

# What do Hyperlink-Proposals and Request-Prediction have in Common?\*

Ernst-Georg Haffner<sup>1</sup>, Uwe Roth<sup>1</sup>, Andreas Heuer<sup>1</sup>,  
Thomas Engel<sup>1</sup>, Christoph Meinel<sup>1</sup>

<sup>1</sup> Institute of Telematics, Bahnhofsstr. 30-32,  
D-54292 Trier, Germany  
{Haffner, Roth, Heuer, Engel, Meinel}@ti.fhg.de

**Abstract.** This paper focuses on fundamental similarities between *proposing links for hypertexts* and *predicting user-requests*. It briefly outlines the theoretical background of both categories of problems and, as an example, explores common implementation strategies for handling them. Even though there are some important differences between link-proposals and prediction of requests, we believe that in regarding these categories as special cases of a more general and abstract mathematical model, improvements on the one side will also result in advantages for the other.

## 1 Introduction

Due to the breath-taking growth of the World Wide Web (WWW), the need for high quality hypertexts is rapidly increasing, and finding appropriate links is one of the most difficult of tasks. Ultra-modern online authoring systems<sup>1</sup> that provide possibilities to check link-consistencies and administrate link management should also propose links in order to improve the usefulness of the HTML-documents.

Another major problem of today's Internet applications - and, at first glance, an entirely different one to finding hyperlinks - is the performance of Client/Server communication: servers often take a long time to respond to a client's request. There are several strategies to overcome this problem of high user-perceived latencies; one of them is to predict future requests. This way, time-consuming calculations on the server's side can be performed even before a special request is being made. If the server is "sure" that certain documents will soon be requested, the associated data can be sent to the client in advance (or can be pre-fetched by the client) even while the user is unaware of this process.

The two problem categories discussed here do not seem to have much in common. In this paper, we mean to show that there are certain, similar, solution strategies to

---

\* In: Proc. International Conference on Advances in Information Systems, ADVIS'2000, Springer LNCS 1909, Izmir, Turkey, pp: 275-282

<sup>1</sup> For instance: *Microcosm* [12], or *Daphne* [28]. Multilingual approaches are sketched in [13].

take care of both problems. Therefore, we will first have a closer look at hyperlink-proposals (section 2). Then, we will present a prediction scenario and outline advanced strategies to foresee future user-requests on a statistical base (section 3). A comparison and an abstraction of both methodologies will be highlighted in section 4. Finally, a summary and an outlook on future events will be presented in section 5.

## 2 Hyperlink-Proposals

The theory of hyperlink-research already has a long history. Hyperlink-research has been carried out ever since the introduction of the World Wide Web-service to the Internet. Kaindl et. al. present a compact outline of the progresses made so far and draw some conclusions [19].

As its final aim, link retrieval research strives to achieve the automatic generation of hyperlinks. Several systems were built to perform link-proposals. A very promising description of Chang's *HieNet* can be found in [7]. Due to various problems in finding links for hypertexts Allan distinguishes between three major characteristics of link-types: *manual*, *automatic* and *pattern-matching* [1]. Every effort in the Hyperlink-Research can thus be categorized. We will focus on retrieving "automatic" links on a statistical base with approved methods (e.g. [27] or [25]).

