Better Client OFF Time Prediction to Improve Performance in Web Information Systems*

Alan Berfield, Bill Simons, Panos K. Chrysanthis, Kirk Pruhs
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260, USA
{alandale,simons,panos,kirk}@cs.pitt.edu

ABSTRACT

Prefetching is a potential technique for reducing latency in Web information Systems. However, it has been shown that the burstiness of standard prefetching can drastically increase network congestion, and can even increase, rather than decrease, average user perceived latency. Accurate OFF time, the idle periods between user requests, prediction potentially allows the document to be downloaded at an even rate over the OFF time, which can ameliorate the burstiness, and significantly improve both network congestion and average user perceived latency. Yet accurate prediction of such OFF times has been difficult to achieve. This paper examines the use of two machine-learning techniques, namely, neural networks and genetic algorithms, for OFF time prediction. Our performance evaluation results show that these techniques provide better accuracy than those previously reported, with an average increase of twice the correlation. Our results also show that document type is the best predictor of OFF time. Further, our functions can be tailored to favor underpredictions, which would have less negative effects on the overall network than overpredictions.

1. INTRODUCTION AND MOTIVATION

As the Internet becomes more and more popular, it also becomes more difficult to deliver content in a timely manner. There is a strong need for reducing user perceived delays, especially in Web Information Systems and E-Commerce applications in which the quality of the offered services is first judged by the web site responsiveness. A lot of work has been done on studying various approaches to either reduce or hide network latency, in particular *caching* (e.g., [7],[4]) and *prefetching* (e.g., [1],[3]). Caching involves storing frequently accessed or large, static documents closer to users. Proxy-caches, for example, can quickly return requested documents previously accessed by other users in the same organization. Prefetching is the retrieval of documents before a user requests them in anticipation of the user's future requests. In this paper, we will focus on prefetching and in particular on *client initiated prefetching*.

In order to be effective, the prefetching algorithm needs (1) to predict which documents a user might reference next with high accuracy and (2) to take advantage of the idle time between user requests to download the document to the user site. Several methods have been proposed in predicting future references with varying degree of accuracies (e.g., [4]). Orthogonal to the choice of the document to prefetch, is the choice of the rate to prefetch this document. The standard option is to prefetch the document at the maximum rate that the network will support. But this option creates bursty traffic patterns, and increases network congestion. In fact, in a paper by Crovella and Barford paper [1] it was shown that the increase in congestion caused by prefetching can actually increase, and not decrease, user perceived latency. Crovella and Barford [1] proposed a rate-controlled prefetching scheme that utilizes client *OFF times*. OFF time is the time between a client's requests (from the completion of the last request until the initialization of new one):

OFF TIME = <completion time of last request>
- <time of initiation of next request>

During the OFF time the user is typically examining the latest transferred document. In this rate-controlled scheme, the document is downloaded at an even rate over the OFF time. The speed of the client's network connection has little affect on the OFF time, though it is utilized by the prefetching scheme to compute the download rate. In [1] it was shown that using their rate-controlled scheme with a good OFF time predictor would significantly decrease both network congestion and user perceived latency.

Unfortunately, good OFF time prediction does not appear to be easy. Barford and Crovella [1] suggested a scheme based on the size of the previous document, with the rationale being that it takes a user longer to process larger documents. This scheme met with only limited success. In [1] it was shown that the correlation between document size and next OFF time to be around 0.14, that is, these two parameters are not that closely related. As far as we know, there have been no other papers that have attempted to improve on this document size predictor, nor have any proposed alternative prediction methods.

There are many other document characteristics besides document size that potentially might be correlated with

*This work is supported in part by National Science Foundation under grants IIS-9812532 and CCR-0098752.

OFF time, but it was not obvious which document property is most closely related to OFF times. In this paper, we report on the use of standard machine-learning techniques in identifying those documents characteristics that best predict OFF times. Specifically, we apply *neural networks* and *genetic algorithms* [5, 6]. Neural networks are mathematical models based on the biology of the brain that can learn certain tasks including pattern recognition. On the other hand, genetic algorithms are modeled after genetics and evolution. Both techniques learn a function from past behavior for predicting future OFF times. These two approaches were chosen because they tend to perform well with continuous data and data without a large number of features.

To facilitate comparison with the only other predictor, that is, the document size predictor given in [1], we ran our experiments on the same inputs that used in [1]. Our results show that these learned functions performed significantly better than the document size predictor in [1]; On average, the correlations for these learned functions were approximately twice those for the document size predictor, with some having correlation as high as 0.5. Further, our results show that document type is the best predictor of OFF time. The rationale for this is probably that users examine different document types in vastly different ways, e.g. a large image can be assimilated more quickly by a user than a large text file.

