A taxonomy of web prediction algorithms

Josep Domenech, Bernardo de la Ossa, Julio Sahuquillo,
Jose A. Gil and Ana Pont
Universitat Politecnica de Valencia.
Camí de Vera, s/n. 46022 Valencia (Spain)
jdomenech@upvnet.upv.es; berospe@doctor.upv.es;
{jsahuqui,jagil,apont}@disca.upv.es

November 10, 2011

Abstract

Web prefetching techniques are an attractive solution to reduce the user-perceived latency. These techniques are driven by a prediction engine or algorithm that guesses following actions of web users. A large amount of prediction algorithms has been proposed since the first prefetching approach was published, although it is only over the last two or three years when they have begun to be successfully implemented in commercial products. These algorithms can be implemented in any element of the web architecture and can use a wide variety of information as input. This affects their structure, data system, computational resources and accuracy. The knowledge of the input information and the understanding of how it can be handled to make predictions can help to improve the design of current prediction engines, and consequently prefetching techniques.

This paper analyzes fifty of the most relevant algorithms proposed along 15 years of prefetching research and proposes a taxonomy where the algorithms are classified according to the input data they use. For each group, the main advantages and shortcomings are highlighted.

1 Introduction

Web prediction algorithms pursue to discover patterns that permit them to foresee subsequent web user demands. Applications of these predictions include recommendation systems [11], e-commerce [1], content personalization [34], and reduction of user-perceived latencies by means of web prefetching [36] and cache prevalidation [17]. Since the goals of these applications are completely different, prediction algorithms have different requirements when used for each of them. For instance, prediction algorithms for web prefetching might work at object-level granularity because prefetching one object might reduce the user-perceived latency. However, recommendation systems focus on page-level granularity since users expect that the system recommends a page, not a single object composing a page.

This paper focuses on prediction algorithms applied to web prefetching. Web prefetching is a technique aimed at reducing user-perceived latencies by processing (usually downloading) user requests before they actually demand them. For example, if a user is likely to visit page B after visiting page A, the browser can download and cache page B just after requesting page A, i.e., while the user is reading page A. In this case, when the user actually requests page B, this is displayed without network latency since it is already in cache. Accurate predictions are required because, on misprediction, prefetched objects waste client, server and/or network resources, and this could degrade the overall system performance. For this reason, the prediction algorithm is the core of web prefetching.

A wide range of prediction algorithms for web prefetching have been proposed in the literature over the last 15 years. Although early proposals focused on exploiting the object popularity to determine the objects that are more likely to be requested in the near future, the most explored option is to predict subsequent requests from past sequences of requests. However, over the last years, other data sources beyond the sequence of accesses, such as the site link structure, the HTTP headers or the page contents, are being included in the prediction algorithms.

Understanding and identifying advantages and shortcomings of prediction algorithms is a hard task even for an experienced researcher in this field because of the large amount of proposals that work according to different rules. This means that selecting the best approach or developing a new one for a given environment represents an arduous task.

Few attempts classifying prediction algorithms have been published and they mainly focus on how sequences of accesses are analyzed. Rabinovich and Spatscheck [41] classify different approaches depending on how predictions are made. They differentiate three categories of Markov models depending on how many previous accesses are used. Other category includes algorithms based on the popularity that web objects exhibit, and another one is dedicated to those algorithms exploiting the structure of the web. The last classification published, presented by Davison [15], divides the prediction algorithms into five categories according to how the sequence of accesses is analyzed. Although it is mentioned that more objects can be predicted if extra information sources are used (e.g., web page content), this is not analyzed in depth.

Unfortunately, the complexity of prediction algorithms published in the literature from Davison's classification has noticeably increased. As a consequence, many of these proposals do not fit well in any of the previously identified categories. Therefore, a new effort is required to order and clarify them to provide a sound understanding of their advantages, progresses and real possibilities of use.

In this paper, we propose a prediction algorithm taxonomy in which the algorithms are classified by the type of information they use to make predictions. The algorithms are organized in four main categories depending on whether they take as input the object characteristics, the sequence of requests, the HTTP headers and/or the content of web pages. This classification is proposed because the input of the predictors determine what kind of accesses can be predicted and how complex the analysis is. For instance, a recently added link can be predicted by an algorithm using as input the sequence of requests only if at least one user has already followed that link.

The main contribution of this paper is to present a comprehensive review of web prediction algorithms grouped by a novel taxonomy, thus easing the understanding of how predictors work and the implications of selecting a given prediction algorithm when designing a new prefetching system. In other words, this paper is aimed at examining the pros and cons of each group of algorithms and identifying current and future trends in web prediction engines.

The remainder of the paper is organized as follows. Section 2 describes the elements that compose the web prefetching architecture, highlighting the function of the prediction algorithm. Section 3 classifies web prediction algorithms according to the proposed taxonomy. Finally, Section 4 draws some concluding remarks.

