A Soft Computing Prefetcher to Mitigate Cache Degradation by Web Robots

Ning Xie*, Kyle Brown*, Nathan Rude, and Derek Doran

Dept. of Computer Science & Engineering, Kno.e.sis Research Center
Wright State University, Dayton, Ohio, USA
{brown.718,xie.25,howard.rude,derek.doran}@wright.edu

Abstract. This paper investigates the feasibility of a resource prefetcher able to predict future requests made by web robots, which are software programs rapidly overtaking human users as the dominant source of web server traffic. Such a prefetcher is a crucial first line of defense for web caches and content management systems that must service many requests while maintaining good performance. Our prefetcher marries a deep recurrent neural network with a Bayesian network to combine prior global data with local data about specific robots. Experiments with traffic logs from web servers across two universities demonstrate improved predictions over a traditional dependency graph approach. Finally, preliminary evaluation of a hypothetical caching system that incorporates our prefetching scheme is discussed.

1 Introduction

A Web robot is an autonomous agent that sends HTTP requests to web servers around the world. Recent studies, including our own [16], indicate that upwards of 60% of the traffic faced by web servers comes from robots [17], while only 20% of traffic came from web robots a decade ago [13]. This rise in traffic may come from the necessity for services to retrieve share-in-the-moment news and social data [15]. Moreover, internet-of-things devices will increase this proportion as more devices which operate autonomously are connected to the web.

Prefetching web resources for caching and content management systems is a common technique to anticipate and pre-load the resources likely to be requested next for fast, low latency access [14, 8, 3]. Prefetchers for human traffic are an essential component of web caches, but as robots exhibit different functionality [5], access patterns [7], and traffic characteristics, traditional prefetching strategies applied to traffic with high levels of web robot activity exhibit degraded performance. The degree of uncertainty and few restrictions on robot behavior requires powerful soft computing techniques to find and utilize the possibly weak latent patterns existing in their requests. This paper proposes a prefetcher with such techniques for robot-laden web traffic. Evaluations show that the synergistic use of a deep recurrent neural network (RNN) and a Bayesian network greatly improves prefetching performance for robot traffic.

* Ning Xie and Kyle Brown are joint first authors of this paper.

2 Related Work

There have been many previous studies which have examined the characteristics of web robot traffic. Doran et al. [7] study the distribution of request types (e.g. image files vs web pages) for human and robot traffic. The authors note that web robots have a strong penchant for larger resources; they are 10 times more likely to request resources larger than 10MB when compared to human traffic. Rude et al. [16] develop an Elman Neural Network to predict global trends of request types for robot traffic. Almeida et al. [1] investigate the impact of web robots on cache hit rate. The authors examined the referencing pattern of web robots and found out the pattern exhibits "round-robin" traits, disrupting locality assumptions which can leave an LRU cache useless. In [5], the authors classify web robots according to their functionality and request patterns. In [2], the authors suggest web servers should implement new interfaces for robot traffic that provide metadata archives describing the content. By querying the metadata instead of requesting all resources from some precomputed list, a robot is able to narrow its selection down to a more refined set of resources reducing the amount of bandwidth consumed. Li et al. [12] propose a hybrid cache design that is broken down into 3 sections: (i) an *index cache* which stores inverted indices for user search queries (ii) a *results cache* which caches a results page returned for a unique user search query and (iii) a *document cache* to cache documents on the server. However, the authors filter out robot traffic to better understand the user behavior and queries.

