

# A Trail Based Internet-Domain Recommender System using Artificial Neural Networks

Tobias Berka, Wernher Behrendt, Erich Gams and Siegfried Reich

Salzburg Research - SunTREC  
Jakob-Haringer-Strasse 5/III  
5020 Salzburg  
AUSTRIA

**Abstract.** This paper discusses the use of artificial neural networks, trained with patterns extracted from trail data, as recommender systems. More specifically, feed-forward Multilayer-Perceptrons trained with the Backpropagation Algorithm were used to assign a rating to pairs of domains, based on the number of people that have traversed between them. This rating, applied to the hyper-graph neighborhood of an HTML document, can be used to suggest related domains to the user. The artificial neural network constructed in this project was capable of learning, and thus reproducing, the training set to a great extent. Outside of the training set, several experiments indicated that the artificial neural network becomes both capable of finding domains that are related, and an expert for domains that are relevant for the user community that produced the trail data.

## 1 Navigating the Web - Trails and Recommender Systems

In today's information society, people are faced with the problem of navigating information spaces every day. This creates a need for effective navigational aids. Recommender systems provide means for assisting users in the decision making process (for a discussion of recommender systems see [11]). Recommender systems are the technical response to the fact that we frequently rely on recommendations when confronted with decisions in a field where we have little or no knowledge. It is a recent development to see the process of navigation in the Internet not as an isolated procedure of a single user, but to make the net knowledge of individual users available to others.

This paper is structured as follows: Section 1 states the basic definitions, the main thesis and gives a brief summary of related work. Section 2 illustrates the neural net specific issues of this paper, namely the basic considerations (neural net construction and input space design), the generation of training sets and the final nets' learning and update behaviour. Section 3 describes first results of a user study, and section 4 summarizes the results and conclusions gained from them.

## 1.1 Definitions

The notion of a *trail* is an established concept in the field of hypertext navigation (see [2] and [9]). A trail is a sequence of *trailmarks*, each consisting of a *node* (representing a document), the *activity* performed by the *user* and other properties such as *time* and *duration*. The set of all trailmarks *Trailmarks*, and a single trailmark *tm* are being defined as follows:

$$\text{Trailmarks} = \text{Node} \times \text{Activity} \times \text{Date} \times \text{Duration} \times \text{User}$$

$$tm \in \text{Trailmarks}.$$

We then define a trail *t* as a string of trailmarks:

$$t = tm_1tm_2tm_3\dots tm_n \in \text{Trailmarks}^*$$

Considering the trail based approach in the context of recommender systems leads one to the following issues:

- The training data is obtained through implicit voting, where the action of following a link from one domain to another is a vote for this pair of hosts. Thus the trail data of a single user only contains pairs of nodes that were given a positive vote. Since there is only one value for a positive vote (visited), the raw trail data is totally uniform regarding the vote value and thus not suitable for training an artificial neural network (ANN). Therefore the training sets for an ANN have to be obtained by modifying the trail data.
- Secondly, the trail data of a single user contains some of the knowledge a user has about the respective information space. This means that the users' trails in the Internet contain information about their net knowledge and habits, suggesting a collaborative, or social [5], approach, in which training sets are obtained by combining the trails of different users, which of course raises major privacy considerations. The users' expectations concerning their privacy could be fulfilled by making the process of activating the trail data logging an explicit action available only to the respective user. This, combined with the idea of making the individual user's trails anonymous before adding them to the pool, should provide sufficient privacy.

The previous considerations allow us to define a mathematical relation *R* as follows:

Let *D* denote the set of all domains,  $R \subset D \times D$  denote a relation over *D*, and let  $d_1, d_2, d_3 \in D$  be domains. The statement  $d_1 R d_2$  expresses that the domains  $d_1$  and  $d_2$  are related. We will define this relation *R* as follows:

- $\forall d_1 \in D : d_1 R d_1$ . Reflexivity holds for *R*. Quite trivially, every domain is theoretically related to itself. However, it makes little sense to include it in the list of search results.
- $\forall d_1, d_2 \in D : d_1 R d_2 \Rightarrow d_2 R d_1$ . Symmetry holds for *R*. The decision to include symmetry was intended to reduce the size of the training set and increase the number of higher ratings. This may be contrary to some situations of a colloquial understanding of the term *related*, but is necessary to gain the benefits stated above.