2 Web prefetching architecture

Web prefetching systems are implemented by extending the generic web architecture (with clients, servers and proxies) with two extra elements: the prediction and the prefetching engines. The function of the prediction engine is to guess future client actions, e.g., requests. Its implementation corresponds to a prediction algorithm whose output is a hint list. In most proposals, this hint list includes references to the predicted objects, although some few approaches suggest other type of actions [12]. These predictions (or hints) are usually made basing on previous experience about user accesses and preferences. These hints are provided to a prefetching engine, which is aimed at preprocessing (e.g., downloading) those objects predicted by the prediction engine. By preprocessing the requests in advance, the user-perceived latency is reduced when these objects are requested by users.

The amount and scope of the information that can be used by the prediction engine depends on the element of the Web architecture (the client, the proxy or the server) at which the prediction engine is located. When the prediction engine is located at the client, the algorithm can only use access patterns, contents seen and preferences coming from one single user. Consequently, those predictions are only useful for this user. If the prediction engine is located at a proxy server, it can take advantage of the multi-user and multi-server information gathered at this element to perform the predictions. When the predictor is located at the server side (origin server or replica in the context of Content Distribution Networks (CDN)), it makes predictions based on multi-user accesses and preferences to the same web site. This option has been the most explored in the research literature because of the accuracy of the predictions made and its potential use in real scenarios. Finally, in more complex proposals, the predictions can be performed by several elements in collaboration. This benefits the accuracy and coverage of the predictions.

The prefetching engine is independent from the predictor and can be located in any element of the web architecture that receives the hint list. However, the common trend is to locate it at the client side as it is currently implemented in Mozilla-based browsers. The prefetched objects are stored in a cache until they are demanded or evicted.

Therefore, only those objects that can be stored at the web client must be predicted or prefetched. This is not a drawback because, although a web page can be the result of a dynamic response, many of the objects composing that page are usually static and consequently cacheable. Furthermore, a dynamically generated response can be cached if its headers properly mark the response. As all the cacheable objects can be prefetched (precached), prefetching techniques

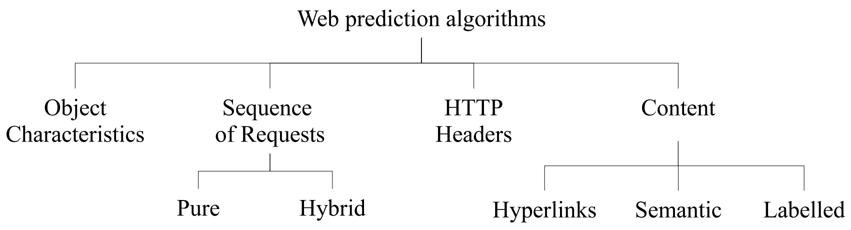

Figure 1: Web prediction algorithm taxonomy

cover, among other technologies, dynamic web server and application programming, and browser application programming like AJAX, Java, Flash, and other Rich Internet Applications.

The performance of prediction algorithms is usually measured by means of metrics that quantify both the efficiency and the efficacy of the proposal. Although some proposals use some specific indexes, the general trend when evaluating prediction algorithms is to trade-off *Recall* and *Precision*, as main performance metrics. *Recall* is defined as the ratio of requested objects that were previously predicted. As the recall quantifies the weight of the predicted objects over the amount of objects requested by the user, it can give an idea about the usefulness or coverage of the predictions. *Precision* is defined as the ratio of good predictions to the number of predictions, so it is related to the quantity of wasted resources.

3 Prediction engine taxonomy

This section presents the proposed taxonomy, which breaks down the prediction algorithms into four categories according to the information used to make the predictions.

Earlier proposals simply consider isolated object popularity, while some others analyze the sequence of user requests to discover user navigation patterns. To discover the structure of web pages and the relation between requests, more recent algorithms take into account the HTTP headers or examine page contents. Content-based algorithms may involve the analysis of the list of hyperlinks, the text surrounding hyperlinks or the labels with meta-information included by the web designer.

This taxonomy, represented in Figure 1, summarizes the prediction algorithms ranging from the earliest works in 1995 to the most recent ones found in 2010. The algorithms were designed considering the characteristics of their contemporary web sites and internet technologies. In this sense, this survey represents a review that covers more than 15 years of research following the evolution of the Web, from the genesis when static pages had few embedded objects and hyperlinks to the current web, with dynamic and personalized pages.

3.1 Object characteristics

Most of the algorithms falling in this category were proposed early in the Web. In those years, more than one decade ago, the Web structure was much simpler than nowadays. As a consequence, and in the absence of other approaches, the most intuitive and simplest algorithms were proposed.

These proposals focus on analyzing object characteristics, like temporal locality and frequency of accesses (i.e., popularity), which were the first aspects studied in the Web. Algorithms falling in this category usually follow the *long-term* approach, where clients are *subscribed* to proxies storing specific objects, which are in charge of keeping updated copies of the objects. The opposite is referred to as the *short-term* approach, which aims to prefetch those objects that are likely to be referenced in the near future, given user's recent accesses.

