Inducing Content Based User Models with Inductive Logic Programming Techniques

Martin E. Müller[1]

Institute for Semantic Information Processing, University of Osnabrück, Germany
Martin.Mueller@cl-ki.uni-osnabrueck.de

Abstract. In this paper we describe an approach for conceptual user modeling as realized in the OySTER meta web search engine. Instead of collaboratively modeling user interests we use web document classifications in order to describe individual user models. Information relevance is expressed with respect to an underlying ontology of text categories. Logic expressions over semi–lattices are interpreted as horn clauses—thus allowing to prove different levels of interestingness. Furthermore, this approach presents a well–defined learning problem for inductive logic programming which yields inspectable user models that include sets of interest aspects. By using both explicit positive and negative feedback for both interest and explicit dis–interest we can use few examples to generate a larger set of labeled data for the learning task.

1 Introduction

Adaptive information retrieval has become a key discipline ever since the amount of information has exploded with the advent of the world wide web. Since individual user modeling on such huge domains is a tough topic, adaptivity in so–called recommender systems mainly is achieved by collaborative user modeling.
Nevertheless, individual user modeling can be carried out provided suitable feedback. Syskill&Webert, [15], is a recommender system that consists of a meta search engine which for each result offers the opportunity to give explicit feedback. Feedback is used to build an individual user model that contains sets of boolean key word vectors. Using trained Bayesian classifiers, web documents are recommended with respect to the user model.
The WebWatcher (see [1, 6, 7]) is a more unobtrusive approach to the same domain. Without explicit feedback, links are recommended during a web browsing session. Browsing behavior is recorded in order to build a user model, again consisting of word vectors that have been derived using TfIdf. Collaborative user modeling is performed using reinforcement learning. The Personal WebWatcher is a system for web page recommendation based upon individual user modeling techniques, [9].
The Ontobroker was a first attempt to define a structured part of the world wide web which allows for content based information retrieval, relying on manually added meta information about a web document's content, [4]. But, the Ontobroker did not allow for user adaptive information retrieval. A further step into

this direction is the WebKB project, where from a part of the world wide web its underlying structure and information content is extracted by methods of inductive logic programming in order to yield a clear description of web document contents and relations, [3].
In the intersection of those approaches, we defined the adaptive meta search engine OySTER.

2 Adaptive Web Search within OySTER

OySTER is a meta search engine for the world wide web. (`http://mir.cl-ki.uni-osnabrueck.de/oyster/`; a more pictorial overview is given in [14]).
It is realized as a multi agent system which allows for an any–time response behavior: Regardless to the current state of responding search services, wrappers or classifiers, the best match to the submitted search query is presented first, followed by an ordered list of all other results. Currently, search results can be ordered with respect to an arithmetic relevance measure (depending on search engine result ranks) and document types and/or categories. Ordering with respect to the user model currently is performed offline.
Instead of defining collaborative user models that are based on word frequencies, we classify web documents with respect to document type and category ontologies. Thus, any document d is represented by a pair

$$\mathcal{C}(d) = \langle t : p, \langle c_1 : p_1, c_2 : p_2, c_3 : p_3 \rangle \rangle \tag{1}$$

with a type t and categories c_i of decreasing confidence values p_i.
Presupposing perfect classifiers, we forget about words contained in d and work on the classification data instead. This motivates the very high idea behind OySTER: Given sufficient evidence for a user's interest, that is a set of conceptual descriptions with relevance feedback f according to some interest aspect a

$$\mathbf{s}_u = \{ \langle \mathcal{C}(d), f_a \rangle_i \, | i \in I \}, \tag{2}$$

one should be able to induce a user model as a hypothesis which describes the user's interest:

$$M_u \models \mathcal{C}(d) \text{ iff } u \text{ is interested in } d \text{ with respect to aspect } a \tag{3}$$

The second idea is that given explicit models of what the user is interested in (M_u^+) and what he is not interested in (M_u^-), each again consisting of different aspects, can be used to prove (or reject) assertions about whether some document d might be interesting or not, [13, 12].