Another significant property of our approach is that our functions can be tailored to favor *underpredictions* over *overpredictions*. Intuitively, underprediction can has a less negative effect on network congestion than overprediction. If there is a serious overprediction then when the user requests the document, much of the document still not been downloaded, and then the file must then downloaded at the rate that network can support, once again causing burstiness and network congestion.

The rest of this paper is organized as follows: In the next section, we explain our experimental setup and provide some brief background on Neural Nets and Genetic Algorithms. In Sections 3 and 4, we present the results of our experiments for both methods. We conclude in Section 5 with some comments on performance and future directions for research.

2. EXPERIMENTAL SETUP

2.1 Training and Test Data

The data used in the training and testing of the neural network and genetic algorithm are client web traces from the computer science department at Boston University [2]. Each client web trace contains data for a single user on a specific machine for a given day. The original data was modified into the following vector:

> *< previous OFF time, picture boolean,*
> *compression boolean, text boolean, movie boolean,*
> *other boolean, file size, retrieval time, target output>*

Previous OFF time is the duration of time since the last transfer completed and the current one began. Only one of the boolean values in each vector could be set to true (1). These booleans represented the type of file that was downloaded. Also, data items that had a zero for retrieval time were ignored since those documents were fetched from the client's cache and did not affect the OFF times.

The specific data sets used for training both the neural net and the genetic algorithm were chosen at random from those trace files with a large file size. This was done because we believed large file size would correspond to a large number of records, but as it turned out this was not the case. Instead, we got a variety, with some having many records and others much less.

We performed experiments on a large number of traces from a variety of users in the set. For brevity, in the rest of the paper we report only on the results of two users: Con55 and Con112. These reults are typical for the entire set.

Specifically, we chose the files con55.rooh.798816043 and con112.rolph.792266342 to be used for training. Both of these had more than 200 records. The data sets used for testing the efficacy of the trained Con55 neural net and genetic algorithm were:

- con55.animal.800142146 (64 records)
- con55.tigger.800122654(47 records)
- con55.beaker.791497209 (105 records)

The data sets used for testing the Con112 neural net and genetic algorithm were:

- con112.piglet.791661469 (160 records)
- con112.pooh.791768842 (238 records)

The names of the files come unchanged from the collection of traces. Each file in the collection corresponds to an individual user for a particular day.

2.2 Neural Network

Neural networks are mathematical models based on the biology of the brain. They can learn to perform certain tasks including pattern recognition and function approximation. Similar to networks of biological neurons, the knowledge acquired by these models as a result of the algorithms by which they learn, is gained through the manipulation of connections between the nodes of the network.

In our experiments, we use the QNet 2000 Shareware program to create a three level, feed-forward, network with eight input nodes, three hidden-level nodes, and one output node. Each of the eight input nodes corresponds to one value in the test data vectors described above. The value of the output node after each input vector has propagated through the network is the network's OFF time prediction. The network was trained using the back-propagation learning algorithm which adjusts the network's inter-node connections by using gradient descent to determine the contribution of each connections' weight to the error between the network's actual output on each training vector and the desired output for that vector. The weights are adjusted in order to minimize the network's error over all the training vectors.

The network was trained through 80,000 presentations of training data. After 80,000 presentations the network became over trained meaning that its error began to steadily increase. (This is commonly referred to as *overfitting*[6], and is a problem that affects every kind of learning algorithm. It occurs because in a large search space, irrelevant attributes in the training data can appear to be patterns even though in the whole population they are actually not.)

