FINAL REPORT

STRUCTURE BASED IDENTIFICATION OF XML DOCUMENTS

ANDRÁS LŐRINCZ

CONTRACT ORDER NUMBER:

F61775-00-WE065

Date: September 4, 2001.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (FROM - TO)
21-09-2001	Final	29-08-2000 to 29-08-2001

4. TITLE AND SUBTITLE
Structure Based Identification of XML Documents
Unclassified

5a. CONTRACT NUMBER
F61775-00-WE065

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Lorincz, Andras ;

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME AND ADDRESS
Pazmany-Eotvos Foundation of Natural Sciences and Informatics
Pazmany Peter setany 1/A
Budapest, HungaryH-1117

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME AND ADDRESS
EOARD
PSC 802 BOX 14
FPO, 09499-0014

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APUBLIC RELEASE
,

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report results from a contract tasking Pazmany-Eotvos Foundation of Natural Sciences and Informatics as follows: The contractor shall have the following topics investigated under the supervision of Professor Andras Lorincz. Application of the eXtensible Markup Language (XML) to building large-scale information resources out of millions of interrelated documents will be investigated. Potential advantages of XML for improvement of document processing by separating presentation from content and by revealing semantic structure of the documents will be exploited.

15. SUBJECT TERMS
EOARD; Agent Based Systems; Data Mining

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	Public Release	134	Fenster, Lynn lfenster@dtic.mil
Unclassified	Unclassified	Unclassified			19b. TELEPHONE NUMBER International Area Code Area Code Telephone Number 703767-9007 DSN 427-9007

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
21-09-2001	Final Report	29 August 2000 - 29-Aug-01

4. TITLE AND SUBTITLE

Structure Based Identification of XML Documents

5a. CONTRACT NUMBER
F61775-00-WE065

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Andras Lorincz (Professor Habil)

5d. PROJECT NUMBER

5d. TASK NUMBER

5e. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Pazmany-Eotvos Foundation of Natural Sciences and Informatics
Pazmany Peter setany 1/A
Budapest H-1117
Hungary

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

EOARD
PSC 802 BOX 14
FPO 09499-0014

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
SPC 00-4065

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report results from a contract tasking Pazmany-Eotvos Foundation of Natural Sciences and Informatics as follows: The contractor shall have the following topics investigated under the supervision of Professor Andras Lorincz. Application of the eXtensible Markup Language (XML) to building large-scale information resources out of millions of interrelated documents will be investigated. Potential advantages of XML for improvement of document processing by separating presentation from content and by revealing semantic structure of the documents will be exploited.

15. SUBJECT TERMS
EOARD, Agent Based Systems, Data Mining

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18, NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Christopher Reuter, Ph. D.
UNCLAS	UNCLAS	UNCLAS	UL	133	19b. TELEPHONE NUMBER *(Include area code)* +44 (0)20 7514 4474

ABSTRACT. Here, the complex problem of structure based identification of XML document is treated. This identification can have strict rules that should apply without exception. In addition, one can envision rules that are not strict, but probabilistic by nature. In fact, any complex system should be aware of the problem that error may occur and may propagate at any time instant.

The approach assumes a backing rule-based system, which can analyze the XML documents, validate and subtract error–free parts of the XML document and pass the uncertain parts for further analysis. Such uncertain parts could be, e.g., tags, attributes, and texts. The structure of the XML document may be uncertain, too. Our work concerns this last example. We assume a distribution of XML trees and perform probabilistic classification of these trees given a set of examples. We show on artificial as well as on real XML databases that efficient classification is possible for XML documents. The artificial database allows us to rigorously vary parameters of the distributions, whereas XML databases from the internet provide realistic examples. We show examples on how one can optimize the probability estimation of the correctness of a document having structure, given the database.

For the sake of completeness, the general case of novelty detection is also considered. The report is meant as a thorough, up-to-date description of recent software and AI technologies, which can be applied for the evaluation, identification and validation of XML documents. We also present results on how to speed-up the rule-based validation. Such fast methods could be applied to solve the rule-based part of the XML validation problem.

The report contains a description of the internet crawler technology that was part of the project and was used to collect documents, including XML documents over the internet.

To:

DEPARTMENT OF THE AIR FORCE

European Office of Aerospace Research and Development (EOARD)

223/231 Old Marylebone Road

London NW1 5TH

United Kingdom SPC 00-4065

Contract order number:

F61775-00-WE065

From:

dr. habil. András Lőrincz

Neural Information Processing Group

Department of Information Systems

Eötvös Loránd University

Pázmány Péter sétány 1/D

Budapest, Hungary, H-1117

Email: lorincz@inf.elte.hu

Acknowledgements

I am grateful to Doug Holzhauer, Barry McKinney, Chris Reuter and Jim Vaccaro for their time and efforts that made this research project to come to life. My group and myself have profited a lot from this interaction. Thanks!

Support of the Pázmány Eötvös Foundation is gratefully acknowledged. In particular, I am grateful to Professor Kálmán Medzihradszky, the president of the Foundation, to Professor András Benczúr, the Dean of the Faculty for their support to establish and to stabilize a new group at the Eötvös Loránd University of Budapest. I started my work from zero – no computer, no student, no start-up found – about three years ago at the University. Without their help and support, my efforts could not have been successful in such a short time. I am most grateful to Professor Péter Arató, the Dean of the Faculty of Electrical Engineering of the Technical University of Budapest. His continuous support and partnership have been crucial in each time instant. Without his generous help in critical situation, we could not have finished the EOARD project in time.

Special thanks are due to György Hévizi (PhD student), István Kókai and to Tamás Marcinkovics (MSc students) who carried out most of the computations and software developments of the project. Although

their work is not so closely related to the present project, the contribution of PhD students Barnabás Póczos, Botond Szatmáry, Gábor Szirtes and Bálint Takács and MSc student Zsolt Palotai on the CNF projects, noise filtering, novelty detection, and pattern completion was more than necessary – as it can be seen from the Appendices. Their professional contributions are excellent examples of the talents of students and the level of education in Hungary.

Contents

List of Figures

ratio:4, Depth of tree:4, 1st dataset: Dat1, 2nd dataset: Dat2

(for generative parameters of artificial datasets see Table 11.3) 45

3 **Modeling results**

For definition of sub-figures (a)-(e) see fig. 2

Model parameters: Training set size:30 samples, Branching

ratio:4, Depth of tree:4, 1st dataset: Dat2, 2nd dataset: Dat1

(for generative parameters of artificial datasets see Table 11.3) 46

4 **Modeling results**

For definition of sub-figures (a)-(e) see fig. 2

Model parameters: Training set size:30 samples, Branching

ratio:4, Depth of tree:4, 1st dataset: Dat2, 2nd dataset: Dat5

(for generative parameters of artificial datasets see Table 11.3) 47

5 **Modeling results**

For definition of sub-figures (a)-(e) see fig. 2

Model parameters: Training set size:30 samples, Branching

ratio:4, Depth of tree:4, 1st dataset: Dat3, 2nd dataset: Dat4

(for generative parameters of artificial datasets see Table 11.3) 48

6 **Modeling results**

For definition of sub-figures (a)-(e) see fig. 2

Model parameters: Training set size:30 samples, Branching

familiar and unfamiliar data sets. (B) Distributions of TICA outputs. 7 inputs were concatenated to form an embedded input. Diagonal (off-diagonal) histograms describe testing for familiar (novel) inputs. The negentropy (N) values [30] of the histograms (discretizations of the rectified distributions) are given within each subfigure. 81

18 **The case of acoustic signals.**

(A) Samples are about 250 ms in length and are from different sources (e.g. music, sounds in a forest or sounds of a whale). (B) TIC output distributions. The training set for the TICA matrix was created from a mix of three samples out of the six signals. Mixing of the other three samples made the 'novel' inputs. Embedding depth is 16. The number of TICs is thus 3x16=48. Here - in contrast to Figure 17 - the histograms of randomly selected individual outputs are shown. Diagonal (off-diagonal) blocks of histograms represent familiar (novel) tests. 82

19 **Graphical representation of the algorithm.**

(A): $\mathbf{x}, \mathbf{s}, \mathbf{h}$ and $\hat{\mathbf{x}}$: input, shrunk (denoised) ICA components, hidden variables, and reconstructed input, respectively. $\mathbf{W}$, $\mathbf{W}^+$, $\mathbf{Q}$ and $\mathbf{I}_{NMF}$ denote demixing matrix, pseudoinverse

of the demixing matrix, NMF matrix and NMF iteration, respectively. Arrow: linear transformation, arrow with black dot: linear transformation with component-wise non-linearity (shrinkage kernel), lines with two arrow-heads: iteration. The algorithm was utilized in a two phase mode (see text for details).