3 Learning User Models by Inducing Logic Programs

In our approach, a user model is represented in terms of document categories.

The category hierarchy has been handcrafted for describing the computer science academic sub–part of the web (parts of it are described in [14]). The category classifiers were developed in course of a student's project, Bikini.[1]
The document type classifier works on regular expressions on document URLS, [5].

3.1 Representing User Models and Feedback Data

In order to be able to induce user models, we also need a suitable representation according to the learning algorithm.
In a more more pictorial view, the document category hierarchy is a tree. Nevertheless, inheritance on concept hierachies used for describing interests is not an easy matter: Interest in a category c does not neccessarily imply interest in category c', where c subsumes c' ($c \sqsubset c'$). The reverse case does not hold either; though interest in c' "supports" interest in c and interest in c can be "explained" through interest in c'. Therefore, we use a representation of our hierarchies based upon entailment and taking into account beforementioned drawbacks in the section about using user models for filtering.
Thus, the document category hierarchy is represented as a set of Horn clauses which models inheritance through entailment:[2]

```
cat_..._cs(X,D) :-
  cat_..._cs_programming(X,C), genthresh(C,D).
cat_..._cs_programming(X,D) :-
  cat_..._cs_programming_languages(X), genthresh(C,D).
cat_..._cs_programming_languages(X,D) :-
  cat_..._cs_programming_languages_procedural(X), genthresh(C,D).
```

Similary, document classification data as required by the sample specification in (1) is represented as facts:

```
type_..._publication_researchpaper(urlid_5121,68).
cat_..._intelligence_machine_learning_symbolic(urlid_5121,92).
cat_..._intelligence_machine_learning_subsymbolic(urlid_5121,20). [...]
```

User Models. Now, an user model is a 'set of subtrees', where each subtree either represents the user's interest or explicit dis–interest. More formally, the user model M_u consists of a pair of sets $\langle M_u^+, M_u^- \rangle$ and each set consists of Horn clauses describing *aspects*. Since a user can be interested in several, distinct topics—say, machine learning and diving—describing the user's interest by a

[1] Bikini is an user adaptive news reader, [2]. The classifiers are simple vector space classifiers, where n-gram vectors (i.e. phrase vectors) have been generated by a kind of a boot–strapping method: To each category we collected a small set of "key–phrases". Each subset of those sets was sent to the meta–search engine and relevant words were extracted from the result documents by a TFIDF measure.

[2] The `gentresh` predicate penalizes generalization during the induction process by demanding varying class membership values.

category that tries to unify those two aspects would lead to very bad results. In general, M_u^+ contains clauses like

$$\begin{aligned}(4)\qquad & \texttt{p\_interest\_}u(a,d,r) :- \\ & \quad \texttt{type\_\_}t_1(d,tc_1),...,\texttt{type\_\_}t_{n_t}(d,tc_{n_t}), \\ & \quad \texttt{cat\_\_}c_1(d,cc_1),...,\texttt{cat\_\_}c_{n_c}(d,cc_{n_c}), \\ & \quad \texttt{thresh}(V_1,\vartheta_1),...,\texttt{thresh}(V_n,\vartheta_n).\end{aligned}$$

where u is the user id, a the aspect id and d the document id under consideration. Document types (t_i) and categories (c_j) are assigned confidence values tc_i and cc_j, respectively. Finally, `thresh`olds can be defined in order to require a certain value V_k (one of $\{tc_1,...,tc_{n_t},cc_1,...,tc_{n_c}\}$) to be greater than a certain boundary ϑ_k. Currently, `thresh` is realized by the two relations $<$ and $>$.