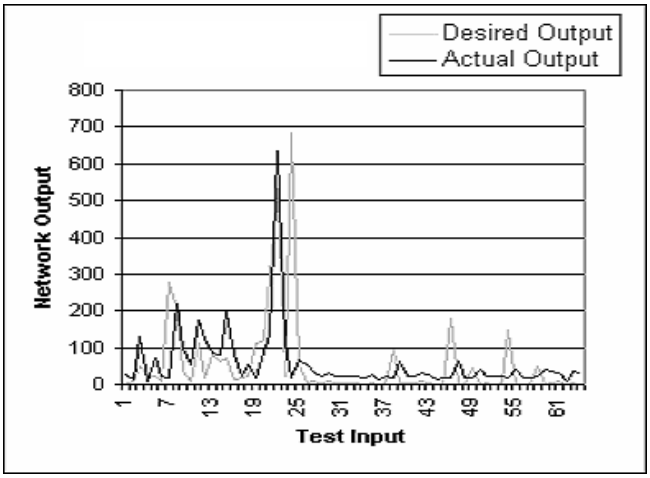

Figure 1: Con55 Animal

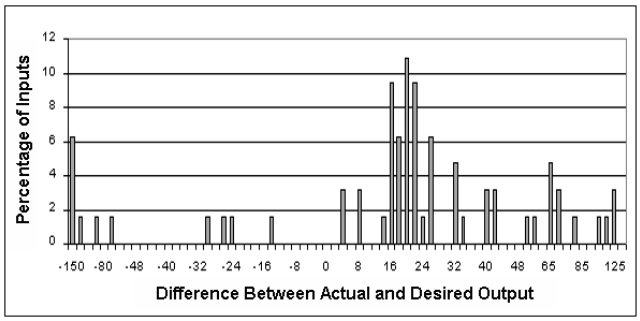

Figure 2: Con55 Animal Error

2.3 Genetic Algorithm

A genetic algorithm (GA) is a technique for searching a large space more quickly. Such algorithms are modeled after genetics and evolution. They incorporate principles of inheritance, mutation, and natural selection. A population of individuals, each a different possible solution, is generated.

Each individual is assigned a *fitness* value based on a predetermined fitness function. Based on the *selection rate*, a certain percentage of those with highest fitness survive to the next generation. These survivors reproduce via crossover of genes to replace the dead ones, and a percentage of the offspring are mutated according to the *mutation rate*. Fitness is calculated again, and the process repeats until a desired fitness is reached or a specified number of generations have elapsed.

The GA we created learns a vector of nine real numbers $< C1, ..., C9 >$, that are used in the following prediction function:

$$Prediction = C1 * F1 + C2 * F2 + C3 * F3 + C4 * F4 + C5 * F5 + C6 * F6 + C7 * F7 + C8 * F8 + C9$$

Where F1 through F8 are the known data features (F1 = previous OFF time, F2 = picture boolean, F3 = compression boolean, F4 = text boolean, F5 = movie boolean, F6 = other boolean, F7 = file size, F8 = retrieval time).

After running a series of tests on various settings for population size, number of generations, selection rate, and mutation rate, the following were determined to perform best and used in our experiments:

- Population Size of 500
- 2000 Generations
- Selection Rate of 30%
- Mutation Rate of 90%

An individual's fitness was defined as the difference between the actual OFF time and the prediction, averaged over the the training set. Thus, a higher fitness would be a number closer to zero.

3. EXPERIMENTAL RESULTS: NEURAL NETWORK

3.1 Experiment 1

The results of the Con55 training session, corresponding to user Con55, are summarized in the following table. The table shows the percentage of contribution of each data feature to the value of the output node. Recall that these features correspond to the parameters of the vector capturing the properties of a document.

Input Node	% Contribution
Previous OFF time	3.79
Picture boolean	16.17
Compression boolean	17.81
Text boolean	10.24
Movie boolean	8.79
Other boolean	31.30
File size	4.12
Retrieval time	7.78

As can be seen, the Con55 network based its OFF time predictions primarily on the type of the previously downloaded file captured by the boolean values.

As mentioned above, the Con55 network was tested for three different daily sessions of user Con55. The results of the first test of the Con55 network for the Animal data set are shown in Figures 1 and 2. Figure 1 compares the actual output values (the predicted OFF time) of the Con55 network to the desired output values (the real OFF time). We can see a number of peaks in the two data series that are aligned. The correlation between the two data series is 0.517314 with a probability of obtaining the results with random data (referred to from now on as the p-value) of 0.001.

The distribution of error over the test inputs, the difference between actual and desired output, is shown in Figure 2. Negative values indicate inputs where the algorithm was underpredicting, while positive values represent overpredicting. In this case, the neural network is clearly overpredicting, mostly between 16 and 24 seconds.