(B): $\hat{\mathbf{h}}$: reconstructed input of second layer (i.e., hidden variables of first layer 'according to' the second layer). Two-layer hierarchy. Lower layers provide inputs (their hidden variables $\mathbf{h}$) for higher layers. Reconstructed input of second layer overruns hidden variables at first layers. Hidden variables are subject to NMF iteration. Grey arrows represent identity transformations (I). 88

20 **Input and its reconstructed forms**

(A): perfect input without noise, (B): noise covered input, (C): reconstructed input (RI) with SCS, (D): RI with NMF, and (E): RI with combination of SCS and NMF. Signal-to-noise ratio is: 0.83. Note the improved reconstruction for the combined method compared to single SCS or single NMF algorithms. 89

21 **Single layer basis sets for SCS alone (A), NMF alone (B) and for the NMF using SCS outputs in linear mode**

(C).

Inputs have 256 (= 16 × 16) dimensions. Number of filters: 32 (= 2 × 16). For each method, all 32 filters are depicted. SNR=0.67 90

22 **Parameter dependences.**

Colors denoted different methods. SCS: cross, NMF: star, SCS and NMF: dot. (A) Dependence of RMS reconstruction error on standard deviation (STD) of the noise, (double-bar inputs) (B) RMS reconstruction error for noise-free perfect single bar inputs versus STD of noise during training phase (single-bar inputs), and (C) RMS reconstruction error for SCS (crosses) and for the combined method (dots) versus kernel width for the noise-free case (black lines) and for noisy input (grey lines). STD=1.5. Note the remarkable independece on the kernel parameter, the single adjustable parameter for a given architecture.. Kernel width for (A) and (B): 0.06 92

23 **Improved noise filtering and pattern completion in the hierarchy.**

Subarchitectures of the first layer have 6 × 6 = 36 input dimension. Dimension of NMF hidden vectors of first layer units is 12. First layer is made of 3 × 3 = 9 units. Input of

List of Tables

1. Introduction

Internet technology is developing at a very high rate. Standardization of existing methods go 'hand-in-hand' with new developments of information theory. In fact, the area makes use of the latest result of mathematics, electrical engineering on the one end, and discoveries of neurobiology, DNA computing, psychology and psychophysics, evolutionary techniques, etc., on the other. The speed of development of sofware components, techniques that allow to reuse such components in a platform free fashion is astonishing. This evolution is to become unlimited. In my opinion, past limitations imposed by 'built-in-obsolences' of Microsoft are about to disappear. Platform is about to become secondary, and we might experience fast 'globalization' much sooner than desired. The central part of this new era is JAVA and related enabling technologies, including T- and, JavaSpaces, and XML Security tools. In turn, AI tools that can deal with 'unlimited' information from the point of view of the human brain, are in need. A small part of this problem, the structure based identification of XML documents, is the subject of the present contract.

1.1. **Remote example.** I give an example, a possibly solved case, of the next three years.

Existing examples prove the need for adaptive asynchronous information collection and control methods for distributed decision-making. Several experiences (see, e.g.,

$http://www.eds.com/case_studies/case_arkansas.shtml$) point towards this direction. My model example is the complex case of Parkinson disease patients (PDPs) using Deep Brain Stimulators (DBS). Simpler systems can be derived as analogies of this case.

For PDPs, histories of individual patients show that sporadic experiences at different hospitals need to be collected to improve evaluation. Remote setting of the DBS, augmented reality with information collection and data management together, could form global ambient intelligence (GAI) for full support of the patients. Data management should easily meet constraints derived from Good Clinical Practice and could be seen as an extension of drug administration procedures. GAI can serve the patient, can help the doctor and can estimate risk. Adaptive tools for such (safety critical) applications have become available by the latest achievements in reinforcement learning (RL). Optimization of decisions on continuing data collection or executing an action can be termed as optimization of perception-action loops. Such challenging optimizations concerns partially observed environments, being in the focus RL research.

One may foresee the birth of a 'new generation', the 'AI nurse race', or the 'hostess' race 'who' represent goal-oriented agents with possibly adaptive and distributed subsystems. The nurse 'serves' someone, e.g., a patient, a visitor, the owner of a home, or groups of those. For the interactions with people served, the hostess makes use of signals that are important for efficient human communication, including facial expressions, prosody, body talk, and behavioral patterns. In case of PDPs GAI needs to exploit: - the medical history of the patient (available locally or over the internet), - the medical history of other patients, - mobile communication outside of safe environments (PDPs with DBS often engage in activities outdoor), - optimization of the neural prosthetic device, the DBS, with option for remote control, - visual, acoustic and haptic augmented reality tools, - distributed data editing, - (provisional) access control, - decision making in this safety critical situations. Particular aspect of the project is that decision is achieved by combining the assessments of experts who may be located at different sites and may have different roles. The access to (medical) data may improve the decisions at the expense of real time, communication time, and computational costs. Risk-sensitive decision-making needs to consider pipeline operations, both in communication and in

computation tasks. One can develop components under the assumption that Quality of Service will be available for internet networks in about two years.

I see no bottleneck for GAI. Today, all essential technology components are within reach, including (i) JavaSpaces, the flexible *message-ware* between available/existing software components at various sites and (ii) XML access control (XACL) for *provisional authorization* and (iii) reconfigurable hardware components.

2. Overview of report

The report is meant to be a thorough overview of technologies that I consider important. The report has the following structure. First (in Section 3), the logistics of 'measuring' XML documents is described. Subsequent sections treat the recent SW components and (a restricted subset of) state-of-the-art AI techniques. These sections review describe the technologies that we have applied. The AI techniques that I describe here, are mostly restricted to novelty detection. Detailed descriptions, including mathematical details, can be found in the Appendices. Results on probabilistic clustering are presented in Section 11. This section is followed by a short discussion and conclusions, which basically say 'yes, this is powerful', and a section (12.1) which

is about possible future directions. The report ends with Appendices and a very short (!) list of references, of cca. 100 key papers and URL addresses. The last Appendix (Appendix 17) contains the list of software attached on the CD ROM. Software is given in source code. Components are written in Matlab and in JAVA. In either cases the appropriate commenting standards are used: Javadoc for JAVA and 'help' system in the header of each file for Matlab.

3. Logistics of XML validation, identification, and probabilistic clustering

Examination of XML documents can have a series of steps. A possible scenario is shown in Fig. 1. Other scenarios may well compete with the procedure depicted. In particular, examination loops needs to be inserted at almost every item.

The basis of the procedure is the following.

(1) Does the document satisfy those strict rules (or checklist) that we have? For example, is it an XML document?
(2) If no, and this the case of interest here, could it be a valid, or a partially useful document or is it 'junk', or could it be 'potentially dangereous', etc?

(3) Partial information can be collected and evaluated together at a final stage, that could be a 'mixture of experts' model, or a 'product of experts', or similar probabilistic AI technologies.

In a more general setting, one needs to talk about 'diagnosis': Any partial information is a symptom and one should look for the probabilities of different diseases. Loops are important in such inferencing, because the 'disease' can be recognized at an early stage or may call for more thorough analysis. Recent AI achievements enable expert level variational Bayesian reasoning using such data from the Quick Medical Reference Database (QMR-DT), which corresponds to a search space of 2^{600}. This astonishing number indicates that the technology will be widely used in diverse fields.