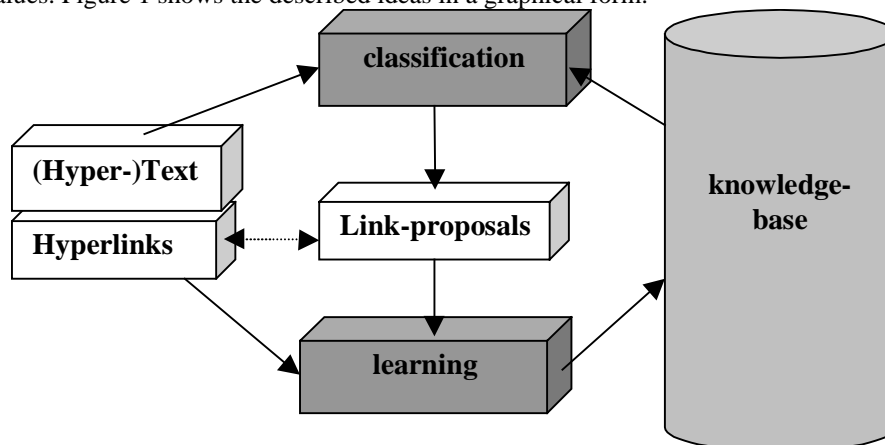
It is a very complex task to measure the quality of hyperlink-proposal algorithms. Cleary and Bareiss mention as important factors the *recall*, the share of appropriate proposals of all good links and the *precision*, the share of appropriate proposals of all proposals [6]. Often, the quality of proposals in detail is only measurable by human experts. The algorithm for the proposal of hyperlinks in this paper is based on the idea of case-based reasoning [20] with some improvements concerning efficiency and complexity.

In order to be able to propose hyperlinks for texts on a statistical base [5] without any further use of a semantic model, one has to build up a database for possible link targets first (see also [24] for another insight regarding this topic). Given a straightforward case, the targets of hyperlinks are simply the documents of the hyperlink-management system. In this paper, we mean to present a hyperlink-proposal-system where all links that are part of any hypertext can be proposed no matter whether the target document is part of the system or not. Storing the possible hyperlinks is not enough, though. Of major importance is the fact that the system must store the relationship between the text and the associated links. This step is called a *learning process*. Therefore, the database can also be called a *knowledge base*. The quality of this knowledge base depends on the quality of the learning process: how can the information of the texts be combined with appropriate hyperlinks?

One possibility would be the *human teacher*. A person or a group of persons could derive the important information of the document *manually*. This process is very cost- and time consuming and is not feasible in practice. Most of today's learning algorithms conquer the problem of high quality knowledge retrieval by extracting some information automatically on the basis of advanced heuristics. The learning process itself evaluates the relevance of the extracted information. At first, the algorithm treats a

text with hyperlinks as if it did not contain any link and derives the relevant information. Next, it proposes one or more hyperlinks and compares the result to the hyperlinks that the document in fact contains. By using this method, the learning process can be carried out without any human teacher. A disadvantage arises from the strong relationship between the quality of the learning process and the quality of the initial documents. Furthermore, only those hyperlinks can be proposed that are already known to the system and at the very beginning of the learning process there are no links to be proposed at all.

After this learning step has been taken, hyperlinks for new documents can be proposed by using the knowledge base. This phase is also called the *classification phase*. The part of retrieving the relevant information is just the same as in the learning phase, and link-proposals can be calculated. In fact, a learning component can also be found in this step. In general, the user of the hyperlink-management system accepts or rejects a proposal of the classification algorithm and thus plays the role of a human teacher. The learning algorithm can - just as in the learning phase - compare the proposal of the system with the reaction of the human user and thus adapt the relevance values. Figure 1 shows the described ideas in a graphical form.



**Figure 1.** Principle structure of a statistical-based hyperlink-proposal algorithm

A straightforward implementation of such a hyperlink-proposal algorithm could be realized by using mathematical objects such as vectors or matrixes. If the entire knowledge of the link-proposal systems were to be stored in a *relevance-matrix*  $R$ , every column would represent all the existing text attributes and every row would stand for a possible link-proposal. Furthermore, a text (without links) in the classification process could be modeled by an attribute vector  $t$  where every element corresponds to an - existing or not - textual property. The classification process would then be a multiplication of the relevance matrix with the vector and the resulting vector  $r$  would contain the probabilities for all existing links in the knowledge-base whether they are appropriate as hyperlink of the new text or not (1).

$$r = R \cdot t. \quad (1)$$

The learning process has to guarantee that a given resulting vector  $r'$ , that represents the really existing hyperlinks in the text  $t$ , would just be the result in a classification step of  $t$ . Therefore, the relevance-matrix  $R$  has to be adapted. With other words, the learning process applies a function  $f$  on  $R$  so that the following equation holds true (2):

$$r' = f(R) \cdot t. \quad (2)$$

In practice, the learning and the classification steps are combined so that the system is able to make proposals even though the knowledge base is still rather small. A description of an implementation and an evaluation of this principle can be found in [17].