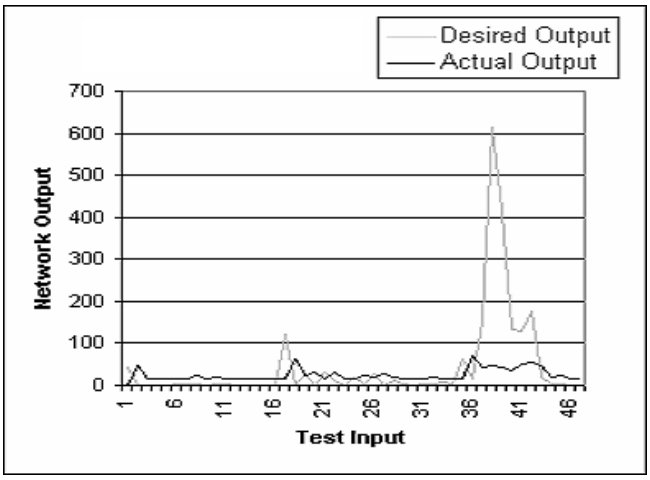

Figure 3: Con55 Tigger

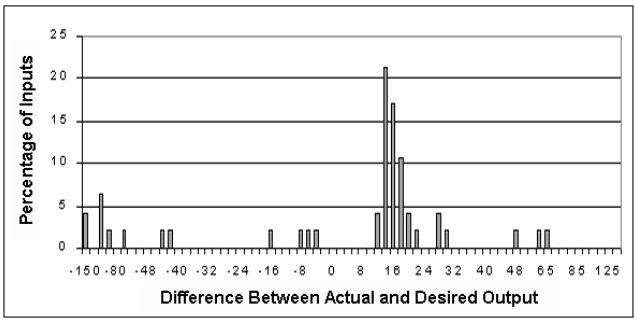

Figure 4: Con55 Tigger Error

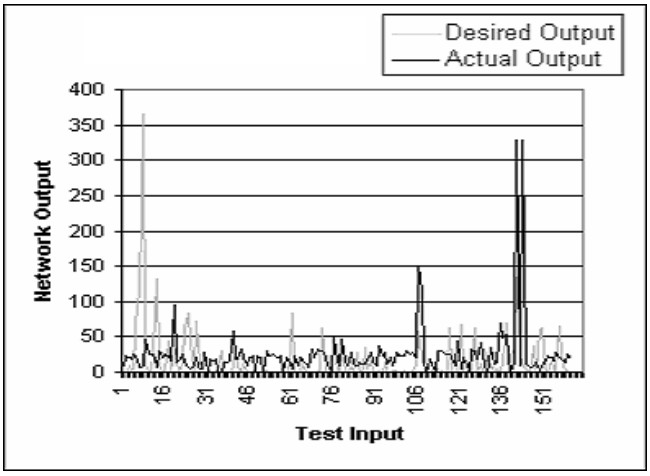

Figure 5: Con112 Piglet

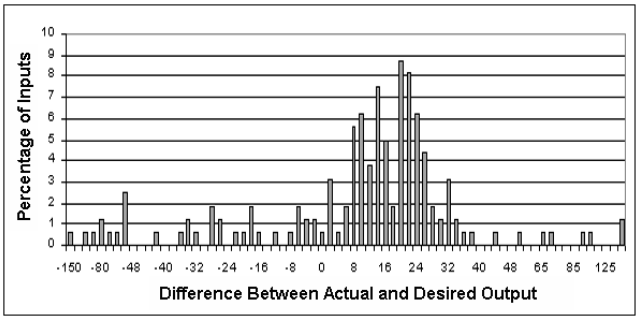

Figure 6: Con112 Piglet Error

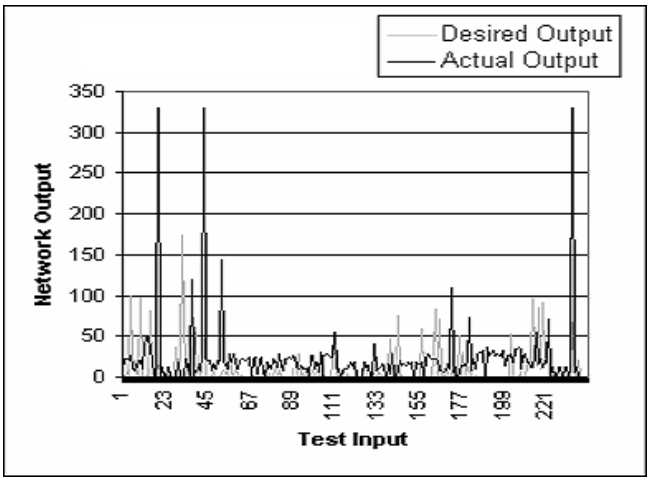

Figure 7: Con112 Pooh

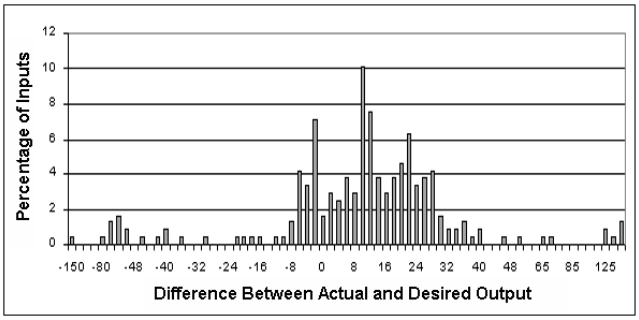

Figure 8: Con112 Pooh Error

The results for the second test on the Con55 Tigger data set are shown in Figures 3 and 4. The relationship is less obvious visually, yet there is a relatively high correlation of 0.416825 and 0.01 p-value. The error is less than the first case, although the neural network is still overpredicting, mostly around 16 seconds.