User Feedback. Initially, we are given feedback (ratings $f \in \{-2,1,0,1,2\}$) for a set of documents d_i which is interpreted as feedback with respect to categories. In our system, we simulated such samples by a randomized distribution over a set of centroids in the category hierarchy based upon a asymmetric distance measure δ. The distance measure $\delta(n,m)$ used is asymmetric; it is defined by the sum of costs from n (up) to l and from l (down) to m, where l is the least upper bound of n and m and generalization (up) and specialization steps (down) are penalized differently.[3] This approach might be questionable but actually yielded feedback which very much resembled real data (including "nasty" users). A part of sample is displayed with respect to the underlying category in figure 1.[4] User feedback is both stored as factual knowledge and examples for our learning task (note, that a single feedback event concerning a document is used to generate examples for both `p_interest` and `n_interest`; this method can be expanded to multiple aspects as well):

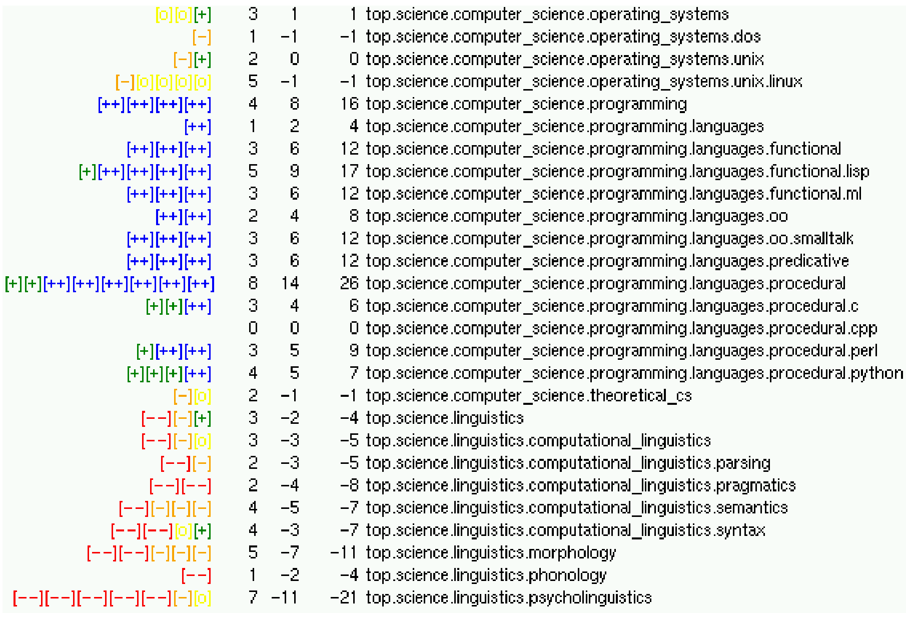
```
   p_interest_88(urlid_5232, 20).     :- p_interest_88(urlid_5234, 20).
:- n_interest_88(urlid_5232, 20).        n_interest_88(urlid_5234, 20).
```

3.2 Induction of User Models

Interpreting feedback f as a 'noisy subset' of the user's interest $\mathfrak{I} = \langle I^+, I^- \rangle$, we want to accurately approximate $\mathfrak{I}$. Given the document type and category

[3] The closer we get to the leaves, the cheaper are the edges (this is motivated by growing similarity in those classes; leaf siblings like *symbolic machine learning* and *subsymbolic machine learning* are closer to each other than top–level siblings like *linguistics* and *computer science*). Furthermore, longer generalization paths need to be penalized stronger than shorter ones. Thus, $\delta(n,m) \neq \delta(m,n)$ for two categories n and m of different depth. In our category hierarchy, this means that *user modeling* is less related to *symbolic machine learning* than vice versa.

[4] Note, that the visualization veils some important information: Each feedback entry (indicated by square brackets) for a category c corresponds to a feedback event (which is given with respect to URLS), where the *most confident* classification of the URL was c. Nevertheless, the same URL also contributes to feedback data with respect to the two other classes which is *not* displayed here.