In section 4, we will come back to the main ideas of this approach and we will show that these algorithms can be improved by drawing parallels for the - at first glance - quite different field of request-prediction.

### 3 Request-Prediction

The notion of predicting future events originates from compiler construction (branch prediction) and is now being applied to Internet applications [18]. Many attempts have already been made to find the best and most efficient algorithms for fulfilling prediction needs (e.g. [2]). Discrete Markov-Chain-models were used as a basis for the Web's first algorithms of prediction [21]. Their conception is to store the frequency of user-requests and to apply the adequate statistical model. Later, these ideas were extended to a continuous chain-approach [22] and to path profiling [26] which focuses on the order of document demands and the resulting request path. By using predictions, average latency and system load may be reduced but several risks may result from inaccurate data prediction. The negative effects of incorrect predictions are discussed in [9] and [4]. Performance modeling in general is described in [10].

Our former prediction-approach [14] is based on an idea of Padmanabhan and Mogul [23]. We have improved this straightforward approach to model time and document aging [15]. It is within the focus of this paper to verify that these ideas can also be applied to the hyperlink-proposal concept.

The prediction of user-requests in general aims at reducing user-perceived latency. The first concept models a group of requests as a *session*. Even though the session cannot be justified on the basis of the standard WWW protocol HTTP<sup>2</sup>, it is very important to group several requests together. In general, a session is regarded as a time period of about 30 minutes [26]. During such a session, a user can request several documents (or other kinds of data packets). The main goal of prediction aims at foreseeing some of the upcoming requests during the same session on the base of the requests that have already been made. Therefore, the prediction-algorithm generates *relative probabilities* for future document requests on the basis of the *relative frequencies* of requests in the past. Certainly, there are also some other approaches, but the perhaps most straightforward one simply stores the actual user-requests at first and then calculates probabilities for future request wishes. The former process is a (rather

---

<sup>2</sup> Hypertext Transfer Protocol

simple) kind of *learning phase*, while the latter one can be called a *classification phase*.

If we do not take care of maintaining the correct request order, a straightforward mathematical conception of a session will lead to a vector  $s$ . Every request of a document corresponds to an element of  $s$ . The size of the vector corresponds to the number of documents that should be part of the prediction algorithm (*predictable documents*). Details about finding the appropriate documents and criterions whether prediction should be made can be found in [14] and [16].

To store the relative frequencies of requests, e.g. the number of requests for every document depending on the requests of all the other documents, we need - at least - a quadratic matrix with the dimension of the number of predictable documents. We call this matrix the *Memory Matrix*  $M$ . In the case where the request order is irrelevant, the matrix  $M$  is also symmetric. For efficiency, the relative probabilities are not stored directly in  $M$ , but they are calculated from the relative frequencies at the moment when they are needed. If the function  $g$  retrieves the relative probabilities from the according frequencies, the classification of the (not completed) session  $s^n$  to the predicted final session  $s'$  can be described as (3):

$$s' = g(M) \cdot s^n. \quad (3)$$

Again, the learning process can compare the results of the classification step ( $s'$ ) to the real user behavior that leads to the (completed) session  $s$ . The function  $g$  must be adapted by using a heuristic function  $h$  so that the following equation will be true (4):

$$s = h \cdot g(M) \cdot s^n. \quad (4)$$

Additionally, the Memory Matrix  $M$  must be adapted to  $M'$  after every new request during a user session so that it still represents the relative frequencies (5).

$$M' = (M_{ij}') = \begin{cases} M_{ij} + 1 & \text{iff } s_i = 1 \wedge s_j = 1 \\ M_{ij} & \text{otherwise} \end{cases} \quad (5)$$

A very critical point in generating request-prediction is to take care of the different costs. Not only the system costs for the prediction itself must be taken into account, but also the network load and the server load for wrongly predicted documents. These considerations are very important especially for making pre-fetches. In this paper, cost-aspects are not part of the focus. Detailed information about this topic can be found in [3] and [8].