The work by Bestavros et al. [5] provides an overview of the performance benefits of propagating information from servers to proxies that are closer to their clients. No algorithm is described but only some ideas on which the algorithm should be based. The suggested algorithm takes as input the rate of transferred bytes among clients and proxies and the cumulative distribution of accesses to the objects as well. This idea was deeply worked in other work by the same author [6], which explores the performance gains of propagating information from servers to proxies. That is, the information is replicated at proxies which are closer to the final users. The level of propagation is based on the popularity of the objects and on the expected reduction in traffic. This approach exploits the properties of temporal and geographical locality of references, which, unlike the previous proposals, require a large community (e.g., thousands) of users.

The Top-10 approach proposed by Markatos and Chronaki [33], also published in the last century, is one of the first attempts to exploit the web documents' popularity for prefetching purposes. Additional information obtained from the client side is also used to adapt prefetching to different client profiles. In this approach, the web server publishes a list of the most popular objects and this information is complemented with client access preferences to predict future user accesses. This approach has been included in this category since the main idea of the proposal is the use of the object popularity to predict future accesses, although client profile information is also used to control and adapt prefetching.

In [26], Jian et al. propose an algorithm based on prefetching objects exhibiting a given characteristic. In this work, three main characteristics are explored and quantified: object lifetime, probability of being accessed, and user request arrival rate. The algorithm prefetches those objects where the estimated values for a given characteristic exceed an specific threshold. The proposal works among clients and, either proxies or servers. The server collects information characterizing the objects and makes it available to the prefetching engine as well.

Wu et al. [47] propose an algorithm aimed at improving a given performance metric, e.g., hit ratio, bandwidth consumption, or a ratio of both (H/B). Depending on the target metric, the algorithm prefetches objects showing some particular characteristic (e.g., popularity, longer update intervals or a mix of both). The best performance is achieved by the algorithm which maximizes the H/B metric. The approach assumes that web servers push fresh copies into web caches (assuming unlimited cache size) or proxy servers whenever such copies are updated.

In [46], Venkataramani et al. propose a long-term approach where proxy caches are subscribed to those popular objects that are updated with low frequency. This approach is aimed at being used in the last level caches of a CDN hierarchical structure while the typical prefetching (i.e., short-term approach) should remain in the lower level caches. In [30], Lau and Ng propose a prefetching algorithm that analyzes the browser's cache according to the long-term approach. The algorithm predicts and prefetches Web documents and their linked documents as well. Two main characteristics are analyzed for each object: its access rate and the last time it was accessed. As in [26], those objects whose characteristic exceeds a given threshold are prefetched. The algorithm uses the detection theory to determine the threshold value. In addition, users can take an active part in the prefetching engine by using a graphic user interface where they can modify any computed prefetching schedule.

The work by Rangarajan et al. [42] can be characterized as a short-term approach. This paper presents a technique that dynamically groups users based on their frequency of access to each object. In this way, the server structure can be reorganized according to the needs of each group of users. To avoid overloading the network, the proposal predicts in a cluster-based fashion instead of predicting for each single user; that is, whenever a host connects to the server or a proxy, the prefetching strategy returns the URIs predicted for the cluster to which the host belongs.

Unlike the previous proposals, in the work by Duchamp [21], servers manage not only statistics about the objects they store but also about objects in other servers. All this information is then summarized and a probability of being used is estimated. The prediction algorithm takes information both from the server and client sides. Once the prediction is received, the client side can interact to decide whether or not to prefetch the hinted hyperlinks according to some local parameters.

Algorithms in this category made sense when they were proposed (i.e., early in the Web), since the Web was simple and static regarding the object's point of view as well as its lifetime. Nowadays, this approach fits better to CDNs, which control both the origin and replica contents.

3.2 Sequence of requests

Most of the web prediction algorithms proposed in the literature lie in the analysis of the paths followed by each client, that is, in the sequence of requests. By comparing the user's recent accesses to the paths previously recorded, algorithms are able to predict future accesses.

Algorithms falling into this category require to distinguish which client is issuing the request in order to keep track of the right sequence of accesses. Although solving this issue is trivial when the prediction engine is implemented in the web browser, it becomes more complex when it is implemented in the proxy (keeping track of the clients IP could be enough) or in the server (frequently requiring the use of cookies).

The element of the web architecture (i.e., client, proxy or server) in which the prediction engine is implemented is also a performance limiting factor for prediction engines of this type [19, 29]. This is because predictors falling in this category cannot predict accesses to those resources that have not been requested before to the element where the predictor is placed (first-seen resources). As expected, the ratio of first-seen resources is low when predictors are located at the server but high when located at the client [19]. However, this limitation can be overcome if additional sources of information are provided to the prediction engine. Therefore, algorithms in this category have been divided into *pure* and *hybrid*, depending on whether the predictor considers only the sequence of requests or also takes into account additional information to refine predictions.