[o][o][+]	3	1	1	top.science.computer_science.operating_systems
[-]	1	-1	-1	top.science.computer_science.operating_systems.dos
[-][+]	2	0	0	top.science.computer_science.operating_systems.unix
[-][o][o][o][o]	5	-1	-1	top.science.computer_science.operating_systems.unix.linux
[++][++][++][++]	4	8	16	top.science.computer_science.programming
[++]	1	2	4	top.science.computer_science.programming.languages
[++][++][++]	3	6	12	top.science.computer_science.programming.languages.functional
[+][++][++][++][++]	5	9	17	top.science.computer_science.programming.languages.functional.lisp
[++][++][++]	3	6	12	top.science.computer_science.programming.languages.functional.ml
[++][++]	2	4	8	top.science.computer_science.programming.languages.oo
[++][++][++]	3	6	12	top.science.computer_science.programming.languages.oo.smalltalk
[++][++][++]	3	6	12	top.science.computer_science.programming.languages.predicative
[+][+][++][++][++][++][++][++]	8	14	26	top.science.computer_science.programming.languages.procedural
[+][+][++]	3	4	6	top.science.computer_science.programming.languages.procedural.c
	0	0	0	top.science.computer_science.programming.languages.procedural.cpp
[+][++][++]	3	5	9	top.science.computer_science.programming.languages.procedural.perl
[+][+][+][++]	4	5	7	top.science.computer_science.programming.languages.procedural.python
[-][o]	2	-1	-1	top.science.computer_science.theoretical_cs
[--][-][+]	3	-2	-4	top.science.linguistics
[--][-][o]	3	-3	-5	top.science.linguistics.computational_linguistics
[--][-]	2	-3	-5	top.science.linguistics.computational_linguistics.parsing
[--][--]	2	-4	-8	top.science.linguistics.computational_linguistics.pragmatics
[--][-][-][-]	4	-5	-7	top.science.linguistics.computational_linguistics.semantics
[--][--][o][+]	4	-3	-7	top.science.linguistics.computational_linguistics.syntax
[--][--][-][-][-]	5	-7	-11	top.science.linguistics.morphology
[--]	1	-2	-4	top.science.linguistics.phonology
[--][--][--][--][--][-][o]	7	-11	-21	top.science.linguistics.psycholinguistics

(Truncated example)

Fig. 1. Relevance feedback for documents with respect to categories.

hierarchies and URL classifications as background knowledge Σ, we want to induce a hypothesis h using a sample $\{\langle \mathcal{C}(d), f_a \rangle_i \, | i \in I\}$ such that:

$$\begin{aligned} \Sigma \cup h &\models \texttt{p\_interest\_}u(a,d,r) \quad \text{iff} \quad I^+(d) = 1 \\ \Sigma \cup h &\models \texttt{n\_interest\_}u(a,d,r) \quad \text{iff} \quad I^-(d) = 1 \end{aligned} \tag{5}$$

For h, several restrictions apply: We know the argument structure of the clause head and the set of possible body clauses thus yielding a strong bias in terms of so–called *mode–declarations*

The learning set includes 10,000 URLs which were randomly assigned category and type classifications.[5] For each of the ten users we generated feedback. In order to simulate feedback, for each user up to three centroids on the category hierarchy were chosen randomly; feedback was generated using a δ–distorted distribution around those centroids. For each centroid per user, 200 evidences were generated.

Induction of User Models. Using Progol 4.4 (see [10, 11] and following) we induce a sequence of hypotheses describing the user's interest based upon the sample. Progol 4.4 is based on the inverse entailment method: From background knowledge Σ and examples E a set of ground literals $\overline{\text{msc}}$ is deduced, the conjunction

[5] This method guarantees that our input data is noisier than one would expect it in real world samples.

of which is true in all models of $\Sigma \wedge \overline{E}$. Then, a hypothesis can be induced by searching for $h \preccurlyeq_{\theta} \mathsf{msc}$. As an example, consider figure 1. The rules induced on the underlying feedback data are:[6]

```
p_interest_88(A,B) :-
  cat__top_science_computer_science_programming_languages(A,B).

n_interest_88(A,B) :-
  cat__top_science_linguistics(A,B).
n_interest_88(A,B) :-
  cat__top_science(A,B),
  cat__top_science_computer_science_artificial_intelligence(A,C).
```