4. Review of a few useful XML tools

There is a tremendous number of SW tools for XML documents. The proliferation of XML based markup languages is still vivid and increasing. We list three of these tools. These tools are not the most important tools, however, these tools were necessary for us during the project. For other general tools, the interested reader is referred to the literature. Best starting page is possibly http://www.xml.org. The

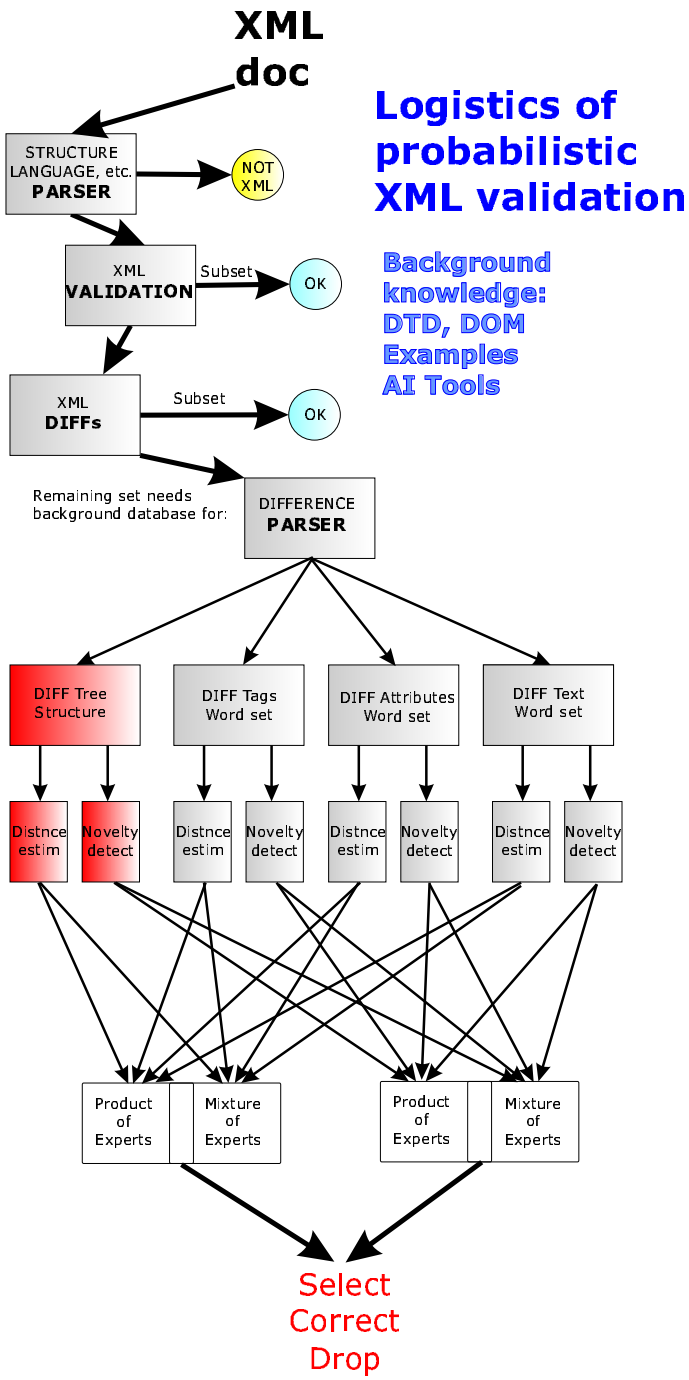

FIGURE 1. **Logistics of XML identification**
Every rule-based system, which is of help, should be applied. Cross-validated parts of the XML documents should be eliminated. The remaining part of the XML document is the 'difference'. Differences may appear in tags, in attributes, in the text itself, and in the tree structure. All of these 'differences' should be analyzed to estimate its the probabilities. Outputs of these analyzers may be considered as 'expert opinions' and may serve a further stage that make a decision about the document. This stage could apply – beyond others – mixture of expert and/or product of experts strategy for estimation.

first two tools (Tidy and Jtidy) allow to convert html documents to XML form. Some of our studied examples concerned such documents.

(1) Tidy (see Appendix 14.1)

(2) Matlab JAVA (see Appendix 14.2)

The last tool (Matlab JAVA) allowed us fast algorithmic development. Efforts made by Matlab developers to meet the Java challenge are noticable and the resulting software is useful.

5. Review of exisiting XML validation, etc. tools

There is a tremendous number of SW tools for XML, including tools (see Appendix 13) for

(1) validation,

(2) constraints,

(3) comparisons, and

(4) assertion grammars).

These tools are reviewed in the Appendix and/or are attached in the CD-ROM.

6. Used AI tools

The AI tools that we used and describe here, include

(1) Probabilistic Tree Classifier (see Appendix 15.1)

(2) Connectionist novelty detection (see Appendix 15.2)

(3) Pattern completion with probability estimation (see Appendix 15.3)

(4) Fast solution to the satisfiability problem (see Appendix 15.4)

These tools are detailed below.

6.1. **Joint distributions.** One of the challenges of density estimation as it is used in machine learning is that usually the data are multivariate and often the dimensionality is large. Examples of domains with typically high data dimensionality are pattern recognition, image processing, text classification, diagnosis systems, computational biology and genetics. Dealing with joint distributions over multivariate domains raises specific problems that are not encountered in the univariate case. Distributions over domains with more than 3 dimensions are hard to visualize and to represent intuitively. If the variables are discrete, the size of the state space grows exponentially with the number of dimensions. For continuous (and bounded) domains, the number of data points necessary to achieve a certain density of points per unit volume also increases exponentially in the number of dimensions. One other way of seeing this is that if the radius of the neighborhood around a data point is kept fixed while the number of dimensions, in the forthcoming denoted by n, is increasing the relative volume of that neighborhood is exponentially decreasing. This constitutes a problem

for non parametric models. While the possibilities of gathering data are usually limited by physical constraints, the increase in the number of variables leads, in the case of a parametric model class, to an increase in the number of parameters of the model and consequently to the phenomenon of overfitting. Moreover, the increased dimensionality of the parameter space may lead to an exponential increase in the computational demands for finding an optimal set of parameters. This ensemble of diffculties related to modeling multivariate data is known as the curse of dimensionality. Graphical models are models of joint densities that, without attempting to eliminate the curse of dimensionality, limit its effects to the strictly necessary. They do so by taking advantage of the independences existing between (subsets of) variables in the domain. In the cases when the dependencies are sparse (in a way that will be formalized later on) and their pattern is known, graphical models allow for effcient inference algorithms. In these cases, as a sideresult, the graphical representation is intuitive and easy to visualize and to manage by humans as well.

6.2. **Representation capabilities.** If graphical representations are easy to grasp by means of human intuition, then the subclass of tree graphical models will be even more intuitive. For once, they are sparse

graphs, having n-1 or fewer edges. More importantly and more precisely, between each two variables, there is at most one path, or, in other words, the separation relationships, which are not easy to read out in a general Bayes net topology, are obvious in a tree. Thus, in a tree, an edge corresponds to the simplest common sense notion of direct dependency and is the natural representation for it. However, the very simplicity that makes tree models intuitively appealing also limits their modeling power. In other words, the class of dependency structures representable by trees is a relatively small one. For instance, over a domain of dimension n there are n^{n-2} distinct (undirected) spanning trees, but a total of $2^{n*n-1)/2}$ undirected graphs.

6.3. **Probabilistic trees.**

7. Probabilistic Trees

As discussed above we apply probabilistic tree model in our methods. The model is described here.

7.1. **Probabilistic tree classifier.**

Consider n data set given: $D_1 \ldots D_n$. The elements of each data set belong to a particular probability distribution. We assign a probabilistic tree model to each data set. The

$T_1 \dots T_n$ probability distribution learned by the probabilistic tree models minimizes the Kullback-Leibner (KL) divergence between the distribution of the data set and the distribution represented by the tree models.