3 Prefetching Scheme

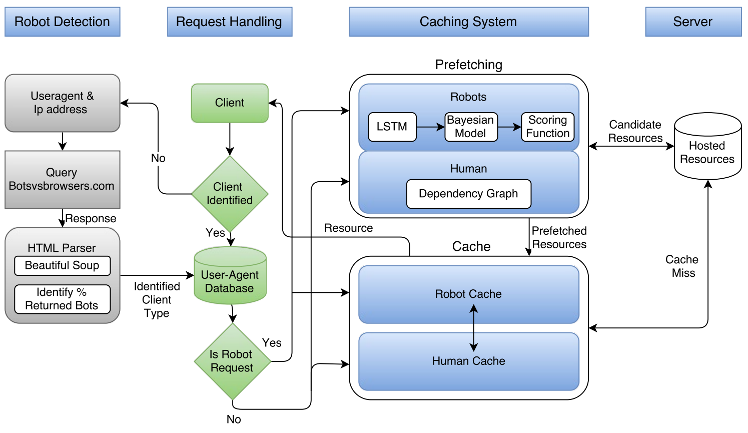


Fig. 1: A caching architecture with robot request prefetching

This section introduces the design of the request prefetcher. Its design and integration into a hypothetical caching architecture is illustrated in Figure 1. After a request is processed, it is labeled as originating from a robot or a human through a real-time detection algorithm (of which many exist in the literature [6]). If the request is labeled as a robot, the proposed prefetcher will predict what subsequent robot requests will be made. To identify the more subtle patterns that exist in robot request sequences, prefetching is done by combining a deep recurrent neural network (RNN) with a Bayesian network model. The RNN is used to predict the 'orientation' of the robot request stream by predicting the subdirectories future robots will likely visit next given their past history. From the k subdirectories the RNN predicts robots will visit next, a Bayesian network selects the subset of resources within those directories to be prefetched. The Bayesian network incorporates prior information in the form of global web robot patterns, as well as the request patterns of the particular web robot who made the last request, to tailor the prediction to future requests from that robot. This idea is motivated by the idea that robot request sequences often come from the same robot, who send a number of requests within a short period of time.

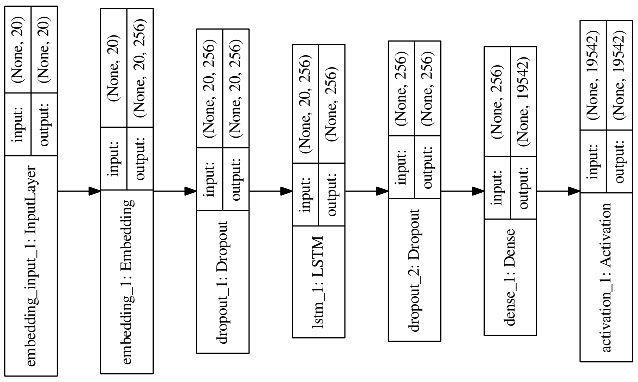


Fig. 2: RNN structure to predict subdirectory requests

RNN Specification. The RNN architecture is presented in Figure 2. It has an embedding layer that translates ordered sequences of the subdirectories robots visited to an input, a long short-term memory (LSTM) layer that learns sequence patterns, dropout layers that control for overfitting, and a fully connected layer to compute the likelihood a subdirectory will be visited next. An LSTM layer is often used for sequence processing since LSTMs can hold "memory" over long periods [10, 9]. The RNN was trained with dropout parameters set to 0.2. Validation loss was monitored with a patience of two. Training ended when no improvement in loss over a validation set was observed or after 128 epochs (passes over the training data).

Bayesian Networks. Two Bayesian models were developed to predict which resources a robot will request next in a given set of subdirectories. The first model, called the simple model (SM), first draws a resource type from a multinomial distribution and then an individual resource from another multinomial distribution corresponding to the set of resources of the drawn type. Request types are modeled based on past work that demonstrated request type preferences for Web robots [16, 11]. Prior knowledge is given by hyper-parameter settings for the resource type and resource request distributions, which considers the number of times we observe *any* robot requesting resources in a subdirectory. This is useful if a specific robot has not made many requests to the web server in the past. The generative process and its hyper-parameters are shown in Figure 3 using the notation in Dietz [4].