For the last test set, Con55 Beaker, we got similar behavior as in the first two ones although lower correlation. The neural network is overpredicting around 16 seconds, and correlation is 0.202934 with a p-value of 0.05.

3.2 Experiment 2

For the Con112 user, the results of the training session are summarized in the following table. The table shows the percentage of contribution of each data feature to the value of the output node.

Input Node	% Contribution
Previous OFF time	16.84
Picture boolean	13.36
Compression boolean	2.67
Text boolean	4.99
Movie boolean	4.89
Other boolean	8.04
File size	24.04
Retrieval time	24.87

The table shows that, as opposed to the Con55 network which based its prediction on file type, the Con112 network based its predictions for user Con112's OFF times primarily on the previous OFF time and the size and retrieval time of the previously downloaded file.

The generated neural network was tested on two different daily sessions of user Con112. In the Con112 Piglet graph, Figure 5, we see some alignment of peaks and valleys but the two data series have a low correlation of 0.153071 with a high p-value of approximately 0.06. Error is shown in Figure 6. As in the case of the Con55 network, we again observe more overprediction rather than underprediction.

The results of the second test case on the Con112 Pooh data set is shown in Figures 7 and 8. There is a correlation of 0.126419 and a high p-value around 0.06. Error is similar to the first test.

These relatively low correlations are equivalent to those found in [1]. One possible explanation is a difference in user behavior from that exhibited in the training set.

4. EXPERIMENTAL RESULTS: GENETIC ALGORITHM

4.1 Experiment 1

The genetic algorithm (GA) learned the following vector for the Con55 user:

$< 0.00, 1.68, 19.51, 1.46, 5.17, 2.50, 0.00, 2.02, 11.72 >$

This means that the algorithm based its predictions mostly on the type of file that was downloaded last. While there appears to be an emphasis on compressed file type (C3=19.51), this is not actually the case. After some thought and examination of the training data, we came to realize that very few of the training inputs actually contained a true value for the compression boolean. This means that most of the training had that coefficient multiplied by zero (false), thus rendering the coefficient practically useless for the prediction. With no information to guide the learning, the value of the coefficient became a meaningless random number. Interestingly, the coefficient for the file size is 0.00 meaning that no importance was placed on it at all.

As in the case of the neural network, we perform three tests using the same daily sessions of user Con55. Figure 9 shows the performance of the Con55 genetic algorithm on the Animal test data. The correlation between the desired output of the algorithm and its actual output is 0.530229 with a p-value of 0.001. Error, shown in Figure 10, is less diverse than in the Neural Net and is concentrated around an overprediction of 16 seconds. The GA primarily ranges between -150 and 85 seconds as opposed to -150 and 130 seconds.

For the Tigger test data, the correlation between the desired and actual output shown in Figure 11 is 0.230931259 with a p-value of 0.1. The error is shown in Figure 12, and again appears concentrated around 16 seconds but with significantly better range (-150 to 24 seconds).

The last test data set for the Con55 algorithm, the Beaker data, is shown in Figures 13 and 14. The data series have a 0.395252112 correlation with a p-value of 0.001. The error is similar to the second test.

4.2 Experiment 2

The vector that the genetic algorithm learned for the Con112 user is:

$< 0.08, -4.19, -32.26, 1.50, 6.76, -1.38, 0.00, 0.98, 4.61 >$

For this user, the algorithm again gives more emphasis to file type than to the other features. As stated previously, the large negative coefficient on the compressed boolean is insignificant. File size is once again being ignored.

We perform two tests for this user on the same test data as in the case of the neural network. These tests produced the best results for the GA. The graphs for the Con112 test data sets are shown in Figures 15, 16, 17, and 18.

The correlation between the desired and actual output for the Pooh test is 0.233832703 and 0.274605 for the Piglet test. Both have a p-value of 0.001. Most of the error for each test is around 0 seconds, meaning that most of the time this GA has no actual and predicts the correct OFF time.

4.3 Biased Learning

The concentration of error in the overpredicting range intrigued us in both the neural network and the GA. If the error cannot be eliminated, it would be preferable for the error to be in the underpredicting range, as close as possible to zero. Underprediction affects the individual user but has less impact on the overall network performance.

As a further test, we decided to try biasing the genetic algorithm's learning. We hoped that this would shift the bulk of the error shown in the graphs to the left, closer to zero. The biasing was done quite easily by modifying the fitness function to include information about the percentage of overpredictions. Fitness was scaled by a factor proportional to this percentage. The results were very encouraging. We ran the new learner on both the Con55 and Con112 user.