Those three rules covered ten evidences all together. From other samples more complex rule sets like those in figure 2 have been derived. User 90 seems to

```
p_interest_90(A,B) :-
  cat__top_science_computer_science(A,B),
  cat__top_science_computer_science_[ai]_machine_learning(A,C),
  C>56.
p_interest_90(A,B) :-
  type__top_publication(A,C),
  cat__top_science(A, B).
%-------------------------------------------------------------
p_interest_93(A,B) :-
  cat__top_science_computer_science_programming(A,B),
  cat__top_science_computer_science_artificial_intelligence(A,C).
n_interest_93(A,B) :-
  cat__top_science_computer_science_programming(A,B),
  cat__top_rec_sports_water_scuba_diving(A,C).
n_interest_93(A,B) :-
  cat__top_science(A,B),
  cat__top_science_computer_science_operating_systems_dos(A,C).
```

Fig. 2. Rules describing a user's interest

be interested in any 'publication'–like document about 'science'. Furthermore, any document about 'computer science' and 'machine learning' is relevant, if the confidence for 'machine learning' is at least 57. User 93 is interested in 'programming' if it coincides with 'artificial intelligence'—but definitely is not interested in documents about 'diving computers' or 'DOS'.
We have generated different feedback sets for ten simulated users, where for two feedback sets Progol did not deliver any compressing rule at all. [7] Results are

[6] Note, that the predicates used here do **not** take into account multiple aspects a of the user's interest; i.e. the predicate is missing one argument.

[7] In the first case this was due to a nearly equally distributed feedback. In the second case, we exceeded the search depth limit.

	`p_interest`			`n_interest`			
user	*cov*	*acc*	$r_{\mathrm{p}}(c_{\mathrm{p}})$	*cov*	*acc*	$r_{\mathrm{n}}(c_{\mathrm{n}})$	time
88	93.8%	72.2%	1(2)	65.6%	26.1%	2(8)	2'07"
92	51.7%	52.4%	1(2)	57.1%	29.5%	4(15)	4'07"
93	41.6%	57.8%	3(12)	32.2%	26.8%	4(12)	7'54"

r_i is the number of rules induced for the target i; c_i is the number of facts (i.e. evidences) that are covered by the rules.
Note, that accuracy and coverage are computed only with respect to rules which actually compressed the sample; remaining rules covering only single evidences are not taken into account.

Table 1. Coverage & accuracy of induced rules

shown in table 1. The feedback given by user 88 is shown in figure 1. His interested was modeled by a single centroid (located in the 'procedural programming' tree). Most negative feedback was given in the category `linguistics` which formed a very clear image. Accordingly, only three rules were induced which deliver a considerable high coverage and accuracy for `p_interest`. Since `n_interest` is modeled by low δ values instead of special centroids, the training data is unspecific and rather noisy.

A growing number of centroids chosen within the feedback simulation function corresponds to multiple aspects in a user's interest. Since multiple aspects were not covered in the first test series, results are rather bad (the image becomes blurred): User 92's interest was simulated using two centroids that were both located in the upper levels of the ontology's 'science' part thus yielding a rather uniform distribution of positive feedback with average noise of negative feedback. The large numbers of rules for `n_interest` can be explained by the noisy negative feedback of the large positive field which also might explain the slightly better result for accuracy of `n_interest`. Most important is the dramatic decrease in coverage and accuracy of `p_interest`, though the latter one can be easily explained by inducing only one rule which subsumed the 'linguistics' branch and left out the whole branch of 'computer science' (containing approximately 70% of all positive feedback).

Finally, user 93, whose interest was defined by three centroids, showed worst results. Seven rules were induced, three for `p_interest`, four for `n_interest`. Nevertheless, the induced rules showed interesting results (see figure 2).

A first conclusion shows that for increasing number of interest topics (as simulated by growing number of centroids for the feedback function), coverage decreases since compressing rules need to be more precise—thus generalizing too carefully. The bad values for `n_interest` are due to our simulation of negative feedback (see conclusion).