#### 4 Abstraction and Generalization

Closer analysis of the categories *finding hyperlink-proposals* and *predicting user-requests* leads to astonishingly similar results. The core function of both solution-algorithms is the storage and classification of information and the completion of partially given information in a foreseeing-step by calculating probabilities which are

based on statistical data. Moreover, mathematically speaking, existing information can be stored in a (dynamically adapted) matrix, and the classification step is executed by multiplying a vector with this matrix. Differences arise when the matrix and the vector elements are modeled. When searching for hyperlinks, this is a very difficult task: in this case, extracting appropriate keywords to retrieve important text attributes for later classification steps is rather delicate. Too many keywords will result in a very large matrix and thus in a long duration of the process of calculation. Too few or even false keywords, though, may lead to a wrong classification and inappropriate link-proposals. In the case of predicting user-requests, the modeling of a session-vector is a very simple and straightforward task. Nevertheless, calculation of costs for incorrectly predicted and pre-fetched data is highly complex. And even though wrongly proposed hyperlinks are somehow disturbing, wrongly predicted data can cause enormous network and system load and thus presents a much higher danger than the one mentioned before.

Table 1 shows a short and abstract characterization of the main tasks in predicting user-requests and proposing hyperlinks for texts as presented in the sections 2 and 3. In both cases, the core data structure is a matrix that stores the entire knowledge. The major difference is in the methods and the complexity of modeling the problem in order to gain vectors.

Phase	Step	Hyperlink-Proposals	Request-Prediction
Learn- ing phase	Retrieval of knowledge	Keyword extraction of hypertexts and attached links	Different document requests during the same session (with or without taking care of the order)
	Storing of knowledge	Vectorization of keyword attributes and storage of values in a relevance matrix (idea of case-based reasoning [20])	Transforming sessions into (mostly binary) vectors and storing their values in a memory matrix
Classi- fication phase	Recognition of information	Generating attributes (keywords) from texts (without considering link information)	Interpreting the first client request as ignition for calculation of relative frequencies
	Calculation of (likely) candidates	Multiplying the relevance matrix with the new attribute vector and thus getting link-relevance-probabilities → generating link-proposals	Multiplying the memory matrix with the current session-vector and thus getting relative probabilities for other documents to be requested soon → generating request-prediction

**Table 1.** Similarities between hyperlink-proposal and request-prediction algorithms

Still, the question remains: what is the advantage of recognizing similarities between link-proposals and request-prediction? Obviously, knowledge of the one can improve evolution steps of the other.

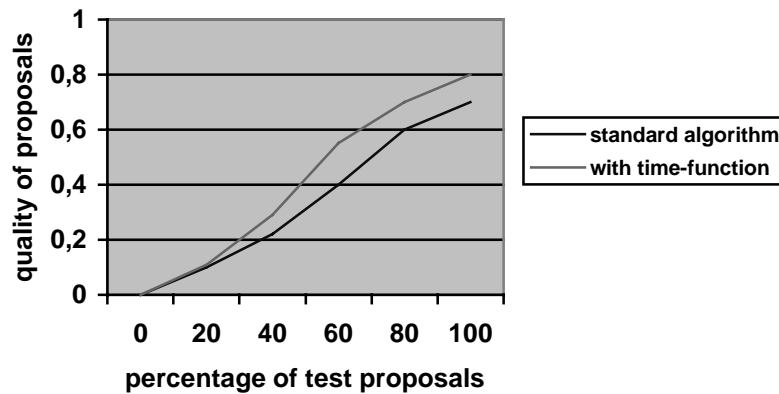
We will try to illustrate this through the following example. In [15], an advanced prediction model is shown where time and document aging is applied<sup>3</sup>. The evaluation of the usefulness of this approach is also confirmed. A time function  $f_t$  is applied to change the elements of the memory matrix  $M$  to model document aging (6) (corresponds to equation 4).