- $\neg(\forall d_1, d_2 \in D : d_1 R d_2 \wedge d_2 R d_1 \Rightarrow d_1 = d_2)$ . Anti-Symmetry does not hold for  $R$ .
- $\exists d_1, d_2, d_3 \in D : \neg(d_1 R d_2 \wedge d_2 R d_3 \Rightarrow d_1 R d_3)$ . Transitivity does not hold for all  $d_1, d_2, d_3 \in R$ . This is obvious when considering a domain 1, relevant for the user communities A and B. It is related to a domain 2 of the communities B and C. Domain 3 deals with topics of communities C and D, and is thus related to domain 2. But it is *not* related to domain 1, which would be induced for all such triples of domains if transitivity would hold.

In the context of a trail based system, two domains are considered to be related if users often traverse between them. Introducing a rating from the interval  $[0..1]$ , based on the number of users that move between two domains, leads us to a function  $r$  that assigns a real valued rating to each pair of domains, lower for domain pairs with few traversals, higher for domain pairs with many traversals. We can thus define the relation  $R$  as follows:

$$\forall d_1, d_2 \in D : \quad a R b \quad \text{iff} \quad r(d_1, d_2) > \varepsilon,$$

where the threshold  $\varepsilon$  is a number between 0 and 1. Its value has to be determined experimentally (we used a value of  $\varepsilon = 0.5$ ). In a later state, this binary logic (necessary to define a single relation) can be extended to a three-valued system, with intervals for the three states, namely *not related*, *undecided* and *related*. To ensure the symmetry of the relation  $R$  the following must hold for the function  $r$ :

$$r(d_1, d_2) = r(d_2, d_1).$$

This paper deals with the idea of using artificial neural networks to extract information about rules in such trails. For this paper, a rule is a correlation between the domain name pairs and the corresponding value of the function  $r$ , which holds for a sufficient number of domain pairs in the trails, so that an ANN can generalize this correlation and apply it to domain pairs that are not part of the trail data, and are thus unknown to the system. This ANN is then to be used in a recommender system.

We have constructed an ANN, which is fed with two domain names and returns a relation rating from the net's output neuron. This means that if a useful search space can be generated, a list of related documents can be obtained. To gain the search space, we generate the hypergraph of a given HTML-document seed, and compare its domains with the domain of the seed. This results in a quite long computational time of the algorithm, based mainly on the speed of the user's Internet connection, but has the advantage that it can be used for any HTML document, since it does not issue queries to databases, and is thus operable in previously uncharted areas of the Internet.

## 1.2 Related Work

Since we operate on domains only, other papers dealing with ANNs operating on domains or URLs, and machine learning for web browsing in general, were

of interest. D. Mladenic and M. Grobelnik [8] have researched the use of ANNs for predicting the content of an HTML document from the link that points to it. Since we have a content-independent approach, we differ somewhat from this approach.

A good source of information concerning the accuracy of common collaborative recommender system algorithms was the study performed by J. S. Breese, D. Heckerman and C. Kadie [1]. Their paper contains a description of some of the classical algorithms that are also known as *collaborative filtering*. These are adaptive algorithms designed to predict a users vote (e.g. for a product of an e-commerce site). The vote is computed by weighting the other users' votes for this product, based on comparisons between the users' votes on other, or representative, products. One of these algorithm uses Bayesian Networks, which differ from our approach mainly due to the fact that their structure is modified as the number of votes increases. The ANN used in our approach has a fixed number of neurons and connections. Learning occurs only through connection weight changes.

Another approach of interest deals with the processing of binary votes in collaborative filtering [15], having quite a different approach due to the basic structure of the recommender used.