The new vector learned for the Con55 user was:

$< -0.03, 2.05, 20.68, 3.71, 9.54, 2.13, 0.00, 1.13, -2.59 >$

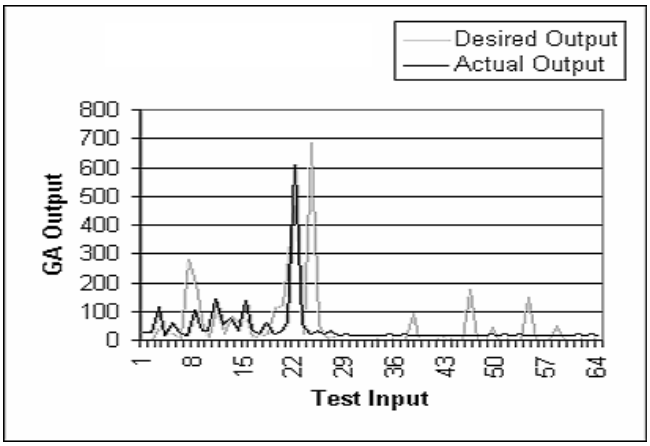

Figure 9: GA Con55 Animal

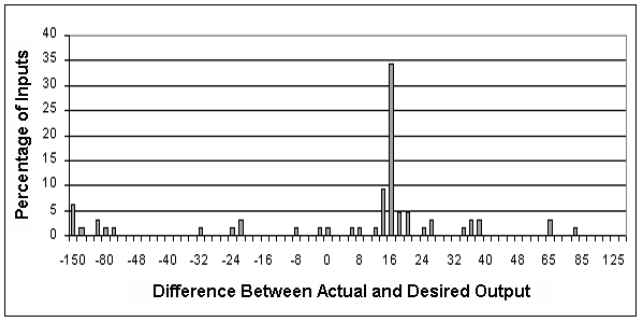

Figure 10: GA Con55 Animal Error

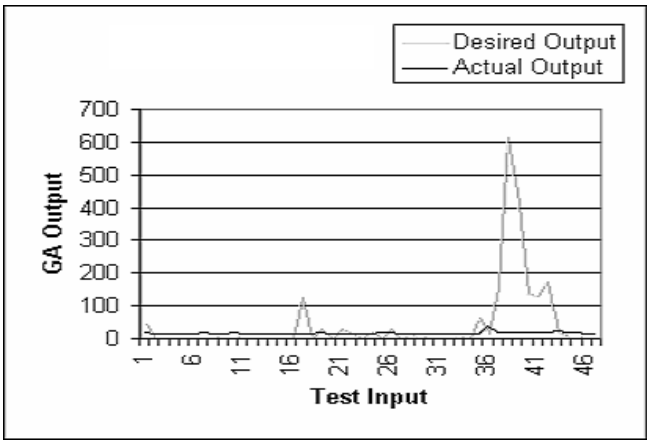

Figure 11: GA Con55 Tigger

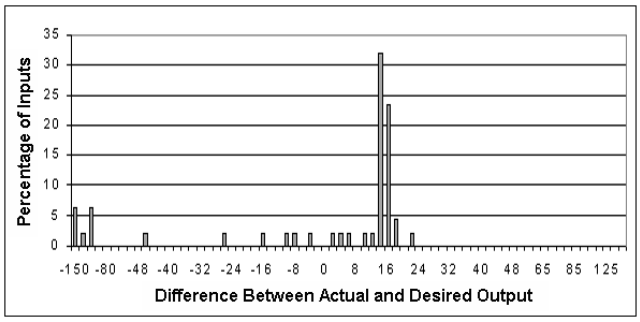

Figure 12: GA Con55 Tigger Error

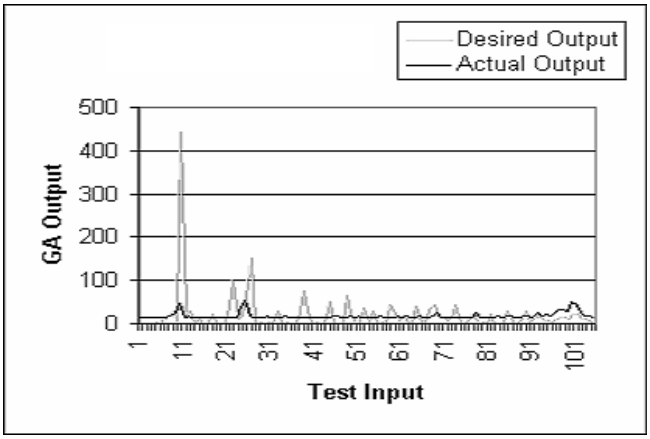

Figure 13: GA Con55 Beaker

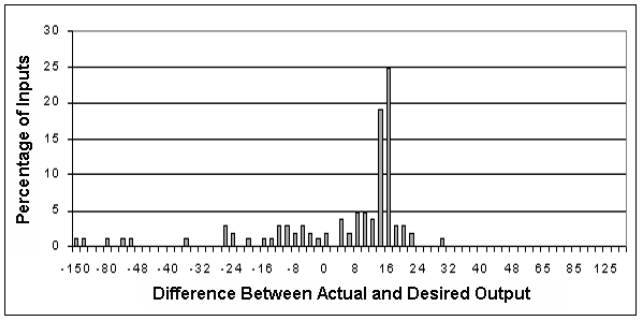

Figure 14: GA Con55 Beaker Error

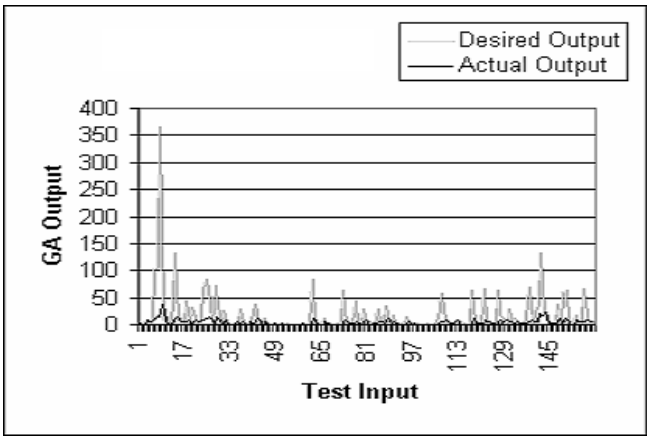

Figure 15: GA Con112 Piglet

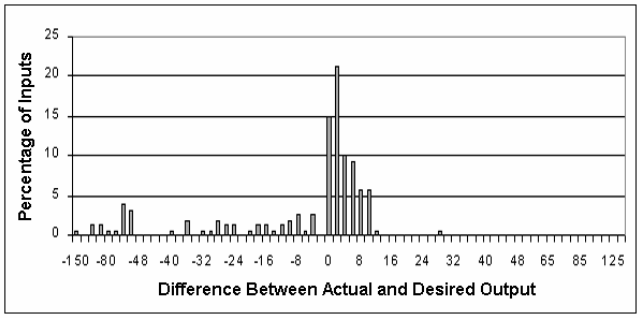

Figure 16: GA Con112 Piglet Error

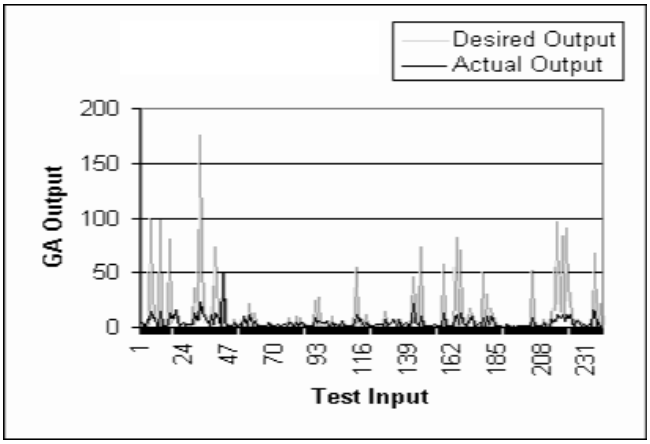

Figure 17: Con112 Pooh

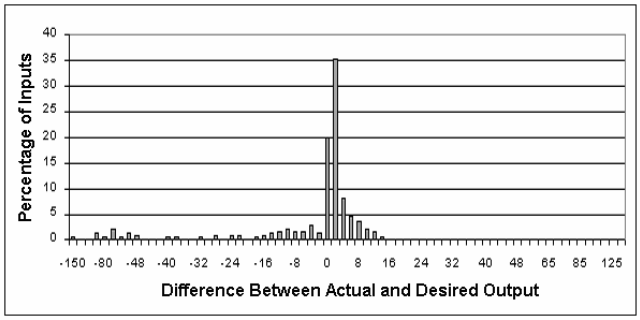

Figure 18: Con112 Pooh Error

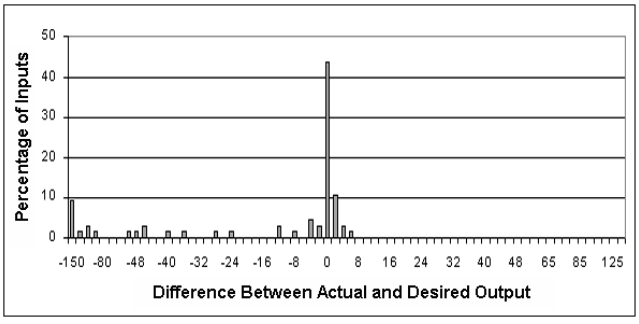

Figure 19: Con55 Animal Error w/Bias

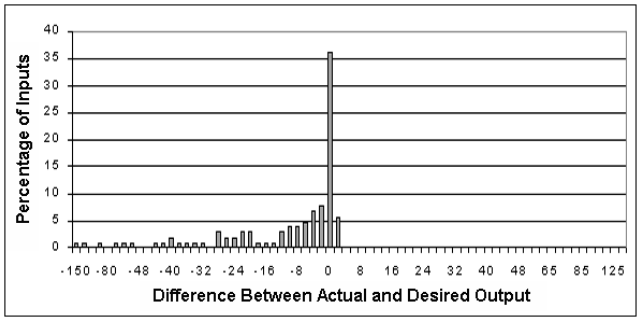

Figure 20: Con55 Beaker Error w/Bias

and for the Con112 user:

$$< 0.08, -3.70, -17.42, 1.57, 4.09, -0.89, 0.00, 0.97, 4.13 >$$

Although the coefficients are different than before, the algorithm is still basing its predictions on file type and ignoring file size.

The graphs of the error for the Con55 Animal and Beaker data sets are shown in Figures 19 and 20. As can be seen, there is very little positive error, and most of the error is now concentrated directly above zero. For both data sets, the correlation actually increased very slightly by about 0.01.

The Tigger data set had a similar shift in error, but its correlation dropped significantly. This set also had the lowest correlation with the unbiased learner, and could be an example of user behavior differing from that exhibited in the training data.

For the Con112 user, the bias had only minimal effect, with a very slight shift in error and virtually no change in correlation. The original function found by the unbiased fitness function is already fairly strong in this respect.