3.2.1 Pure predictors

There is a large number of proposals specifically addressed to predict subsequent accesses from the sequence of requests that a web server receives. Some variations of Markov models are often used to predict and hint resources that other element of the web architecture (i.e., the proxy or the clients) will prefetch [36, 13, 4, 38]. The dependency graph (DG) proposed by Padmanabhan and Mogul [36] is usually taken as baseline for performance comparisons. The graph has a node for every object that has ever been accessed. There is an arc from node A to B if and only if at some point in time a client accessed B within w accesses after A, where w is the *lookahead window* size. The weight of the arc is the ratio of the number of accesses to B within a window after A to the number of accesses to A itself.

Some other algorithms [43, 31] use the predictions to preprocess resources at the server side. Schechter et al. [43] use session paths to make predictions for generating some dynamic content in advance. In the context of a distributed web server cluster, the proposal of Lee et al. [31] combines a prediction algorithm with a workload distribution algorithm to prefetch objects from disk in each web server.

Kim et al. [28] propose a predictor that takes into account the sequence of requests from the client perspective. This sequence is used to predict the most likely link in the current page that the user will follow. Analyzing the sequence of requests at the client side has as main advantage that no accesses are hidden due to the effects of caching. However, since it considers single user information, it can only predict previously followed links, but not accesses to new content.

Focusing on the proxy perspective, Fan et al. [22] employ the well-known prediction by partial matching (PPM) algorithm in such a way that it aggregates all users data to make predictions for each single user in a client-proxy context. The joint benefits of caching and prefetching are also analyzed by Teng et al. [45] in a proxy-server context. Its prediction algorithm is based on rules, each one with a confidence level, in which the next requested object depends on the sequence of objects recently requested. In contrast, in the proposal of Huang et al. [25], the prediction engine tries to make a prediction from the same user's previous accesses. Only if it is not possible, the algorithm uses the aggregate information of other users.

Since the range of accesses that are recorded is high, an important issue when the prediction engine is located at the proxy is to reduce the resource consumption. To deal with this concern, Yang et al. [49] propose a data structure to record web sessions in which it is possible to apply any data mining algorithm. The proposed algorithm works in two steps: first, user sessions are clustered; second, transition probabilities between pages in the same cluster are computed.

In a similar way, the algorithm proposed by Pallis et al. [37] identifies clusters of pages whose accesses are correlated as its first step to make predictions.

Other approaches take into account the sequence of requests from several points of view to complement the sequences perceived in one element. In this context, requests that hit in a proxy cache are hidden for the origin server. This fact distorts the sequence of requests at the server and prevents the server from making a prediction for that request. Chen et al. [9] deal with this problem by proposing an algorithm that coordinates servers and proxies to make predictions. Another coordinated approach is taken by Pons [39] in the context of a web application, but considering coordination between proxies and clients. In this proposal, the proxy builds a generic prediction model that each client receives and adapts according to its own patterns.

Some other prediction algorithms use the sequence of accesses as the only input, but they are not proposed to work in a specific element of the architecture. To improve PPM prediction accuracy, Ban et al. [3] propose to include extra data in the model to describe a node prediction capability. Chen and Zhang [8] propose a memory-efficient version of a PPM algorithm whose prediction tree branches have different height depending on the root object popularity. A similar algorithm, using Markov models of different orders, is proposed by Dongshan and Junyi [20] to deal with the scalability of the prediction engine. The model presented by Gunduz and Ozsu [24] compare session similarities to make predictions. Sessions are modeled taking into account the sequence of user requests and the time spent in each page.

The work by Bonino et al. [7] describes an initial set of predictors (represented as finite-state machines) that evolve according to genetic operators. One of these predictors is selected every time a request is received in order to make a prediction. The graph representing user sessions is different from those usually employed in Markov models, since requests are represented by edges, and nodes represent predictions instead of objects.

3.2.2 Hybrid predictors

Predictors falling into this subcategory are those that complement the sequence of requests with another source of information, e.g., the HTTP headers or page contents. The main reasons to take into account extra sources are usually either to increase the prediction accuracy or to control the computational complexity of the algorithms.

Three of the prediction algorithms analyzed by Zukerman et al. [50] are based on Markov models that employ the sequence of requests at the server as the main source. One of them, the Linked Space-Time Markov model, combines the sequence of accesses with the referrer information found in the HTTP headers. In a later work [2], they combine all the models to build a single predictor.

The Double Dependency Graph (DDG) prediction algorithm (see Section 3.3) proposed by Domenech et al. [18] employs the sequence of user accesses to build a request dependency graph similar to the DG algorithm proposed in [36]. This predictor also requires to analyze response headers to distinguish container from embedded objects.

The prediction engine proposed by Georgakis and Li in [23] (described in Section 3.4) also takes as input the sequence of requests received by the server.

However, it is more interested in the sequence of content read than in the path followed by the user.