In our approach the users are not grouped as the system receives data, but before the system is launched, since the system recommends domains based on the Internet habits and knowledge of all users in the predefined group. The groups used in our experiments were based on membership to sections of a research organization with different research divisions, and thus different navigation interests (see below for further information).

There are various other studies that have a text-based approach, recommending HTML documents based on their content, either using the vector representation, or bag-of-words approach (e.g. see [7]), or using latent semantic indexing, which leads to a language independent approach (as described in [6]). But neither of these allow the consideration of images, sound files or other non-plain-text data formats, which may well be part of a trail - and thus the training set - because this approach disregards content.

It was our intention to model a content independent, trail based approach. The only information we had about the documents were the URLs, and thus the domain names, and in what sequence the documents were visited. Therefore, our motivation was to associate the domain name pairs with the number of people that traversed between them.

Since we have an approach similar to that of social filtering, a paper of interest was [12], providing descriptions for a framework for hypertext-navigation with social aspects.

## 2 Artificial Neural Networks in a Trail Based Recommender System

The ANNs used for our experiments were all feed-forward Multilayer-Perceptrons trained with the Backpropagation Algorithm (as proposed in [13]).

### 2.1 Comparative Rating of Domain Names with ANNs

Using strings as input values for ANNs induces a number of problems. Strings must have a fixed length, or a maximum length, to be used as input vectors since the number of input neurons has to be fixed. Secondly, the dimension of the input space increases rapidly for longer strings. Since it is hard to compress strings non lossy and still keep a well defined input space, the number of dimensions in the input space is usually equal to the (maximum) number of characters in the string. This causes nets operating on strings to have a very high number of connections, resulting in long learning times. Some experiments conducted early during this project indicated that ANNs operate better on binary encodings of strings. This, of course, increases the input dimension again. domain names have an upper limit of 63 characters, but since the number of domains with a name consisting of more than 25 characters is as low as 0.8% (see [http://www.zooknic.com/Domains/dn\\_length.html](http://www.zooknic.com/Domains/dn_length.html)), we used 26 characters to encode the domain name. These 26 characters are chosen from a set of 38 characters (`{'a', ..., 'z', '0', ..., '9', '-', '.'}`), thus the number of bits needed to encode a whole domain name is 156 bits (6 bits per character  $\times$  26 characters).

One of the first considerations was to have all domain name parts starting at fixed positions. This increased the number of characters from 26 to 40, raising the number of bits needed to encode the string to 240 bits. Since two domain names form an input pattern, the total number of bits, and input neurons, is 480, which is acceptable in terms of training speed. A higher number of neurons would cause a higher number of connections, and since the learning algorithm trains the net by adapting the weights of the connections, the learning time is highly dependent from the number of connections and weight changes that have to be computed.

### 2.2 ANN Structure Used in Our Experiment

The neural network used for the Trail Based Recommender has a 480-24-24-4-12-1 architecture with a low degree of connectivity (a total of 1436 edges), but with shortcut connections. This architecture was created during a number of experiments with ANN architectures. The neurons in the hidden and output layer all have the logistic activation function, which is a sigmoid function. It was trained using the Stuttgart Neural Net Simulator (SNNS v 4.2, see <http://www-ra.informatik.uni-tuebingen.de/SNNS/>) using standard Backpropagation.

### 2.3 Obtaining Training Sets

The main motivation for this project was to use trail data for the prediction of related domains. We used proxy access logs as a source of primer trail data. We extracted all successful GET accesses of the research and development team of an IT enterprise and split them into (anonymous) user trails. These user trails were then analyzed in order to generate training data as described below:

If  $N(domA, domB)$  is the number of times the domain names  $domA$  and  $domB$  appear as neighbors in a trail, the relation rating  $r$  computes as follows:

$$r(domA, domB) = \left( 1 - \frac{1}{1 + N(domA, domB)} \right)^n,$$

where  $n$  has to be adjusted to give a sufficiently broad number (we used a value of  $n = 4$ ). If the function  $s$  assigns the binary encoding to every pair of domains, a learning task  $L$  is defined as follows:

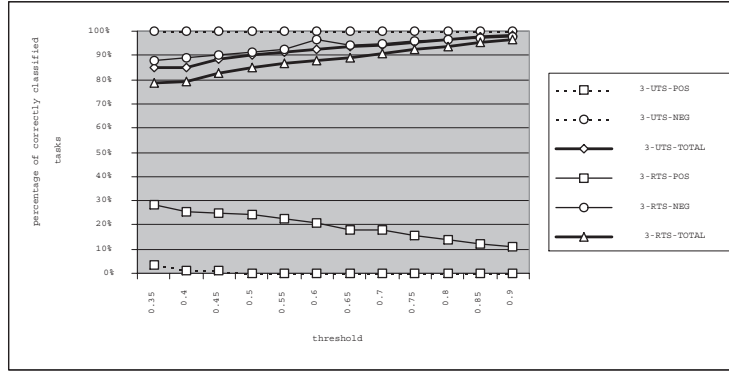
$$L = ((s(domA), s(domB)), r(domA, domB))$$

We used the proxy access logs of two months, limited to 47 research and development users, from which some 30,000 training tasks per log file could be extracted.

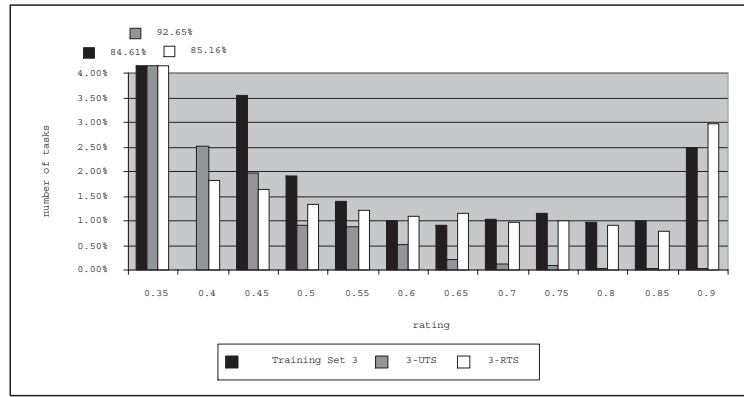
### 2.4 Learning and Update Aspects

The final version of the Trail Based Recommender will use trails collected by the TrailBlazer framework, which is developed for the Trailist project (see [10]). A first analysis of the training set showed that they consist mostly of totally unrelated domain pairs, with a rating below 3.5. An ANN trained with this raw data obtained an MSE of 0.044 and correctly classified some 90.1% of these tasks, but more detailed analysis of these results showed that this high percentage is based mostly on correct classification of the negative training tasks with a rating below 0.5. Further experiments with a selectively reduced training set, in which most examples with a rating of below 3.5 were removed, showed an *increase* in error on the whole training set (MSE 0.079), as well as a decrease of correctly classified negative tasks, but an increase in the percentage of correctly classified positive tasks, which is desirable for our project. Figure 1 depicts the percentage of correctly classified positive, negative and total tasks for ANNs trained with the unmodified training set 3 (3-UTS) and the reduced training set 3 (3-RTS), sampled over the range of the threshold  $\varepsilon$ . A quantitative analysis of training set 3 and these nets is given in Figure 2.

The generalizational capacity of net 3-RTS was then tested on the training set generated from the proxy access log of the following month - training set 4. The result of this evaluation is depicted in Figure 3, along with the results of the net obtained by training with the reduced training set 4, denoted by 4-RTS. Figure 4 shows the quantitative analysis for this training set, the net trained



**Fig. 1.** Qualitative ANN Analysis on Training Set 3



**Fig. 2.** Quantitative Training Set and ANN Analysis on Training Set 3

with the entire training set 4 (4-UTS) and the RTS nets of the current and previous training set.