5. CONCLUSIONS

The innovation of this paper is the use of machine learning techniques in predicting user OFF times. These OFF times can be utilized by a rate-controlled prefetching scheme to improve network performance while reducing individual user perceived latency. This will have a significant impact on the overall performance of a Web Information System or an E-Commerce application.

In our experiment we used two different machine learning techniques, namely, *neural networks* and *genetic algorithms*. The results of the neural network's performance were surprising to us. We had expected the network that had learned to predict the next OFF time based on the previous OFF time, file size, and retrieval time to perform much better than one that used the file types as its primary inputs. A possible explanation for this behavior is that the amount of time that a client spends examining the downloaded material may be dependent on the content (type) of file.

The genetic algorithm's primary prediction methods did not vary quite like the neural network's. It based its prediction primarily on file type for both users, while the neural network used file type for only one of the users presented. Our additional test sets exhibited similar results. The genetic algorithm's behavior supports further the hypothesis that the type of a document plays an important role in determining the duration between successive requests.

It is clear from our results that machine learning techniques do in fact perform better than the prediction method used by Mark Crovella and Paul Barford in [1]. In fact, both of our algorithms performed better. The neural network and genetic algorithm each had an average correlation approximately twice that of the old method, which only averaged 0.14. Of our two methods, the genetic algorithm appears to perform slightly better overall. We came to this conclusion because it does much less overpredicting and can be easily biased to do even less. We believe it is better