$$s_i = h \cdot g \cdot f_t(M) \cdot s^n. \quad (6)$$

But how can algorithms for the proposal of hyperlinks be improved by this perception? Time and aging also play an important role in the latter task as the relevance of former attributes changes. Analogously, the same time function can be used to change the relevance matrix in order to improve the learning and classification phases of the hyperlink-proposal algorithm (7).

$$r_i' = f_t f_i(R) \cdot t. \quad (7)$$

The usefulness of this idea is currently being evaluated by the authors and the first results are very promising. Figure 2 shows briefly the improvement-results for the introduction of the timing factor to the area of hyperlink-proposing algorithms after the initial tests.<sup>4</sup>



**Figure 2.** Improvements of hyperlink-proposal efficiency by using time-functions to model document aging

The quality of hyperlink-proposals rises for higher degrees of knowledge-base load. Using time-factor modeling the highest qualities can be increased about 10 percent above the standard values. Thus - presented as an example - ideas of prediction can help to improve the efficiency of proposing hyperlinks.

<sup>3</sup> General information on the topic of document aging can also be found in [11].

<sup>4</sup> More detailed results on that special topic will follow soon.

## 5 Summary and Outlook

In this paper, we presented two different problem categories of special modern Internet applications. The task of proposing hyperlinks for texts should help the author of a hypertext to improve the quality of his/her work by adding additional link information automatically. The prediction of requests aims at reducing the user perceived latency while waiting for an answer from the server.

Both problems - even though very different at a first glance - could be proved as similar during a closer analysis. We presented two mechanisms to solve these problems by learning and classifying and we elaborated their common properties.

With this knowledge, improvements on the one side should help to make progress on the other side as well. As an example of this idea we briefly outlined how to apply the concepts of modeling time and document aging of request-prediction to the area of proposing hyperlinks. The first results were very promising.

In the near future, besides evaluating exactly the advantages of time-modeling for the area of proposing hyperlinks, we will try to find several further aspects of the one side to improve the other and vice versa. Perhaps, it makes sense to use a single algorithm to solve both problem categories. Thus, we could try to find further areas that fulfill the same pre-conditions as prediction and link-proposals. Even though the comparison of different topics of modern Internet applications is very useful and advantageous, we should not forget that differences always remain and that the most difficult parts of problem aspects are, as a rule, quite unique.

## References

1. J. Allan. Automatic hypertext link typing. In *Proceedings of the Seventh ACM Conference on Hypertext (Hypertext '96)*, 1996
2. A. Bestavros, Speculative Data Dissemination and Service to Reduce Server Load, Network Traffic and Service Time in Distributed Information Systems, *Proceedings of ICDE'96*, New Orleans, Louisiana, March 1996
3. P. Cao, S. Irani, Cost-Aware Proxy Caching Algorithms, 1998
4. R. Cáceres, F. Douglis, A. Feldmann, G. Glass, M. Rabinovich, Web Proxy Caching: The devil is in the details, *Workshop on Internet Server Performance*, June 1998, Madison, WI
5. R. J. Chitashvili, R. H. Baayen. Word frequency distributions of texts and corpora as large number of rare event distributions. In *Quantitative text analysis, (Quantitative linguistics, Vol. 52)*, 1993. WVT Trier
6. C. Cleary, R. Bareiss. Practical methods for automatically generating typed links. In *Proceedings of the Seventh ACM Conference on Hypertext (Hypertext '96)*, ACM, 1996
7. D. T. Chang. HieNet: A user-centered approach for automatic link generation. In *Proceedings of the Fifth ACM Conference on Hypertext (Hypertext '93)*, ACM, 1993
8. E. Cohen, B. Krishnamurthy, J. Rexford, Improving End-to-End Performance of the Web Using Server Volumes and Proxy Filters, 1998
9. M. Crovella, P. Barford, The Network Effects of Prefetching, IEEE Infocom 1998
10. S. Erickson, D. Yi, Modelling the performance of a Large Multi-Tiered Application, American Management Systems, AMS Center For Advanced Technology, 1999