After training the models, the system is able to classify efficiently.

When the data originates from an unknown source, we present the data to the $T_1 \dots T_n$ models. Then the source of the data can be identified as the class of the model giving the greatest probability (probabilistic 'distance', 'membership') for the unknown source.

7.2. **Learning a tree model.** Tree models can be learned by the Chow-Liu algorithm [28]. Details can be found in 15.1

8. Mixture of Tree model

The Mixture of Tree model [83] is a method which can be considered as the generalization of the Probabilistic Tree model presented earlier in the text.

The advantage of the method is that it can also be applied in the case when the categories used in chapter 7.1 are unknown. If the data set is from many sources, the Mixture of Trees method is capable of clustering the data set. The mixture model created this way minimizes

the KL divergence between the distribution of the input data set and the distribution represented by the model.

The disadvantage of the method is that depending on the task, the order of magnitude of its time complexity is twice as great as the time needed to train the simple Probabilistic Tree model. Its applicability is limited by the fact that the clusters it consists of have to be known. When the classes are from similar distributions, the clustering doesn't correspond to the input data set. There are some cases when the method gives good KL divergence results with different clustering.

When data from identified sources are not available, the Probabilistic Tree models can not be trained separately. Under such circumstances one may apply the Mixture of Trees method for probabilistic validation.

9. Further state-of-the-art AI tools, which are required

The breakthrough of probabilistic models originates from the recent achievements on

(1) probabilistic diagnostic tools (see previous note on QMR-DT), which are closely related to

(2) probabilistic pattern completion techniques, which however, do not require known (measured) probabilities but, instead make use of noise models and can discover sub-components of the data

(3) database-dependent adaptive noise filtering methods that extract structureless unmodeled components (possibly having maximal entropy content).

The strength of these methods is so attractive, that recently traditional AI fields are making use of them. Our example is about solving CNFs with function approximators (subsection 9.3) Some details can be found below, more detailed descriptions are given in the Appendices and in the cited literature.

9.1. **Novelty detection.** Recent advents on adaptive information maximization can discover structured components (generalized 'directions') of databases. Such directions form a subspace in the database. In another context, one might say that compression methods could be database dependent. These statements are in line with recent theoretical unification of prediction, gambling, and coding [40, 92, 120].

Information maximization provides us the tool to extract structureless components. The method can work with a (possibly overcomple) quasi–linear thresholded transformation, called sparse code shrinkage

(SCS) [54]. SCS, which is a generalization of wavelet denoising technique has the advantage that the part of the data, which is worth to analyze (i.e., has structure in it), and the part of the data, which is not worth to analyze (i.e., has no structure) can be separated given a database. Making use of the database dependent structured 'directions', novelty would classify as noise. In turn, it is rejected by the quasi–linear filter and can undergo structure analysis. Structure analysis, in this context means tools for maximization of information transfer. Maximization of information transfer for a single input is related the Kolmogorov-complexity of the input. Adaptive tools aiming to solve this problem are subject to extensive research at present. In a broader context, structure analysis can mean a joint analysis using 'collaborating' rule based and statistics based 'experts'. (See Appendix 15.2)

9.2. **Pattern completion.** One may assume that the underlying noise of the external world is Poisson noise. This assumption means that a particular component of the input is either present or not. Discovering the best representation under this assumption is equivalent to discovering the sub-components of the input. This approach has been under investigation. Sometimes it is called 'positive coding' [26] and also 'non-negative matrix factorization' [77]. These methods are related in

their assumptions about the 'world', however, they are very different from the algorithmic point of view. For more details see Appendix 15.3

9.3. **Solving CNFs with function approximators.** A serious problem of computing the 'differences' between an actual input and the database is that large satisfibility problems need to be solved. State-of-the-art methods make use of function approximators to speed up the search. Recent techniques make use of reinforcement learning or a fast approximation to reinforcement learning [20, 22]. The technique is called STAGE. This direction is important because of the tremendous computational requirements posed by the comparisons.

In another project [121, 86], we have adapted STAGE and experienced its attractive properties. Results are summarized in Appendix 15.4.

10. Problem definition: Probabilistic model for XML files

The first task of creating the probabilistic model is to define probabilistic variables. We have created a class definition for trees that covers any tree. It is like numbers $1, 2$ and *many* covers every natural number. This class has been called 'covering convex hull tree' (CCH tree). It allowed us to assign probabilistic variables to tags of XML

documents. The probabilistic variables are assigned to the vertices of the tree (the variables are indexed according to breadth-first traversal: $X_1 \ldots X_n$). A complete tree was generate with a given depth and branching ratio.

The tree structure of an XML document is naturally given by the hierarchical structure of its elements. The XML tree is fit onto the CCH tree of the probabilistic variables in the following fashion. The X_1 probabilistic variable is assigned to the root element, and the other variables are assigned to tags/elements in the corresponding positions.

The probabilistic variable can take the following values:

- 0, meaning "Not exists", if there is no tag in the XML tree in the same position as a probabilistic variable,
- $nrAtts + 1$, where $nrAtts$ denotes the number of attributes of the element in the position of the probabilistic variable.

The depth or branching ratio of the XML tree could be greater than the tree of probabilistic variables. In this case, there are tags in positions which don't have corresponding variables. These tags are not represented in the model. The number of these ignored tags can be decreased by increasing the number of probabilistic variables. The size of the model is limited by the available computational capacity. As seen from our experimental results (see Table 11.3) , the error can

be decreased by choosing the depth and branching ratio of the model appropriately. The ideal settings are task-dependent.

10.1. **Generating Sample XML Files.** The XML generation is done by a Java class, TreeBuilder. This class handles the generation of trees, and is highly customizable. The main idea was to generate one tree according to some parameters, and permute it several times to create a class of documents. Parameters of tree generation are:

- depth of tree
- upper bound of branching ratio (number of children of an element) is generated as a random number with this upper bound at each element
- probability distribution of elements
- probability distribution of attributes
- or an XML document can also serve as a basic tree

The probability distributions are encapsulated by the Probability class. So far uniform and Gaussian probability distributions have been implemented. At each element, the element distribution is used to generate the number of children of the node, while the number of attributes are generated according to the attribute distribution parameter. The upper bound for the attribute number can also be set.

We created two basic tree operations to permute a given XML document. These are *deletion* and *adoption*. The delete operation deletes a subtree from the XML tree. In the adoption method, one children (if exists) of an element is adopted by the element's next sibling. Both operations are performed recursively at each element in the tree. These are also customizable: the user of class treeBuilder can use a Probability object to set the probability distribution of deletion and adoption, and the increase (decrease) by which the probabilities of these operations change at each level from the root.

The deletion permutation makes sense because it reflects that in the content model of an element, some child elements can be optional (sometimes are present, sometimes not). By using deletion, we can simulate optional elements. Similarly, adoption is useful to reflect that an XML was restructured (an element has a new position, for example when a database structure changes, or the DTD/Schema changes).

If many files will be generated with these operations from one tree, they will follow a particular probability distribution, and can be perceived as a class of documents.

In real-world applications, many variations of one element can exist when its content model is not strict. In this case, the sequence of its child elements is not constrained, for example it is specified as a choice

or set of (by <xsd:choice> and <xsd:all> elements in XML Schema). It would be difficult and require too many samples to simulate. Therefore we transform all of the trees in our model to be *leftish* (the children of every node are ordered so that the one the depth of which is greater is on the left of the other).

10.2. **Representing XML Files.** The XML files are represented as an array which will be used to train the probabilistic variables. This is done by the XML2Tree class (or can be invoked as a TreeBuilder method also). Only a subtree of the XML is represented. The depth and branching ratio of the subtree can be set. This subtree is a complete tree, and can be deeper or wider than the xml document. Then, using breadth-first traversal, an array representation of this subtree is created, where the elements (and attributes) of the document are represented by numbers.