to underpredict OFF time as overprediction could result in increased burstiness and network latency. If OFF time prediction is tied to the number of data documents to be prefetched, the overprediction might unnecessarily overload the Web Information System with requests that are less likely to be used. Correlations produced by the two methods were practically the same.

One area still to explore is retraining. This could be done periodically as the user works, based on either time elapsed or amount of data collected since the last training. The old learned function would provide the starting point for the learning algorithm to make modifications based on the new user data. Such retraining would ensure the algorithm kept up with the changing nature of the user and the network. As an initial investigation, we ran some additional experiments, training on data from one month and testing on data from 6 months later. Our techniques still showed good predictions and similar correlations.

6. REFERENCES

[1] M. Crovella and P. Barford. The Network Effects of Prefetching. In *Proceedings of IEEE INFOCOM*, San Francisco, CA, 1998.

[2] C. Cunha, A. Bestavros, and M. Crovella. Characteristics of WWW Client Traces. *Boston University Department of Computer Science Technical Report TR-95-010*, April 1995.

[3] B.D. Davison and V. Liberatore. Pushing Politely: Improving Web Responsiveness One Packet at a Time. In *Proceedings of PAWS00*, 2000.

[4] L. Fan, P. Cao, and Q. Jacobson. Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performance. In *Proceedings of SIGMETRICS*, 1999.

[5] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[6] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 1994.

[7] B. Williams. Transparent Web Caching Solutions. In *Proceedings of WCW'98*, June 1998.

[8] A. Wolman, G. Voelker, N. Sharma, N. Carwell, A. Karlin, and H. Levy. On the scale and performance of cooperative Web proxy caching. In *17th Symposium on Operating Systems Principles*, December 1999.