Nanopoulos et al. [35] propose an algorithm that builds association rules from the sequence of accesses, allowing only those page transitions found in the link structure of the web site. In a similar way, the proposal by Mabroukeh and Ezeife [32] builds a Markov model to make the predictions. However, in this case, the prediction tree is pruned by using the concept of *semantic distance*. To compute this distance, a domain ontology, which is provided during the design of the web site, is required.

3.3 HTTP headers

This category includes those prediction algorithms that use the HTTP headers either as the only input or in combination with other inputs.

Web client requests are issued by sending HTTP requests, which contain several headers providing information about request details, besides the URI. One of the most used request headers is the "Referer". Although this request header is optional in the standard protocol, the most popular web browsers always provide it in their requests. A prediction algorithm can discover first-order dependences by only observing (Requested Object URI, Referrer URI) pairs.

Together with the requested object, the web server returns some HTTP response headers. By looking at the response headers, a prediction algorithm can take some features of the requested objects as input. Response headers used in prediction algorithms include "Content-Type" and "Content-Length" headers.

Zukerman et al. [50] compare several Markov-based prediction models that take into account different factors: the sequence of documents requests, the web site structure inferred by using the "Referer" request header, and a combination of both. Their results show that the model that combines all these factors in the Markov model yields the best predictive power, in terms of time and space consumption. The same authors propose, in a later work [2], the max-Hybrid prediction algorithm, which uses as input the sequence of accesses with timestamp, the "Referer" request header, and the "Content-Length" response header.

The DDG algorithm [18], classified and introduced in the previous section, employs the "Content-Type" response header to distinguish between container and embedded objects. Container objects are those that the user demand explicitly, while secondary objects are those embedded in the page and requested automatically by the web browser. This allows DDG to differentiate the dependences of objects of the same page and objects of different pages. Thus, DDG represents the structure of the web site more faithfully than DG.

The Referrer Graph (RG) prediction algorithm proposed by Ossa et al. [16] is based on DDG. Both algorithms use the "Content-Type" response header to classify objects as primary (container) and secondary (embedded) objects. However, unlike DDG, which uses the temporal sequence of accesses, RG uses the URI and the "Referrer" request header provided in each individual object request to build the arcs in the graph. In this way, RG builds arcs only between objects that are really linked with hyper references in the web site, and not between objects that have been requested sequentially by a web browser. This allows RG to represent the structure of a web site more faithfully than DDG.

HTTP headers can be considered accurate since they are provided explicitly by the web client and server. In other words, there is no derived guessing that might be erroneously interpreted as happens in the sequence of requets category.

As it is unnecessary to access the full object content, these algorithms allow shorter processing time and are less intrusive than those in the content category. The headers are provided continually, and this allows the algorithms to actively capture user accesses.

3.4 Content

Prediction algorithms that use the information extracted from the content of the web pages as main input are included in this category. These prediction engines use the current web page to make predictions of future accesses, usually without requiring history information. The algorithms in this category are specially designed to deal with objects that are newly created or never visited before, and with dynamic sites where a URL could have time-dependent contents.

Despite the wide variety of proposals falling into this category, we have identified three trends when using page content as main input information.

3.4.1 Using hyperlinks

The first approach proposed only used the references to objects found when parsing the current web page, without any further analysis. Chinen and Yamaguchi [10] designed and implemented a prefetching proxy server called the WWW Collector (Wcol) by using this idea. This proxy prefetches and stores referenced pages, and also shares prefetched objects with its other clients.

Cohen and Kaplan describe in [12] three simple prefetching techniques (pre-resolving host-names, pre-connecting and pre-warming) that can be used with non-prefetchable URLs. Predictions are performed by extracting URLs from anchor links of previously visited pages. The study considers two classes of requests. The first class includes pages linked in the results page of a search engine or web portal, whereas the second class contains all requests that were not preceded by a request to the server in the last 60 seconds.

More recently, in the context of CDNs, Sidiropoulos et al. [44] identify clusters of “correlated” web pages in a site (web site communities), and make these communities the basic outsourcing unit of content to be prefetched. The approach is based only on the hyperlink information, since they assume that two pages are “similar” or “correlated” if there exists a link between them.

The algorithm by Nanopoulos et al. [35], described in Section 3.2, also falls into this category because it complements the prediction model obtained with the sequence of requests with the hyperlink graph of the web site. This graph is applied to the prediction model to allow only those page transitions that are included in it.

3.4.2 Semantic approaches

Algorithms in this subcategory tend to capture the user surfing interest from semantic characteristics of visited pages.

A keyword-based semantic prefetching approach is proposed by Xu and Ibrahim [48]. This algorithm lies in the fact that client surfing is often guided by some keywords in the text that surrounds hyperlink definitions (hrefs) in web pages. Therefore, this approach predicts future requests based on semantic preferences of past retrieved objects. To do so, the algorithm ranks the links of the last visited page according to the similarity of the keywords found in their "anchor text" to those in the previously accessed links.