- If there is an element (with possible attributes) in the document tree in the which corresponds to a node in the represented subtree the element will be represented at the node's position by $nrAtts + 1\text{m}$ where $nrAtts$ is the number of its attributes.
- if the node in the represented subtree doesn't have a corresponding element in the tree of the XML document, it will be represented by zero.

11. Results

11.1. **Identifying probabilities.** We applied the following method to identify data sets:

A data set from a specified probability distribution is separated into training and control set. Then a Probabilistic Tree model is trained according to the training set. After this, the model gave probabilities for samples from both the training and control set. The histogram of probability distributions can be seen e.g. in Fig.2.(a) and 2.(b) According to our experiments, a small number of training data is sufficient for the model to give similar distribution for the training and control set.

The range of the obtained probability values cover more orders of magnitude. However, on logarithmic scale, the histogram corresponding to the data set looks smooth.

To check what probabilities are assigned to data sets from different distributions, we extended the method above:

Samples from another data set were presented to the model. The histogram of the results obtained this way is in Fig. 2.(c). It can be seen that there is only a slight overlap between the histograms of the two data sets.

To check that the second data set is from a well-defined probability which can be learned, we trained a second Probabilistic Tree model in

this data set also. The histograms about this model's results with the training and control sets of second data set are shown in Fig.2.(d) and Fig.2.(e).

11.2. **Classification.** The classification experiments were performed on a data set from three sources. Training sets with identical cardinality were chosen from the three data sets. Then, three Probabilistic Tree models were trained on these sets respectively. The three model assigned probability values to the remaining elements of the three data sets. We defined the class of the data set as the class of the model which assigned the greatest probability estimate to the data set. During the experiment, we changed the data sets used, the number of elements in the training sets, and the parameters of the covering trees. For the summary of results, see Table 11.3.

11.3. **Novelty detection.** Novelty detection is more difficult than classification, since it's not about choosing the greatest one from many probabilities. Here, one has to make a decision based on a probability value estimated by a model, whether the given sample belongs to a set or not. Determining the threshold value is a difficult task, becauseL

- as can be seen in the figures of this section, the model gives an answer which spans more orders of magnitude
- the number of identified correct samples has to be maximized
- while the number of misclassified samples has to be minimized

When the threshold value is small, we not only reject the samples that are out of the category, but also fail to identify the data belonging to the class. If the threshold value is too high, we accept both the data belonging to the class and those who don't with high probability. See e.g. Fig.(10-16)

Our method to determine the threshold value takes both source of errors into consideration. We choose the threshold value so that the following sum would be minimized: (number of samples belonging to rejected classes + number of wrongly accepted samples from other classes).

We summarized the results obtained with this method in table 11.3.

12. Discussion and conclusions

My view of the validation identification problem is as follows. The final word on validation–identification is given by the complexity of the task. For small problems

- rules can be generated,

	Dat1	Dat2	Dat3	Dat4	Dat5
Size of the dataset	1001	1001	2001	257	431
Sizes of XML's tree structure					
upper bound of branching number	4	4	5	5	5
upper bound of depth	4	4	4	4	5
generated depth for the first sample file	4	4	5	5	5
Probabilities					
Node Probability Data					
name of distribution	U	U	U	U	U
delete probability	0.1	0.1	0.1	0.2	0.1
adopt probability	0.2	0.2	0.1	0.2	0.1
increase to delete probability	0.05	0.05	0.05	0	0.03
increase to adopt probability	0.05	0.05	0.05	0	0.03
Attribute Probability Data					
name of distribution	G	G	G	G	G
medium	4	2	2	3	2
standard deviation	1	1	1	1.5	0.5
probability of updating attributes	0.3	0.3	0.3	0.3	0.3
maximum generated attribute number	6	6	3	5	6

TABLE 1. **Parameters of artificial datasets generated by TreeBuilder software.**

1st Dataset	2nd Dataset	3rd Dataset	B.R.	Depth	N.T.S.	S.N.	F.S.	P.
Dat1	Dat2	Dat3	3	4	10	3973	775	0.804
Dat1	Dat2	Dat3	3	4	30	3913	225	0.942
Dat1	Dat2	Dat3	3	4	100	3703	65	0.982
Dat1	Dat2	Dat3	2	5	30	3913	161	0.959
Dat1	Dat4	Dat5	3	4	30	1599	250	0.843
Dat2	Dat3	Dat5	3	4	30	3343	90	0.973
Dat4	Dat3	Dat2	4	2	30	3169	64	0.980
Dat4	Dat3	Dat2	4	3	30	3169	85	0.973
ADC	CML	Dat1	3	4	30	3341	162	0.952
ADC	CML	Dat1	4	3	30	3341	25	0.993

Abbrev	
B.R.	Branching ratio of the covering tree.
Depth	Depth of the covering tree.
N.T.S.	Number of training samples
S.N.	Sample number. Overall number of samples in the three datasets
Thrs	Threshold. Optimal threshold of probabilities for accepting/rejecting samples.
F.S.	Failed samples. The number of misclassified samples.
ADC	Astronomy Data Center.
CML	Chemical Markup Language.

TABLE 2. **Parameters used for artificial (Dat1-Dat4) and real XML data (ADC, CML).**

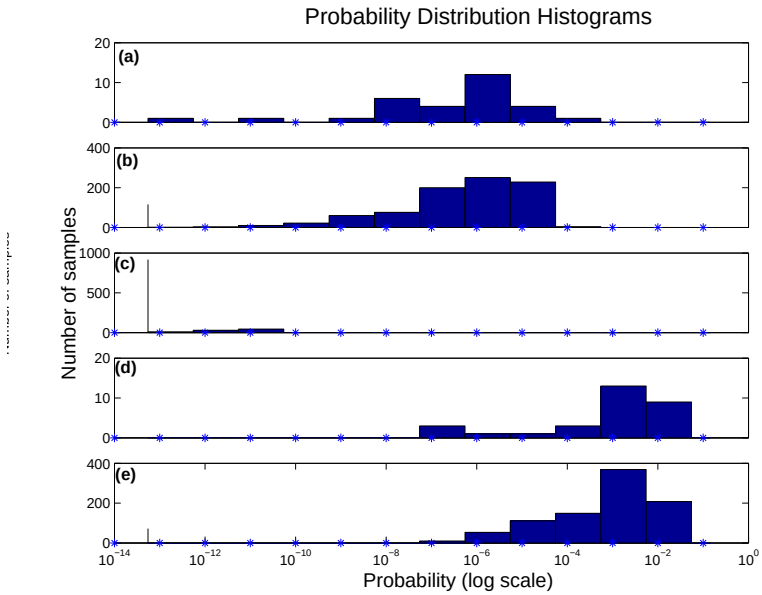

FIGURE 2. **Modeling results**
The figures show the probabilities approximated by Probabilistic Tree Models
(a) Model-1 on its training samples from 1st dataset
(b) Model-1 on unseen samples from the same dataset
(c) Model-1 on 2nd dataset generated from different distribution
(d) Model-2 on its training samples from 2nd dataset
(e) Model-2 on unseen samples from 2nd dataset
Model parameters: Training set size:30 samples, Branching ratio:4, Depth of tree:4, 1st dataset: Dat1, 2nd dataset: Dat2 (for generative parameters of artificial datasets see Table 11.3)

- rules updated, and
- rules can be enforced.

