei and our algorithm respectively, including: 1) statistics of experiment results by User #1–User #10 at home (@H); 2) statistics of experiment results by User #11–User #20 in their office (@O); and 3) statistics of all the experiment results by the twenty users. On the horizontal axis, we use "Web@H", "Web@O" and "Web" to represent the experiments of webpage downloading conducted at home, in offices, and in either environment respectively. The prefixes 'PDF" and "Game" in the labels stand for the PDF and game distribution file downloading experiments. To report these statistics, we utilize the boxplots to convey the statistically important characteristics of all the downloading times in the corresponding experiment (See (h) for the locations of these characteristics in a boxplot element). To compare with results obtained with experiments on the real networks, (e)–(g) list the time consumed to finish the web browsing and file downloading experiments under the simulated network environment with the number of user QoS training records being 10000, 50000, and 100000 respectively. The router latency at each routing operation is perturbed by a Gaussian noise to emulate the uncertainty in the network conditions in reality. Each table reports the average time in milliseconds consumed by the 10000 virtual users to finish the respective tasks. (e'), (f'), (g') compare statistic characteristics of the performance gain in terms of time saved experienced by the 10000 virtual users when they perform these three web access tasks using XunLei and our algorithm respectively with respect to the situation when they do not use any downloading software. Here (e') illustrates the experiment reported in (e), (f') for (f) and (g') for (g). (h) illustrates the statistic characteristics plot in a boxplot element.