More recently, Georgakis and Li [23] propose an algorithm that also considers the anchored text around each of the outbound hyperlinks. To assign a weight to each of these links, the algorithm keeps counters of the frequencies of appearance of bigrams for visited and non-visited links.

Davison [14] proposes an algorithm that predicts the user's next web page request by conducting an analysis of the text of his/her recently requested web pages. The algorithm compares the text in and around the hypertext anchors of the current page to the text contained in previously visited pages.

3.4.3 Labeled approaches

This subcategory includes all those proposals where the web designer is involved in the prediction process by including extra information that will be later used to predict future accesses. The common advantage of these specific and designer-dependent proposals is that they exhibit interesting results for the environments where they are tested, that is, those for which the proposals have been developed. This is also their main disadvantage, because they are not generic solutions that could be easily adapted to general and real scenarios.

Khan and Tao identify in [27] four dominant patterns in hyperlink structure (Chain, Tree, Complete Graph, and Tree with Complete Core). Their proposal consists on labeling links with this kind of information to develop smarter prefetching techniques when the structure of webspace is also brought into consideration.

Pons [40] proposes to conduct web prefetching by using semantic links. That is, semantic information explicitly embedded during the web design and associated with each web page link. A semantic link is different from a hyperlink, since the semantic link represents a pointer with a type or meaning directed from the predecessor web page to the successor web page. Examples of semantic link types are sequential, similar-to, cause-effective, implication, etc.

The proposal of Mabroukeh [32] has been previously classified into the group of algorithms using the sequence of request as input because this is the main information used for prediction. Nevertheless, it is actually a hybrid algorithm that also employs semantic information to prune the Markov model according to a computed semantic distance. This proposal assumes that the semantic information is available in the form of domain ontology, provided during the design of the web site.

4 Conclusions

Since the inception of web prefetching, the proposals of prediction algorithms have adapted to the increasing complexity of the web content. Thus, the complexity of the predictors has increased, too. In this paper, we have summarized

Table 1: Surveyed papers by category and year

CATEGORY		**1995-2000**	**2001-2005**	**2006-2010**
Obj. characteristics		[6, 5, 33, 21]	[26, 46, 30, 42]	[47]
Seq. requests	*pure*	[4, 36, 43, 13, 22, 38]	[8, 20, 7, 24, 28, 49, 9, 25, 39, 45]	[31, 3, 37]
	hybrid	[2, 50]	[35]	[23, 32, 18]
HTTP headers		[2, 50]		[16, 18]
Content		[10]	[12, 14, 27, 35, 48]	[23, 40, 44, 32]

and categorized more than 15 years of research in prediction algorithms applied to web prefetching techniques. To this end, a new taxonomy based on the input information used by the prediction algorithms has been proposed and successfully applied. Through the proposed taxonomy, the pros and cons of each category of algorithms have been discussed.

Table 1 summarizes the classification of the surveyed papers according to the taxonomy proposed in this work and the year of publication. This table permits us to identify trends in algorithm design. As one can observe, during the first period of five years, most of the proposed algorithms dealt only with object characteristics or sequence of requests to make predictions. In the next five-year period, the predominant choice was to use only the sequence of accesses as input of the prediction algorithms, although the number of content-based proposals also increased. Finally, during the last five years, researchers have moved towards more sophisticated inputs. Algorithms considering several input types as well as predictors that analyze web page contents or require the collaboration of the web page designer are gaining relative importance in the current trend. Furthermore, it has also been observed the increasing concern for the computational costs of the algorithms.

Despite of this, there are several gaps in the design space of prediction algorithms that remain unexplored. Few papers, i.e., [21, 9, 39], analyze several web architecture points of view simultaneously, and none of them combine different input categories. Besides, the use of HTTP headers has not been intensively explored and no algorithm combining header and content analysis has been proposed.

As for future work, we plan to classify other aspects of web prefetching systems: proposals of prefetching engines according to its location in the web architecture; and proposals of web prefetching architectures according to how prediction and prefetching engines are coordinated, thus providing a complete study about web prefetching techniques from its beginnings to current days.

Acknowledgments

This work has been partially supported by Spanish Ministry of Science and Innovation under grant TIN2009-08201, Generalitat Valenciana under grant GV/2011/002 and Universitat Politecnica de Valencia under grant PAID-06-

10/2424.

References

[1] Silvana Vanesa Aciar, Christian Serarols-Tarres, Marcelo Royo-Vela, and Josep Lluis de la Rosa i Esteva. Increasing effectiveness in e-commerce: recommendations applying intelligent agents. *International Journal of Business and Systems Research*, 1(1):81–97, 2007.

[2] David W. Albrecht, Ingrid Zukerman, and Ann E. Nicholson. Pre-sending documents on the WWW: A comparative study. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, 1999.

[3] Zhijie Ban, Zhimin Gu, and Yu Jin. A ppm prediction model based on stochastic gradient descent for web prefetching. In *Proceedings of the IEEE 22nd International Conference on Advanced Information Networking and Applications (AINA)*, pages 166–173, Okinawa, Japan, 2008.