This possibilities are excluded

- for problems without strict rules,
- for problems where rules may change (often) and SWs are not thoroughly tested
- for problems wich are large and can be subject to corruption

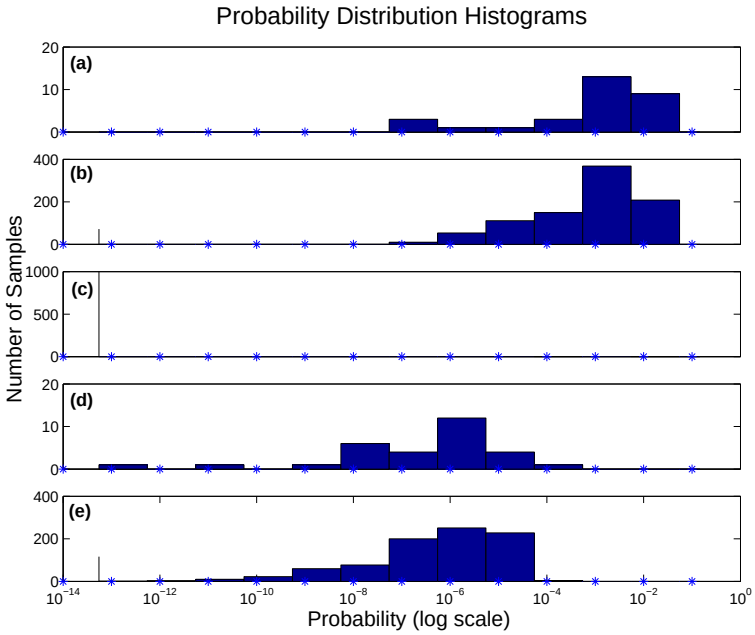

FIGURE 3. **Modeling results**
For definition of sub-figures (a)-(e) see fig. 2
Model parameters: Training set size:30 samples, Branching ratio:4, Depth of tree:4, 1st dataset: Dat2, 2nd dataset: Dat1 (for generative parameters of artificial datasets see Table 11.3)

- for problems that have 'measurements', i.e., for noisy problems

and the combinations of those. In turn, probabilistic models are required for any internet based technology, including XML.

The project has shown the applicability of data based probabilistic analysis of tree-structured XML documents. Results on artificial and internet databases have been presented. The approach is to be judged as one component of a larger validation system.

12.1. **Possible future directions.** Future steps can be directed towards SW development for an applications. This would require (a)

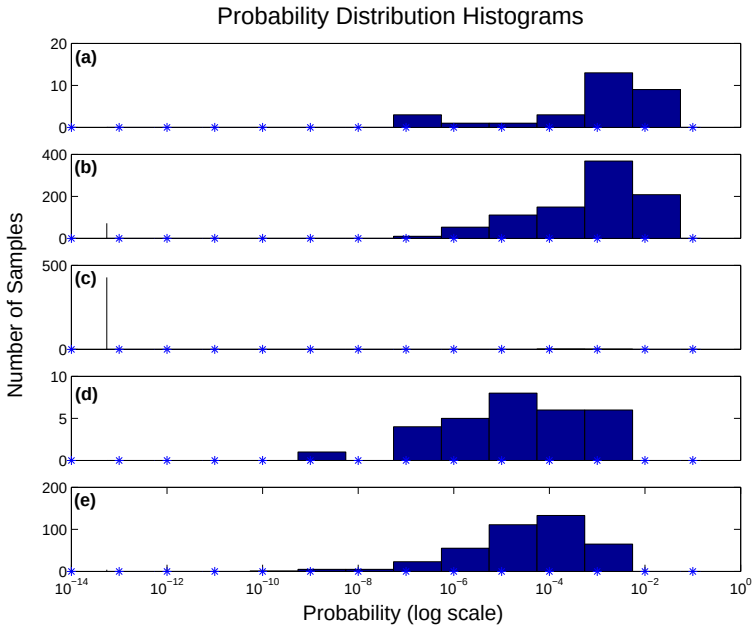

FIGURE 4. **Modeling results**
For definition of sub-figures (a)-(e) see fig. 2
Model parameters: Training set size:30 samples, Branching ratio:4, Depth of tree:4, 1st dataset: Dat2, 2nd dataset: Dat5 (for generative parameters of artificial datasets see Table 11.3)

a particular database and (b) a study of that database for efficient encoding.

Another direction of future work could be the investigatin of the efficiency of the distinct probabilistic methods. The mutual improvement of these methods and the computational overhead could be analyzed this way.

Last but not least, statistical analysis by itself has a limited value. There are too many unimportant details, which should not be analysed for a given purpose. In other words, statistical analysis needs

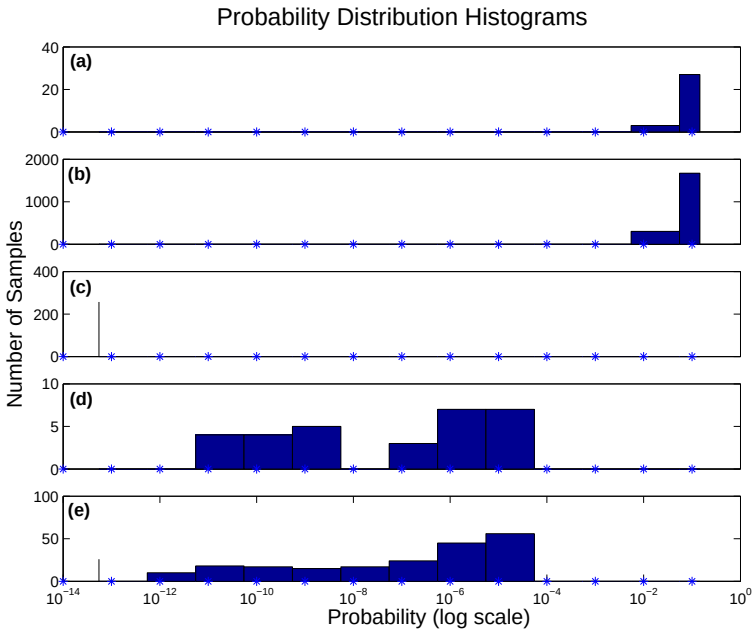

FIGURE 5. **Modeling results**
For definition of sub-figures (a)-(e) see fig. 2
Model parameters: Training set size:30 samples, Branching ratio:4, Depth of tree:4, 1st dataset: Dat3, 2nd dataset: Dat4 (for generative parameters of artificial datasets see Table 11.3)

to be extended by the goals of the system. In turn, goal-oriented statistical analysis is the direction this work should be extended to. Given the attractive properties of learning systems, adaptive statistical methods and adaptive goal-oriented systems need to be combined. Reinforcement learning (RL) seems to be the appropriate tool to do so. The major problem of RL is the exponentially growing learning time as a function of state and action spaces. The internet crawler is a novel example on how to combine RL and classification schemes.

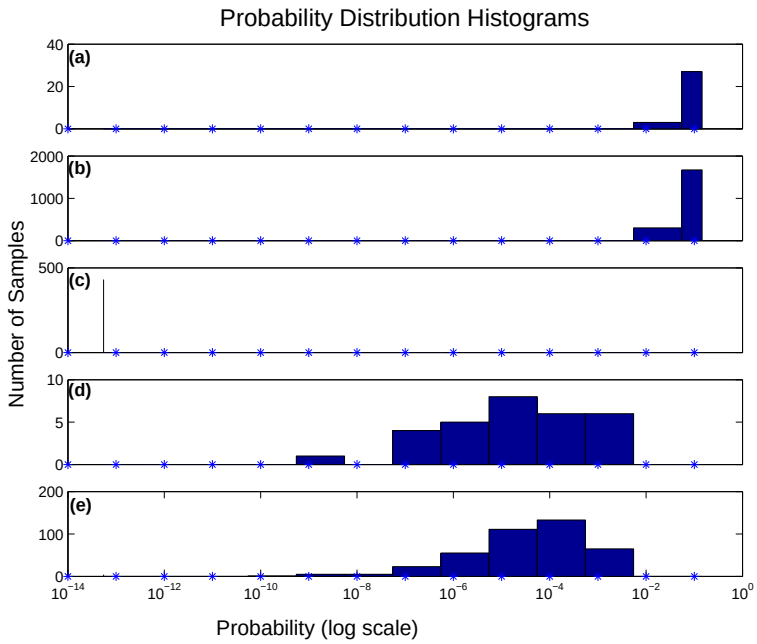

FIGURE 6. **Modeling results**
For definition of sub-figures (a)-(e) see fig. 2
Model parameters: Training set size:30 samples, Branching ratio:4, Depth of tree:4, 1st dataset: Dat3, 2nd dataset: Dat5 (for generative parameters of artificial datasets see Table 11.3)

This approach, which is very similar to the most efficient game playing machines is convincing possibility to 'combine the experts'.