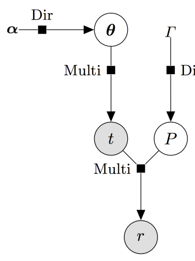


Fig. 3: Bayesian network of the Simple Model

The data likelihood of the simple model is $\Pr(r_1, \cdots, r_M, t_1, t_M | \boldsymbol{\theta}, P) = \exp\{\sum_{j=1}^{K}(m_j \log(\theta_j) + \sum_{l=1}^{R_j} n_{j,l} \log(p_{j,l}))\}$ where M is the total number of observed requests by the robot in this subdirectory, K is the number of resource types, m_j is the number of requests for a resource of type j by the robot in this subdirectory, θ_j is the multinomial parameter for resource type j, R_j is the number of resources of type j in this subdirectory, $n_{j,l}$ is the number of times the l-th resource of type j in this subdirectory was requested by the robot, and $p_{j,l}$ is the l-th component of the multinomial parameter vector $\boldsymbol{p}_j$ for resources of type j in the subdirectory. The parameter vector $\boldsymbol{\theta}$ of resource types is drawn from $\boldsymbol{\theta} \sim \text{Dirichlet}(\boldsymbol{\alpha})$ and for each resource type j, the parameter vector is drawn from $\boldsymbol{p}_j \sim \text{Dirichlet}(\boldsymbol{\gamma}_j)$. The values of the hyper-parameters $\boldsymbol{\alpha}$ and $\Gamma = \{\boldsymbol{\gamma}_j\}_{j=1}^{K}$ are chosen using global statistics from all robots by $\alpha_j = \alpha m_j^{(g)}/M^{(g)}$, where α is the prior strength for $\boldsymbol{\alpha}$, so that $\sum_{j=1}^{K} \alpha_j = \alpha$, $m_j^{(g)}$ is the global number of requests for resources of type j, and $M^{(g)}$ is the global number of requests. $\boldsymbol{\gamma}_j$ is chosen as $\gamma_{j,k} = \gamma n_{j,k}^{(g)}/m_j^{(g)}$ where $n_{j,k}^{(g)}$ is the global number of requests for the k-th resource of type j, and γ is the prior strength for Γ.

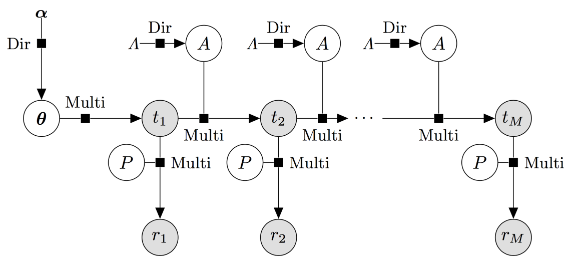


Fig. 4: Bayesian network where request types are generated by a Markov process

A second model is an extension of the simple model, where resource types are now generated by a Markov process instead of draws from a multinomial distribution. It attempts to consider patterns in the request type sequence of a robot, which may be helpful when robots are only interested in specific kinds of resources (e.g. an image scraper) [16]. For the Markov model, an "observation" is not just a single resource-type pair, but a sequence of resource-type pairs. The generative process for this model is depicted in Figure 4. We assume there are L such observations, and that the i^{th} sequence has length M_i and is represented by $(r_1^{(i)}, t_1^{(i)}), \cdots, (r_{M_i}^{(i)}, t_{M_i}^{(i)})$. The entire set of observations is denoted by $\mathcal{R}$. Then the data likelihood for the Markov model can be written as: $\Pr(\mathcal{R}|\boldsymbol{\theta}, P, A) = \exp\{\sum_{j=1}^{K}(m_j \log(\theta_j) + \sum_{k=1}^{K} T_{j,k} \log(a_{j,k}) + \sum_{l=1}^{R_j} n_{j,l} \log(P_{j,l}))\}$ where m_j is the number of times an observation started with a request for a resource of type j, $T_{j,k}$ is the number of transitions from type j to k within an observation were observed, and $n_{j,l}$ is the number of requests for the l-th resource of type j over all observations. The other symbols are the same as in the simple model. The parameters $\boldsymbol{\theta}$ and P in the Markov model are assumed to be generated from the same distributions as the simple model. Each row $\boldsymbol{a}_j$ of the transition matrix A is drawn from $\boldsymbol{a}_j \sim \text{Dirichlet}(\boldsymbol{\lambda}_j)$ for $1 \leq j \leq K$. Here $\boldsymbol{\lambda}_j$ is the j-th row of the hyper-parameter matrix Λ. As for the simple model, hyper-parameters for the Markov model are computed from global statistics for all robots. $\boldsymbol{\alpha}$ and Γ are computed the same way as the simple model. Each element $a_{j,k}$ of A is computed as $a_{j,k} = aT_{j,k}^{(g)}/T_j^{(g)}$ where $T_{j,k}^{(g)}$ is the global number of transitions from a resource of type j to type k, $T_j^{(g)} = \sum_{k=1}^{K} T_{j,k}^{(g)}$ is the global transition count from a resource of type j, and a is the prior strength of A.