[4] Azer Bestavros. Using speculation to reduce server load and service time on the WWW. In *Proceedings of the 4th ACM International Conference on Information and Knowledge Management*, Baltimore, USA, 1995.

[5] Azer Bestavros. Speculative data dissemination and service to reduce server load, network traffic and service time in distributed information systems. In *Proceedings of the Twelfth International Conference on Data Engineering*, pages 180–187, New Orleans, USA, 1996.

[6] Azer Bestavros and C. Cunha. Server-initiated document dissemination for the WWW. *IEEE Data Engineering Bulletin*, 1996.

[7] Dario Bonino, Fulvio Corno, and Giovanni Squillero. A real-time evolutionary algorithm for web prediction. In *Proceedings of the IEEE/WIC International Conference on Web Intelligence*, Halifax, Canada, 2003.

[8] Xin Chen and Xiaodong Zhang. Popularity-based PPM: An effective web prefetching technique for high accuracy and low storage. In *Proceedings of the International Conference on Parallel Processing*, Vancouver, Canada, 2002.

[9] Xin Chen and Xiaodong Zhang. Coordinated data prefetching for web contents. *Computer Communications*, 28:1947–1958, 2005.

[10] K. Chinen and S. Yamaguchi. An interactive prefetching proxy server for improvement of www latency. *Proceedings of Seventh Annual Conference of the Internet Society (INET'97)*, 06/1997 1997.

[11] Wei Chu and Seung-Taek Park. Personalized recommendation on dynamic content using predictive bilinear models. In *Proceedings of the 18th International Conference on World Wide Web*, pages 691–700, Madrid, Spain, 2009.

[12] Edith Cohen and Haim Kaplan. Prefetching the means for document transfer: a new approach for reducing web latency. *Computer Networks*, 39(4):437–455, 2002.

[13] Edith Cohen, Balachander Krishnamurthy, and Jennifer Rexford. Efficient algorithms for predicting requests to web servers. In *Proceedings of the IEEE INFOCOM '99 Conference*, New York, USA, 1999.

[14] Brian D. Davison. Predicting web actions from HTML content. In *Proceedings of the 13th ACM Conference on Hypertext and Hypermedia*, College Park, USA, 2002.

[15] Brian D. Davison. Learning web request patterns. In *Web Dynamics - Adapting to Change in Content, Size, Topology and Use*, pages 435–460. Springer, 2004.

[16] Bernardo de la Ossa, Ana Pont, Julio Sahuquillo, and J. A. Gil. Referrer graph: a low-cost web prediction algorithm. ACM, 2010.

[17] Josep Domenech, Jose A. Gil, Julio Sahuquillo, and Ana Pont. Speculative validation of web objects for further reducing the user-perceived latency. In *Proceedings of the 9th International IFIP TC 6 Networking Conference*, Chennai, India, 2010.

[18] Josep Domenech, Jose A. Gil, Julio Sahuquillo, and Ana Pont. Using current web page structure to improve prefetching performance. *Computer Networks*, 54(9):1404 – 1417, 2010.

[19] Josep Domènech, Julio Sahuquillo, José A. Gil, and Ana Pont. The impact of the web prefetching architecture on the limits of reducing user's perceived latency. In *Proceedings of the 2006 IEEE / WIC / ACM International Conference on Web Intelligence*, Hong Kong, China, 2006.

[20] Xing Dongshan and Shen Junyi. A new Markov model for web access prediction. *Computing in Science and Engineering*, 4(6):34–39, 2002.

[21] Dan Duchamp. Prefetching hyperlinks. In *Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems*, Boulder, USA, 1999.

[22] Li Fan, Pei Cao, Wei Lin, and Quinn Jacobson. Web prefetching between low-bandwidth clients and proxies: Potential and performance. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling Of Computer Systems*, pages 178–187, Atlanta, USA, 1999.

[23] A Georgakis and H Li. User behavior modeling and content based speculative web page prefetching. *Data & Knowledge Engineering*, 59:770 – 788, 12/2006 2006.

[24] Sule Gunduz and M. Tamer Zsu. A web page prediction model based on click-stream tree representation of user behavior. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–540, New York, USA, 2003. ACM Press.

[25] Yin-Fu Huang and Jhao-Min Hsu. Mining web logs to improve hit ratios of prefetching and caching. In *Proceedings of the 2005 IEEE / WIC / ACM International Conference on Web Intelligence*, Compiegne, France, 2005.

[26] Yingyin Jiang, Min-You Wu, and Wei Shu. Web prefetching: Costs, benefits and performance. In *Proceedings of the 7th International Workshop on Web Content Caching and Content Distribution*, Boulder, USA, 2002.

[27] Javed I. Khan and Qingping Tao. Exploiting webspace organization for accelerating web prefetching. In *Proceedings of the IEEE/WIC International Conference on Web Intelligence*, Halifax, Canada, 2003.