11. J. Griffioen, R. Appleton, The design, Implementation, and Evaluation of a Predictive Caching File System, CS-Department University of Kentucky, CS-264-96, 1996
12. W. Hall, H. Davis, G. Hutchings. *Rethinking Hypermedia: The Microcoms Approach*. Kluwer Academic Publishers, 1996
13. A. Heuer, E.-G. Haffner, U. Roth, Z. Zhang, T. Engel, C. Meinel. Hyperlink management system for multilingual websites. In *Proceedings of the Asia Pacific Web Conference (AP-Web '99)*, 1999. <http://www2.comp.polyu.edu.hk/~apweb99/>
14. E.-G. Haffner, U. Roth, T. Engel, Ch. Meinel, A Semi-Random Prediction Scenario for User Requests, *Proceedings of the Asia Pacific Web Conference, APWEB99*, 1999
15. E.-G. Haffner, U. Roth, T. Engel, Ch. Meinel. Modeling Time and Document Aging for Request Prediction - One Step further. *Symposium on Applied Computing, SAC2000*, Como, Italy, 2000
16. E.-G. Haffner, U. Roth, T. Engel, Ch. Meinel. Optimizing Requests for the Smart Data Server. *Applied Informatics, IASTED AI2000*, Innsbruck, Austria, 2000
17. E.-G. Haffner, A. Heuer, U. Roth, T. Engel, Ch. Meinel. Advanced Studies on Link-Proposals and Knowledge-Retrieval of Hypertexts with CBR. *Proceedings of the International EC-Web Conference, ECWeb2000*, Greenwich, United Kingdom, LNCS, Springer-Verlag, 2000
18. Z. Jiang, L. Kleinrock, An Adaptive Network Prefetch Scheme, *IEEE J Sel Areas Commun*, 1998
19. H. Kaindl, S. Kramer. Semiautomatic generation of glossary links: A Practical Solution. In *Proceedings of the Tenth ACM Conference on Hypertext (Hypertext '99)*, ACM, 1999
20. J. Kolodner, D. Leake. A tutorial introduction to Case-Based Reasoning. In *Case-Based Reasoning*. AAAI Press, the MIT Press, 1995
21. Achim Kraiss, Gerhard Weikum, Vertical Data Migration in Large Near-Line Document Archives Based on Markov-Chain Predictions, *Proceedings of the 23<sup>rd</sup> VLDB Conference*, 1997
22. A. Kraiss, G. Weikum, Integrated document caching and prefetching in storage hierarchies based on Markov-chain predictions, *The VLDB Journal*, Springer-Verlag, 1998
23. V.N. Padmanabhan, J.C. Mogul, Using Predictive Prefetching to Improve World Wide Web Latency, *Computer Communication Review*, 1996
24. F. J. Ricardo. Stalking the paratext: speculations on hypertext links as second order text. In *Proceedings of the Ninth ACM Conference on Hypertext (Hypertext '98)*, ACM, 1998
25. J.H. Rety. Structure analysis for hypertext with conditional linkage. In *Proceedings of the Tenth ACM Conference on Hypertext (Hypertext '99)*, ACM, 1999
26. Schechter, Krishnan, Smith, Using path profiles to predict HTTP requests. In *Proceedings of the World Wide Web Conference*, 1998
27. J. Tebbutt. Finding links. In *Proceedings of the Ninth ACM Conference on Hypertext (Hypertext '98)*, ACM, 1998
28. Z. Zhang, E.-G. Haffner, A. Heuer, T. Engel and C. Meinel. Role-based Access Control in Online Authoring and Publishing Systems vs. Documentation Hierarchy. In *Proceedings of the SIGDOC '99*, ACM, 1999