Parameter estimation for all models was done using maximum *a posteriori* estimation (MAP) because of the ability to obtain a closed-form solution for parameters for the simple and Markov models. Since the models need to be updated as requests come in to the web server, this approach was used to enable an efficient implementation. The MAP parameter values for the simple model are $\tilde{\theta}_j = (\alpha_j + m_j - 1)/(\alpha + M - 1)$ for $1 \leq j \leq K$ and $\tilde{p}_{j,l} = (\gamma_{j,l} + n_{j,l} - 1)/(\Gamma_j + m_j - 1)$ for $1 \leq j \leq K$ and $1 \leq l \leq R_j$, where $\Gamma_j = \sum_{l=1}^{R_j} \gamma_{j,l}$.

4 Prefetching Evaluation

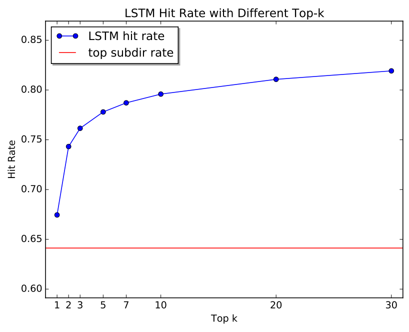
To evaluate the models, Web logs across all web servers at Wright State University (WSU) and the University of Pavia were collected. The WSU logs represent a 3 month period in 2016 while the Pavia logs span a 5 month period over 2014. Robot traffic was extracted from the logs by following a simple heuristic procedure using the crowdsourced database *botsvsbrowsers*[1]. Checking the user-agent of each request in the log against the nearly 1.5 million agents recorded on this database, a user-agent match to a known robot is used to mark the session as a robot. While many probabilistic detection methods exist [6], this heuristic approach guarantees that we only evaluate the prefetcher on true robot traffic while still giving us a sizable number of sessions. Specifically, 221,683 and 14,401 robot sessions were extracted from WSU and the University of Pavia logs, respectively. The models were trained on two-thirds of earlier arriving sessions from each dataset, while the remaining third was used for evaluation.

As a baseline for comparison of the system, a multinomial distribution was fit over all resources in the directory. In the plots below, this is called a *base*

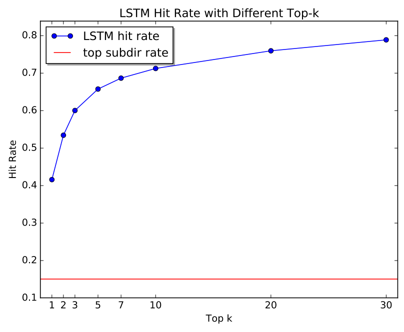
[1] www.botsvsbrowsers.com

model. To show the improvement of the Markov model when using prior information it was fit using both MLE and maximum a-posteriori estimation (MAP). Evaluation was carried out by: (i) examining the top-k accuracy of the RNN for subdirectory prediction; (ii) comparing the performance of the Bayesian models for representative subdirectories on both servers; and (iii) testing the prefetchers capability to accurately identify the next request a robot in a caching system.

RNN Evaluation. RNN evaluation was carried out by checking its top-k accuracy, which is defined as the percentage of time the true subdirectory visited next by a robot is among the k most probable subdirectories predicted by the RNN (called a *hit*). These accuracies are compared against a simple predictor that always predicts the most commonly requested subdirectory as the next one a robot will visit. the empirical probability of the most frequently requested subdirectory in the in Figure 5[2]. They both show promising results as the RNN is able to identify the subdirectory of the next resource to be requested 68% of the time on WSU and 42% of the time over the Univ. of Pavia data. If we allow the RNN to suggest $k = 2$ subdirectories to the Bayesian model, the accuracy substantially improves to 74% and 54%, respectively. It is interesting to note that the accuracy of subdirectory predictions taper off as we let $k > 5$, where the RNN sports accuracies of 77% and 66% on the two servers. This is a desirable property for cache prefetching; the RNN should submit a minimum number of subdirectories to the Bayesian model, thus minimizing the number of resources it needs to predict requests for.