Improvements. Most of the rules that were not taken into account in the last section are rules which yield no compression, but nevertheless carry valuable information. For example, user 93's interest in a certain URL yielded a most specific clause as shown in figure 3. Such clauses could easily be generalized

```
p_interest_93(A,B) :-
   type__top_publication_publishedbook(A,C),
   cat__top_science(A,D),
   cat__top_science_[cs](A,E),
   cat__top_science_[cs]_[ai]_machine_learning_learning_theory(A,F),
   cat__top_science_[cs]_[ai]_machine_learning(A,G),
   cat__top_science_[cs]_programming(A,H),
   cat__top_science_[cs]_programming_languages_functional_lisp(A,I),
   cat__top_science_[cs]_programming_languages_functional(A,J),
   cat__top_science_[cs]_programming_languages_procedural_perl(A,K),
   cat__top_science_[cs]_programming_languages_procedural(A,L),
   cat__top_science_[cs]_programming_languages(A,M),
   cat__top_science_[cs]_[ai](A,N),
   C>67, F>71, G>46, I>7, K>75, L>50, N>21.
```

Fig. 3. A non–compressing rule

by a information gain guided literal dropping method (as a kind of inverted Foil method, see [16]). The rule set S generated by Progol can be roughly divided into two sets of compressing rules C and redundant rules R.[8] Thus, for each rule $r \in R$ we recursively drop least informative pairs of literals $\langle l(\_, c_l), \mathtt{thresh}(c_l, \vartheta_l)\rangle$ yielding more general rules $r' \in R'$.[9] Since coverage increases with each step, the process is stopped if the information content of the whole rule r' drops below a predefined value. In a second step, we delete rules from R' until $acc(R' \cup C)$ reaches a lower bound and output $H = R' \cup C$ as a final hypothesis.

4 Content Based Filtering with Logic Programs

In traditional meta search engines, results are aggregated and ordered using an arithmetic measure that integrates over result ranks as delivered by the utilized search engines. OySTER additionally offers conceptual ordering—where the results are ordered by document categories. Furthermore, the use of user models allows for an individual content based filtering of search results.

[8] Rules in R are called redundant since they have the same expressive power as the fact they were generated by. But since the encoding length of the rule is much greater than the length of the example, they are discarded. Thus, R is replaced by the examples E and the output hypothesis $H = C \cup E$ is of less complexity than $\mathbf{s}$.

[9] A similar technique will help in identifying aspects: sudden leaps in decreasing information gain while literal dropping suggest a border crossing.

Proving Relevance of Web Documents. Given a user model M_u, relevance actually can be proven: If there is a subset $P_u^+ \subseteq M_u^+$ such that $P_u^+ \vdash_{\mathsf{SLD}} \texttt{p\_interest}(d)$, u is interested in d according to the user model. The same holds for disinterest and a program $P_u^- \subseteq M_u^-$.
Taking aspects into account again, documents can be of different levels of interestingness, too: Given a document d, for which $P_u^+(a) \vdash_{\mathsf{SLD}} \texttt{p\_interest}(a, d)$, we have shown, that d is relevant to u with respect to a. If, however, the proof fails, and there is different aspect a', for which $\texttt{p\_interest}(a', d)$, d is still of some interest. The notion of "some" can be quantified by trying to classify the search query q and computing $\delta(\mathcal{C}(q), \mathcal{C}(d))$. Finally, if there is no a, such that relevance of d can be proven, it is likely to say that d is not interesting. However, d definitely is not interesting if there is some aspect $\overline{a}$ for which $M_u^-(\overline{a}) \models \mathcal{C}(d)$. Furthermore, any successful proof of $P^+ \subseteq M_u^+$ or $P(a)^+ \subseteq M_u^+$ (of which there might several) has a certain length. The minimum number of resolution steps used for a proof thus can be interpreted as a quality measurement, which can be improved by δ–weighed resolution steps in literal proofs.

5 Conclusion & Prospects

Results as described in this paper are based on a very pessimistic simulation of user feedback. The pessimistic approach is realized by growing numbers of centroids (thus simulating different aspects) and by defining negative centroids through δ distances to positive centroids (which explains bad results for `n_interest` in table 1). Further evaluation will show, whether accuracy increases with a more optimistic simulation of feedback. Noisy data about negative interest on the other hand, corresponds to the general user's behavior of giving only sparse negative feedback. Finally, pessimistic simulations suggest a better performance of the system using real world data.