[28] Yuna Kim and Jong Kim. Web prefetching using display-based prediction. In *Proceedings of the IEEE/WIC International Conference on Web Intelligence*, Halifax, Canada, 2003.

[29] Thomas M. Kroeger, Darrell D.E. Long, and Jeffrey C. Mogul. Exploring the bounds of web latency reduction from caching and prefetching. In *Proceedings of the 1st USENIX Symposium on Internet Technologies and Systems*, Monterey, USA, 1997.

[30] Kelvin Lau and Yiu-Kai Ng. A client-based web prefetching management system based on detection theory. In *Proceedings of the Web Content Caching and Distribution: 9th International Workshop (WCW 2004)*, pages 129–143, Beijing, China, 2004.

[31] Heung Ki Lee, Gopinath Vageesan, Ki Hwan Yum, and Eun Jung Kim. A proactive request distribution (prord) using web log mining in a cluster-based web server. In *Proceedings of the International Conference on Parallel Processing (ICPP'06)*, Columbus, USA, 2006.

[32] Nizar R. Mabroukeh and Christie I. Ezeife. Semantic-rich markov models for web prefetching. In *Proceedings of the IEEE International Conference on Data Mining Workshops*, pages 465 – 470, Miami, FL, USA, 2009. IEEE, IEEE.

[33] Evangelos Markatos and Catherine Chronaki. A top-10 approach to prefetching on the web. In *Proceedings of the INET' 98*, Geneva, Switzerland, 1998.

[34] Bamshad Mobasher. *The Adaptive Web*, volume LNCS 4321, chapter Data Mining for Web Personalization, pages 90–135. Springer, 2007.

[35] Alexandros Nanopoulos, Dimitrios Katsaros, and Yannis Manolopoulos. A data mining algorithm for generalized web prefetching. *IEEE Trans. Knowl. Data Eng.*, 15(5):1155–1169, 2003.

[36] Venkata N. Padmanabhan and Jeffrey C. Mogul. Using predictive prefetching to improve World Wide Web latency. *Computer Communication Review*, 26(3):22–36, 1996.

[37] G Pallis, A Vakali, and J Pokorny. A clustering-based prefetching scheme on a web cache environment. *Computers & Electrical Engineering*, 34:309 – 323, 07/2008 2008.

[38] Themistoklis Palpanas and Alberto Mendelzon. Web prefetching using partial match prediction. In *Proceedings of the 4th International Web Caching Workshop*, San Diego, USA, 1999.

[39] A Pons. Improving the performance of client web object retrieval. *Journal of Systems and Software*, 74:303 – 311, 02/2005 2005.

[40] A Pons. Semantic prefetching objects of slower web site pages. *Journal of Systems and Software*, 79:1715 – 1724, 12/2006 2006.

[41] Michael Rabinovich and Oliver Spatscheck. *Web Caching and Replication.* Addison-Wesley, 2002.

[42] S.K. Rangarajan, V.V. Phoha, K.S. Balagani, R.R. Selmic, and S.S. Iyengar. Adaptive neural network clustering of web users. *Computer*, 37:34 – 40, 04/2004 2004.

[43] Stuart Schechter, Murali Krishnan, and Michael D. Smith. Using path profiles to predict http requests. In *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia, 1998.

[44] Antonis Sidiropoulos, George Pallis, Dimitrios Katsaros, Konstantinos Stamos, Athena Vakali, and Yannis Manolopoulos. Prefetching in content distribution networks via web communities identification and outsourcing. *World Wide Web*, 11:39 – 70, 3/2008 2008.

[45] Wei-Guang Teng, Cheng-Yue Chang, and Ming-Syan Chen. Integrating web caching and web prefetching in client-side proxies. *IEEE Transactions on Parallel and Distributed Systems*, 16(5):444–455, 2005.

[46] Arun Venkataramani, Praveen Yalagandula, Ravindranath Kokku, Sadia Sharif, and Mike Dahlin. The potential costs and benefits of long-term prefetching for content distribution. *Computer Communications*, 25:367–375, 2002.

[47] Bin Wu and Ajay D. Kshemkalyani. Objective-optimal algorithms for long-term web prefetching. *IEEE Transactions on Computers*, 55(1):2–17, 2006.

[48] Cheng-Zhong Xu and T.I. Ibrahim. A keyword-based semantic prefetching approach in internet news services. *IEEE Transactions on Knowledge and Data Engineering*, 16:601 – 611, 05/2004 2004.

[49] Qiang Yang, Joshua Zhexue Huang, and Michael Ng. A data cube model for prediction-based web prefetching. *Journal of Intelligent Information Systems*, 20(1):11–30, 2003.

[50] Ingrid Zukerman, David W. Albrecht, and Ann E. Nicholson. Predicting users' requests on the WWW. In *Proceedings of the 7th International Conference on User Modeling*, Banff, Canada, 1999.