(a) WSU



(b) University of Pavia

Fig. 5: RNN subdirectory prediction performance

Bayesian Model Evaluation. Bayesian models were evaluated by checking its average top-k accuracy for predicting resources across each subdirectory of the WSU and Univ. of Pavia logs. These averages were taken over models for all robots seen making at least one request to a given subdirectory. Comparisons of the model within subdirectories of the same web server and between the different web servers provides information about when the models perform well or poorly.

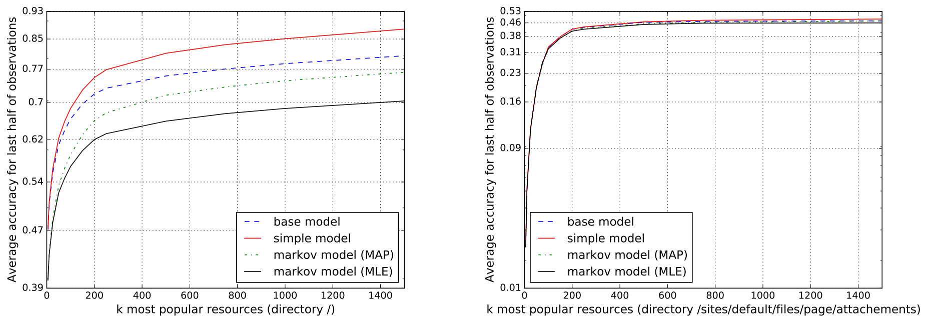
[2] The RNN is labeled *LSTM* in the figures.

Figure 6 compares the performance of the Simple Model to the Markov Model (with parameters fitted using both MAP estimation and MLE) and to a 'base model', which is defined as the prediction by drawing from a multinomial distribution fitted to the proportion of times each resource was requested in the training data, to two directories on WSU. The 'base model' represents a basic empirical approach for predicting requests made by a single robot with only information about that robot. The WSU root directory is extraordinarily popular, with robots requesting resources from it over 60% of the time, and it contains around 5% of the resources on the server. Despite the fact that about a thousand resources are located in this directory, Figure 6a shows how the large number of requests provide many observations from various robots to yield a rich prior distribution that allows the Simple Model to make better predictions compared to the base model. It is also interesting to note how poorly the Markov Model performs, even compared to the Base Model. This may be because the Markov Model seeks to fit resource type patterns that are common to both global and local robot traffic when no such pattern exists. For example, if the root subdirectory contains an even distribution of resource types, it may be the case that the Markov Model's attempt to find patterns in resource type sequences may add noise that reduces performance.

Figure 6b shows a comparison of the models against a different, less popular subdirectory on the website. In contrast to the root directory, we find that both Bayesian models perform almost identically to the Base Model. Such a pattern may emerge when very few requests from all robots are made to the subdirectory, since the Simple Model reduces to the Base Model when using MLE, which could occur when the prior information is the same as the observations for the current robot, or if there is no prior information. This suggests that the power of the Bayesian models relies on observing a large number of requests from a diverse set of robots in a subdirectory. It may also imply that, out of a risk for admitting a number of resources that have low probability of being requested, prefetching resources from some directories should not be done at all.

Figure 7 examines the performance of the Bayesian model over subdirectories at the University of Pavia. This web server represents the interesting condition where the simple MLE-based Base Model for request prediction performs astonishingly well (with 96% prediction accuracy on the root and on a seldom requested image directory). This may be due to the different structure of the University of Pavia's website along with the smaller size of the dataset. As the top-k plots tend to plateau quickly, this might indicate that there are fewer resources commonly requested by robots for each subdirectory. Having a smaller dataset size would provide less prior information, reducing the ability of the Bayesian models to outperform the Base model.

Despite this, the Simple Model is still able to over-perform the Base Model by fusing together data about global and specific-robot patterns. As with the WSU subdirectories, the Markov Model still underperforms the Base and Simple Models in the root directory. This probably indicates that there is little pattern between the resource types of robot requests in high-traffic directories such as the root directory. Note that the Markov Model fit with MAP performs as well as the Simple Model in the `/contents/instance1/images` directory (Figure 7b), which could indicate the lack of a pattern that is picked up on by the Markov Model, i.e. the probability of requesting a resource of a certain type given the previous type is always the same no matter the previous type and matches the type probabilities predicted by the Simple Model.