Current work on user model induction. As already pointed out, we will enhance the quality of user models by taking into account redundant clauses and applying the literal dropping method to search for better compressing rules. The user model induction component will be completed by a module for user aspect detection which will further improve the accuracy of the user models (see footnote 9). Final results are expected by March 2001.

Further development of OySTER. The traditional meta search functionality of the search engine will be soon enhanced by a query refinement procedure which will use additional search terms that are derived from the user models. Concerning the user model induction process, we will have to automate the process and include the filtering process based upon detached Prolog proofs into the search engine interface. Secondly, we need more empirical data on real users for reliable statistics about whether the theoretical improvement of search results actually corresponds to a better performance from the user's point of view. Finally, we will integrate the Bikini wrapper inducing component, as e.g. described in [8] and, of course, we need to redesign the user feedback functionality.

References

1. R. Armstrong, D. Freitag, T. Joachims, and T. Mitchell. Webwatcher: A learning apprentice for the world wide web. In *Working Notes of the AAAI Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments*, pages 6–12, Stanford University, 1995. AAAI Press. To order a copy, contact sss@aaai.org.
2. Thomas Braun, Stefan Eilert, Axel Dörfler, Ralph Haase, Anja Kleber, Ingo Lahrkamp, Christoph Reiter, Rüdiger Rolf, Frank Trenkamp, and Frank Sievertsen. Abschlussbericht des Studentenprojektes Bikini. Technical report, Institute for Semantic Information Processing, University of Osnabrück, 2000. In Preparation.
3. M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. In *Proc. of 15th Nat. Conf. on Artificial Intelligence (AAAI-98)*. AAAI Press, 1998.
4. Dieter Fensel, Michael Erdmann, and Rudi Studer. Ontobroker: The very high idea. In *Proceedings of the 11th International Flairs Conference (FLAIRS-98)*, 1998. Available from `ftp://ftp.aifb.uni-karlsruhe.de/pub/mike/dfe/paper/flairs98.ps`.
5. Christian Heißing. Klassifizieren von URLs durch Generalisierung von Pfadnamen. Master's thesis, Institute for Semantic Information Processing, University of Osnabrück, 2000.
6. T. Joachims, D. Freitag, and T. Mitchell. Webwatcher: A tour guide for the world wide web. Technical report, CMU, September 1996. Available from `http://www.cs.cmu.edu/afs/cs/project/theo-6/web-agent/www/`.
7. T. Joachims, D. Freitag, and T. Mitchell. Webwatcher: A tour guide for the world wide web. In *Proceedings of IJCAI 97*, August 1997.
8. N. Kushmerick and R. B. Doorenbos. Wrapper induction for information extraction. In *IJCAI 97*, 1997.
9. Dunja Mladenic. *Machine Learning on non-homogeneous, distributed text data.* PhD thesis, University of Ljubljana, Slovenia, 1998.
10. S. Muggleton. Inverse entailment and Progol. *New Generation Computing*, (13), 1995.
11. S. Muggleton. Learning from positive data. In *Proceedings of the Sixth International Workshop on Inductive Logic Progrramming.* Springer, 1997.
12. M. E. Müller. Inductive Logic Programming for Learning User Models. In *Proc. FGML-2000.* GI, Fachgruppe 1.1.3 "Maschinelles Lernen", GMD, 2000.
13. M. E. Müller. Learning comprehensible conceptual user models for user adaptive meta web search. In *AAAI Spring Symposium on Adaptive User Interfaces*, Stanford, CA, 2000.
14. M. E. Müller. OySTER: Web Search as a playground for User Modeling techniques. In M. E. Müller, editor, *Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen.* Institut für Semantische Informationsverarbeitung, Universität Osnabrück, 2000.
15. M. Pazzani, J. Muramatsu, and D. Billsus. Syskill & Webert: Identifying interesting Web Sites. In *Proc. of the National Conference on Artificial Intelligence*, Portland, OR, 1996.
16. J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3):239 – 266, 1990.