Still, it is known, that feature extraction is the most stringent requirement for RL. In turn, a method that makes use of RL requires:

- automatic feature extraction (automatic extraction of sub-components) and
- problem partitioning.

The former problem may assume an efficient solution given our hierarchical ICA + NMF algorithm (Appendices 15.2, 15.3). The latter

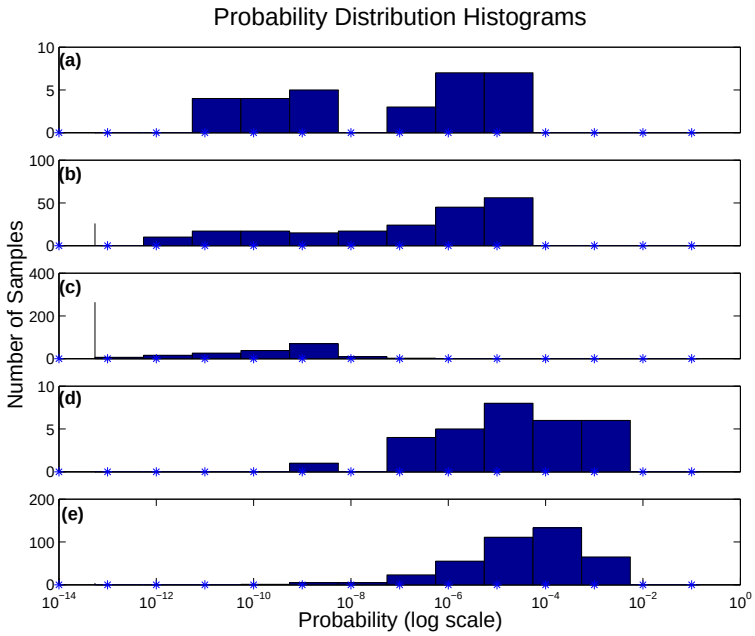

FIGURE 7. **Modeling results**
For definition of sub-figures (a)-(e) see fig. 2
Model parameters: Training set size:30 samples, Branching ratio:4, Depth of tree:4, 1st dataset: Dat4, 2nd dataset: Dat5 (for generative parameters of artificial datasets see Table 11.3)

seems to have its solution in a re-phrasing of RL in terms of the so called event-learning formalism [63, 64, 78].

APPENDICES

13. APPENDIX A: EXPRESSING CONTRAINTS ON XML DOCUMENTS

There are many tools which can be used to check the syntax and semantics of XML documents. First, well-formedness is checked by an XML parser, then additional constraints can be expressed:

- the document can be validated against different schemata or DTDs

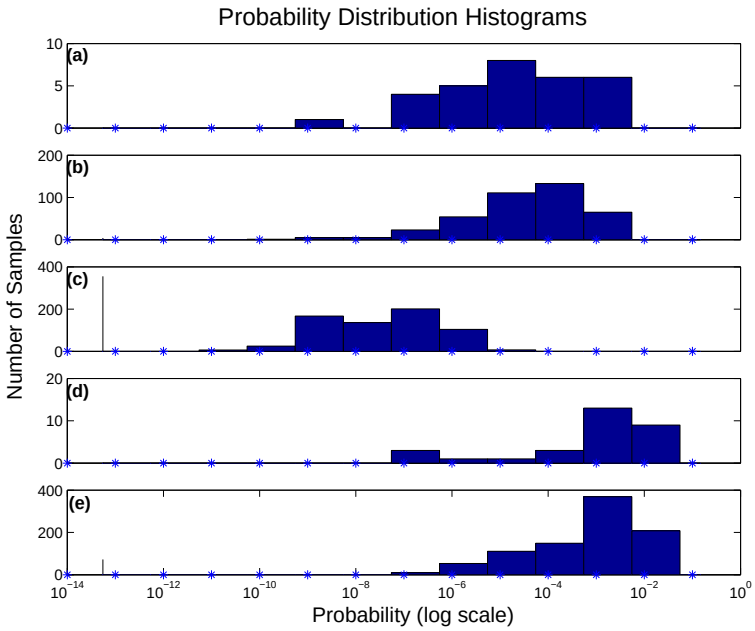

FIGURE 8. **Modeling results**
For definition of sub-figures (a)-(e) see fig. 2
Model parameters: Training set size:30 samples, Branching ratio:4, Depth of tree:4, 1st dataset: Dat5, 2nd dataset: Dat2 (for generative parameters of artificial datasets see Table 11.3)

- an XSL stylesheet can express additional assertions, or act as a schema language
- an XSL extension can support user-defined functions which can perform further checks
- an EXSLT (Extended XSLT) processor's built-in functions (mathematical, set functions) can be used
- the document can be compared to other XML documents (diffing and merging tools)

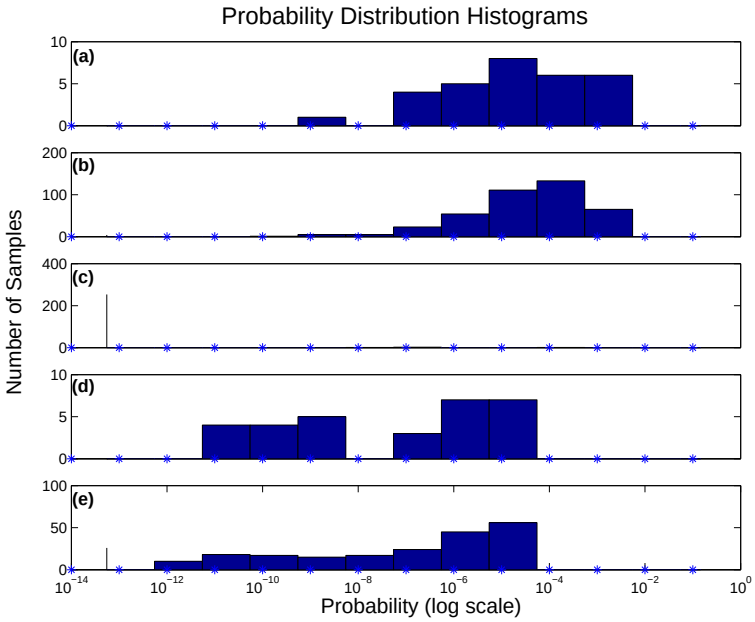

FIGURE 9. **Modeling results**
For definition of sub-figures (a)-(e) see fig. 2
Model parameters: Training set size:30 samples, Branching ratio:4, Depth of tree:4, 1st dataset: Dat5, 2nd dataset: Dat4 (for generative parameters of artificial datasets see Table 11.3)

- a query language (XQL, XQuery) can be used to look for the same information (similar elements, contents)
- an XML processor with DOMHash implementation can be used to check whether two nodes in an XML tree are the same

Validating XML documents

A schema is a collection of rules about a document's structures and syntax. An XML document is valid if it has a schema and it satisfies the constraints described in that schema. The terminology is often 'a document is valid against a schema/DTD'. There are several schema languages (XML Schema [XSD], TREX, RELAX, RELAX NG, SOX,