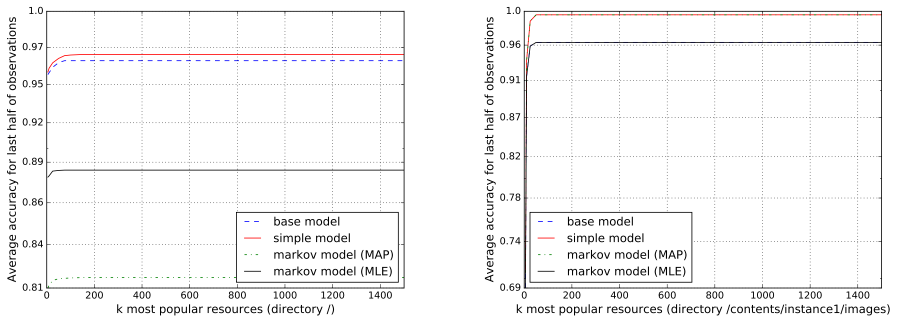


(a) Root directory, WSU (b) Attachment subdirectory, WSU

Fig. 6: Bayesian model performance, WSU subdirectories

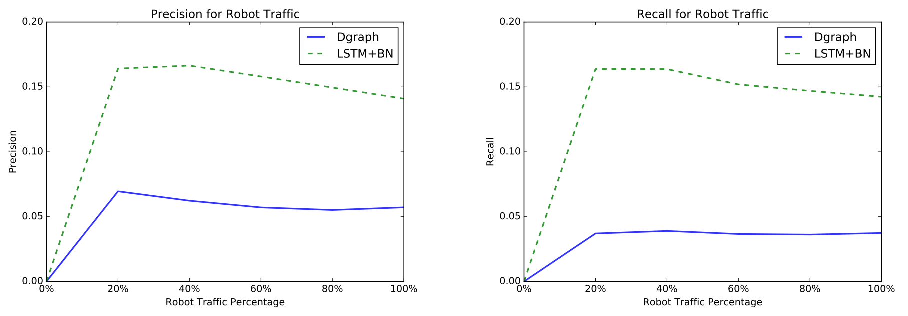
Implications for Web Caching. Next, the implications of this prefetcher to a caching system like the one shown in Figure 1 is investigated. In this preliminary study, the actions of this architecture were simulated over the later 1/3 of the WSU logs (the same data used for testing the RNN and Bayesian models). The cache limit was set to 100MB, of which 80MB was reserved to cache human traffic and 20MB was reserved to store requests predicted to be requested by web robot traffic by the proposed prefetcher. Experiments were ran with varying levels of robot sessions in the data by interweaving a random oversampling of human sessions that were extracted by the *botsvsbrowsers* identification scheme.

The performance of the prefetcher is compared to a dependency graph, a popular and often used Markov model for prefetching resources in caching systems. Figure 8 compares the precision and recall of these two prefetching approaches over robot traffic. Here, precision is defined as the number of times the prefetcher successfully prefetched the resource requested next divided by the total number of prefetches that were made. Recall is defined as the percent of time that a prefetched resource still in the cache was requested by *any* robot. Recall measures instances where, for example, a resource may be requested multiple times after it is added to the cache, or a resource prefetched but not requested next by a robot was eventually requested in the future while it was still in the cache. The top-k subdirectories considered and the top-n resources chosen by the soft prefetched were set to $k = 3$ and $n = 2$ following a parameter sweep of different values, choosing the pair that yielded the highest F_1 score. In these settings, Figure 8 shows how the precision of a dependency graph is $\sim 5.5 - 7\%$ and its recall is 4% regardless of how many robots are visiting the cache. The soft prefetcher, however, enjoys a 2-4x gain in performance, leading to significant improvements in the efficiency of a web cache. Moreover, the soft prefetcher is able to maintain these strong values even in the face of extraordinarily high levels of Web robot traffic, which the web may experience in the future.