We also tested the generalizational capacity of the 3-RTS net on a training set generated from the trail data of a training unit of an IT corporation, having somewhat different Internet habits than the research and development unit. For a threshold of  $\varepsilon = 0.5$  it correctly classified some 80.43 % of all tasks, but only 19.61 % of the positive tasks, having an MSE of 0.103 on this training set. The learning behaviour and generalizational capacity of these nets suggests monthly update steps with reduced training sets generated from the respective month's trail data, as well as a priori user grouping by interest and research topics.

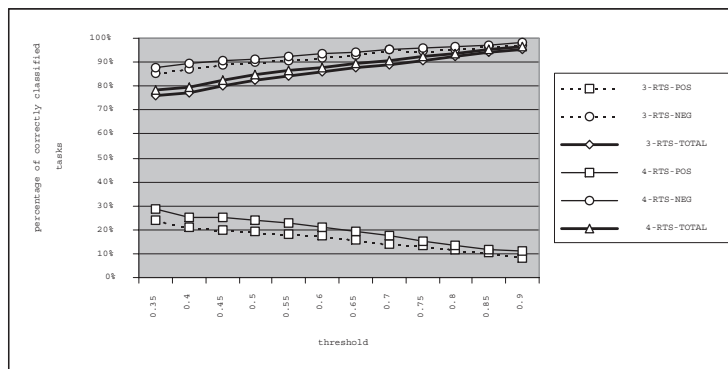


Fig. 3. Qualitative ANN Analysis on Training Set 4

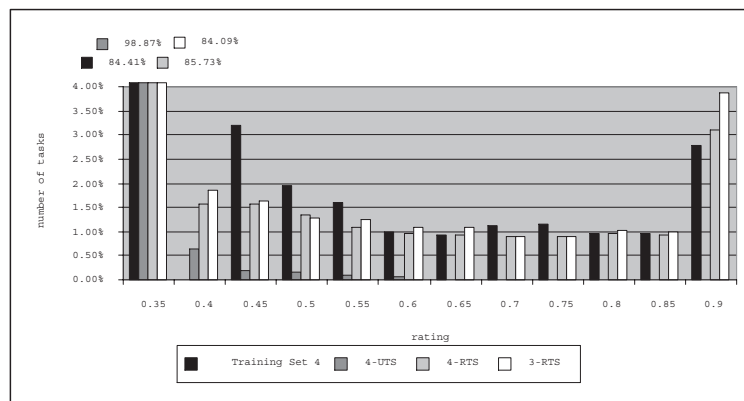


Fig. 4. Quantitative Training Set and ANN Analysis on Training Set 4

### 3 First Test Runs and Experiences

In order to test the ANN, we integrated a sidebar into Netscape 6.2, and used a breadth-first traversal in the hypergraph-neighborhood of the current HTML document to obtain a search space, an algorithm which is time consuming. It is obvious that the response times may vary greatly based on the speed of the Internet connection and the performance of the user's computer.

First test runs performed by ten users with varying IT skills and educational backgrounds with the prototype of the TrailBlazer framework indicated that the algorithm's time consumption is acceptable, if it can be used in a "single shot" manner, being launched by the user to find related domains to the current HTML document, or if other non-real-time methods of user notification as e.g. in [3] are used. Another important issue is the presentation of the algorithm's results. Since the algorithm parses a great number of documents, simply displaying the

domain name or title of the HTML document residing at that address may lack transparency, a problem which can be evaded by presenting the results in a manner similar to search engines (by displaying link target title, meta-data or excerpts).

The advantage of the algorithm is however that it can exploit the generalizational property of the ANN, and thus operate on unknown and yet unmapped regions of the net, which clearly distinguishes it from other approaches to the “What’s Related” - Problem (as described in [4] or [14] in detail).

Summarizing, the algorithm and ANN seem to perform well together, though there is one restriction one has to bear in mind: the algorithm can never find related documents that are either far away from the seed, or not connected at all. The advantage is however that it can process a large number of documents in the neighborhood of the seed. This, combined with the fact that the search is highly non-linear leads to the advantage that the algorithm may return domains of links the user will never, or only after a long search, come to.