(a) Root directory, Univ. of Pavia (b) Image subdirectory, Univ. of Pavia

Fig. 7: Bayesian model performance, Univ. of Pavia



(a) Robot prefetching precision (b) Robot prefetching recall

Fig. 8: Cache prefetching performance: soft prefetcher vs. dependency graph

5 Conclusions and Future Work

This paper introduced a novel soft computing prefetcher for web caches tailored to mitigate the degradation of web system caches by web robots. The approach is rooted in the idea that a deep recurrent neural network would be able to find patterns in the ordering of subdirectories visited by web robots, and that Bayesian models can incorporate observations about all robot traffic with data about a particular robot to formulate accurate request predictions within subdirectories. Evaluation results indicate significant gains over MLE-based approaches for predicting web robot request patterns, and significantly better prefetching performance for a web cache compared to the popular dependency graph approach. Future work will further evaluate the dual caching approach and will be exercised over datasets from different universities. Alternative Bayesian models, incorporating patterns besides request type patterns, will also be explored for robot resource request prediction.

Acknowledgment

The authors thank Logan Rickert for data processing support, Maria-Carla Calzarossa for data from the University of Pavia, and Mark Anderson for data from Wright State University. This paper is based on work supported by the National Science Foundation (NSF) under Grant No. 1464104. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

References

1. Almeida, V., Menascé, D., Riedi, R., Peligrinelli, F., Fonseca, R., Meira Jr, W.: Analyzing web robots and their impact on caching. In: Proc. Sixth Workshop on Web Caching and Content Distribution. pp. 20–22 (2001)
2. Brandman, O., Cho, J., Garcia-Molina, H., Shivakumar, S.: Crawler-friendly web servers. Proc. of Performance and Architecture of Web Servers Conference (2000)
3. Chen, X., Zhang, X.: A popularity-based prediction model for web prefetching. Computer 36(3), 63–70 (2003)
4. Dietz, L.: Directed factor graph notation for generative models. Max Planck Institute for Informatics, Tech. Rep (2010)
5. Doran, D., Gokhale, S.: A classification framework for web robots. Journal of the American Society of Information Science and Technology 63, 2549–2554 (2012)
6. Doran, D., Gokhale, S.S.: Web robot detection techniques: overview and limitations. Data Mining and Knowledge Discovery 22(1-2), 183–210 (2011)
7. Doran, D., Morillo, K., Gokhale, S.: A Comparison of Web Robot and Human Requests. In: Proc. of ACM/IEEE Conference on Advances in Social Network Analysis and Mining. pp. 1374–1380 (2013)
8. Gellert, A., Florea, A.: Web prefetching through efficient prediction by partial matching. World Wide Web pp. 1–12 (2015)
9. Graves, A.: Neural networks. In: Supervised Sequence Labelling with Recurrent Neural Networks, pp. 15–35. Springer (2012)
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation 9(8), 1735–1780 (1997)
11. Lee, J., Cha, S., Lee, D., Lee, H.: Classification of web robots: An empirical study based on over one billion requests. computers & security 28(8), 795–802 (2009)
12. Li, H., Lee, W.C., Sivasubramaniam, A., Giles, C.L.: A hybrid cache and prefetch mechanism for scientific literature search engines. In: International Conference on Web Engineering. pp. 121–136 (2007)
13. Menascé, D., Almeida, V., Riedi, R., Ribeiro, F., Fonseca, R., Meira Jr, W.: In search of invariants for e-business workloads. In: Proceedings of the 2nd ACM conference on Electronic commerce. pp. 56–65 (2000)
14. Pallis, G., Vakali, A., Pokorny, J.: A clustering-based prefetching scheme on a web cache environment. Computers & Electrical Engineering 34(4), 309–323 (2008)
15. Qualman, E.: Socialnomics: How social media transforms the way we live and do business. John Wiley & Sons (2012)
16. Rude, H.N., Doran, D.: Request type prediction for web robot and internet of things traffic. In: Proc. of. IEEE Intl. Conference on Machine Learning and Applications. pp. 995–1000 (2015)
17. Zeifman, I.: Report: Bot traffic is up to 61.5% of all website traffic, bit.ly/MoMRxE