## 4 Summary and Conclusions

We have developed various components for a recommender system for obtaining related domains utilizing a neural network. The individual components incorporated an ANN computation package, the HTML Neighborhood search algorithm and various administrative tools for the extraction of trail data and training sets from proxy access logs.

The experiments with Multilayered Feed-Forward Perceptrons have indicated that this classical neural network model, combined with the Backpropagation algorithm, can be used in recommender systems, if the dimension of both input and output space is fixed.

Another indication of the experiments was that there are certain patterns in the domain sequences extracted from user trails, as expressed in the generalizational capacity of the ANN, and that these patterns can be learned by artificial neural networks with an acceptable amount of error. Furthermore, it seems adequate to use proxy access logs as priming data for this type of recommender.

Further experiments should incorporate mapping several neighborhoods of HTML pages and the analysis of the hypergraph for related domains by the people whose trail data was used to train the ANN. On this validation data the overall performance of the ANN could be tested, but this is clearly beyond the scope of our study. Possible future uses of the ANN Recommender would be trail similarity matching, measuring the distance between the user’s current trail and others in a trail database, in order to find the *best match*, and trail segmentation, allowing the system to split long trails into smaller trails, a mechanism that could be used if other trail recording mechanisms fail to provide a satisfying segmentation.

### ACKNOWLEDGEMENTS

This work has been supported in part by the Austrian *Fonds zur Förderung der wissenschaftlichen Forschung* (FWF) under grant No. P14006-INF.

## References

1. John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 43–52, July 1998.
2. Vannevar Bush. As we may think. *The Atlantic Monthly*, 176(1):101–108, 1945.
3. Leslie A. Carr, Wendy Hall, and Steve Hitchcock. Link services or link agents? In *Proceedings of the '98 ACM Conference on Hypertext, June 20-24, 1998, Pittsburgh, PA*, pages 113–122, 1998.
4. Jeffrey Dean and Monika R. Heintzinger. Finding related pages in the world wide web. In *Proceedings of the Eighth World-Wide Web Conference*, pages 1467–1479, 1999.
5. Andreas Dieberger. Supporting social navigation on the world wide web. *International Journal of Human-Computer Studies, special issue on innovative applications of the Web*, 46(6):805–825, 1997.
6. Susan T Dumais, Todd A. Letsche, Michael L. Littman, and Thomas K. Landauer. Automatic cross-language retrieval using latent semantic indexing. In *AAAI Symposium on CrossLanguage Text and Speech Retrieval*. American Association for Artificial Intelligence, March 1997.
7. Dunja Mladenic. Text-learning and related intelligent agents. *IEEE Expert Special Issue on Applications of Intelligent Information Retrieval*, July 1999.
8. Dunja Mladenic and Marko Grobelnik. Predicting content from hyperlinks. In *Proceedings of the ICML-99 Workshop on Machine Learning in Text Data Analysis*, 1999.
9. Siegfried Reich, Leslie A. Carr, David C. DeRoure, and Wendy Hall. Where have you been from here? Trails in hypertext systems. *ACM Computing Surveys — Symposium on Hypertext (published as electronic supplement)*, 31(4), December 1999.
10. Siegfried Reich and Erich Gams. Trailist - focusing on document activity for assisting navigation. In *Proceedings of the Twelfth ACM Conference of Hypertext and Hypermedia*, pages 29–30, August 2001.
11. Paul Resnick and Hal R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, March 1997.
12. Mark O. Riedl. A computational model and classification framework for social navigation. In *Proceedings on the International Conference on Intelligent User Interfaces*, pages 1–8, January 2001.
13. D. Rumelhart, G. Hinton, and J. McClelland. Learning internal representations, 1986.
14. Masashi Toyoda and Masaru Kitsuregawa. Creating a web community chart for navigating related communities. In *Proceedings of the Twelfth ACM Conference of Hypertext and Hypermedia*, pages 103–112, August 2001.
15. Sholom M. Weiss and Nitin Indurkha. Lightweight collaborative filtering method for binary-encoded data. In *Fifth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, 2001.