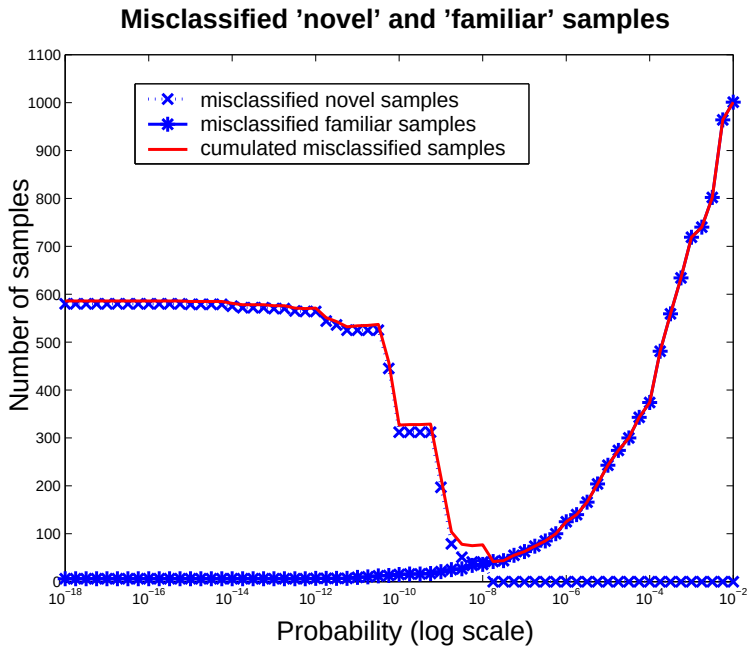

FIGURE 10. **Optimized novelty detection**
Misclassified samples vs. accepting threshold value.
Model parameters: Training set size:150 samples, Branching ratio:3, Depth of tree:4, "Familiar" dataset: Dat1, "Novelty" dataset: Dat2 (for generative parameters of artificial datasets see Table 11.3)

DSD, XDR, BizTalk schema, Schematron, etc.), and people keep on creating new ones. The most important ones are DTD, XML Schema, and Schematron.

DTD

The goal of DTDs is to specify the structure of instance documents.

- check the content model of an element
- check properties of attributes
- check some references (ID, IDREF) in a document

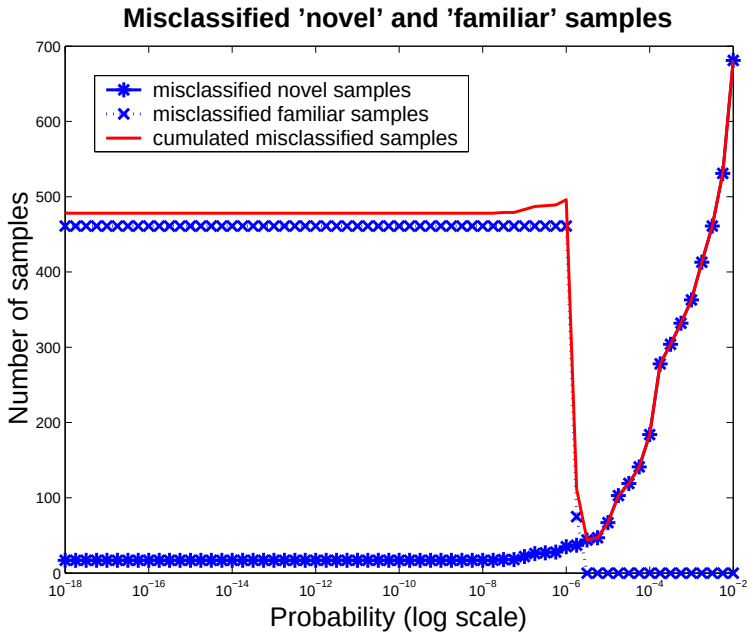

FIGURE 11. **Optimized novelty detection**
Misclassified samples vs. accepting threshold value.
Model parameters: Training set size:150 samples, Branching ratio:3, Depth of tree:4, "Familiar" dataset: Dat2, "Novelty" dataset: Dat3 (for generative parameters of artificial datasets see Table 11.3)

- insert additional information into an XML document (default attribute values)

However, DTDs are very limited, and the DTD syntax is not in XML. A DTD cannot express datatypes that are often desired in commercial environments (validating banking data, range checking, date handling, etc.), and its datatypes are not compatible with those found in databases. Only 'character data' can be specified, in different forms (#PCDATA, NMTOKEN). The other shortcomings of DTDs is that

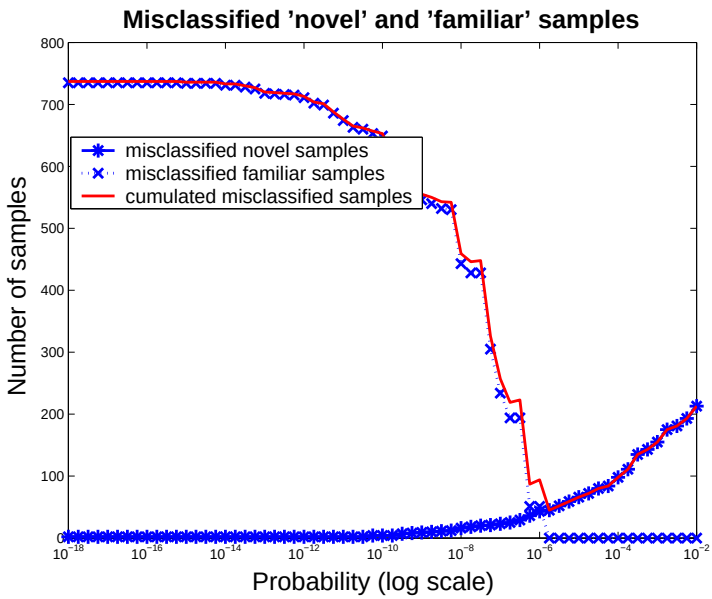

FIGURE 12. **Optimized novelty detection**
Misclassified samples vs. accepting threshold value.
Model parameters: Training set size:150 samples, Branching ratio:3, Depth of tree:4, "Familiar" dataset: Dat4, "Novelty" dataset: Dat2 (for generative parameters of artificial datasets see Table 11.3)

they are not namespace aware (so one cannot specify the same element name in different context to be different).

DTDs are still widely used, due to their simplicity and the support of many DTD-aware validating parsers.

XML Schemas

The purpose of XML Schemas [5, 6, 4, 7] are to specify the structure of instance documents and the datatype of each element/attribute.

Highlights of XML Schemas:

- XML syntax

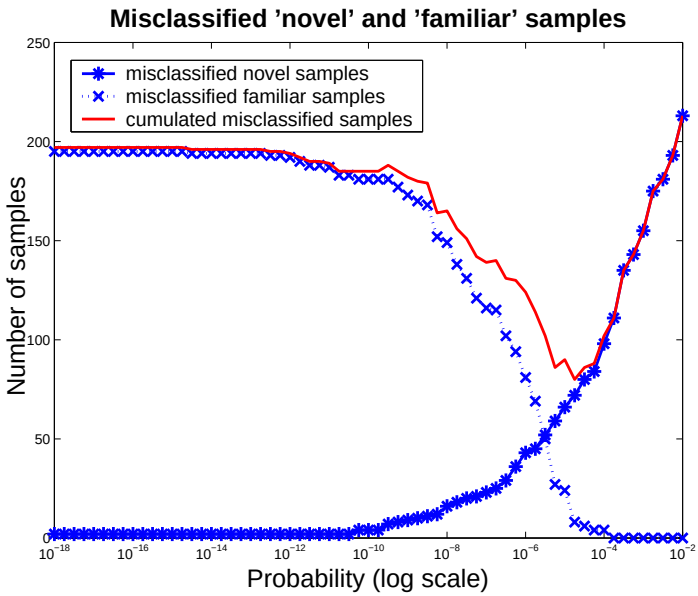

FIGURE 13. **Optimized novelty detection**
Misclassified samples vs. accepting threshold value.
Model parameters: Training set size:150 samples, Branching ratio:3, Depth of tree:4, "Familiar" dataset: Dat4, "Novelty" dataset: Dat5 (for generative parameters of artificial datasets see Table 11.3)

- rich datatyping
  - more than 44 internal datatypes
  - user-defined datatypes are possible (many facets to make restrictions, such as minValue, maxValue, pattern, etc.)
  - some other languages also use XML Schema's datatypes (TREX)
- object-oriented'ish
  - type extension/restriction
  - modularity (import/include mechanism, type libraries)

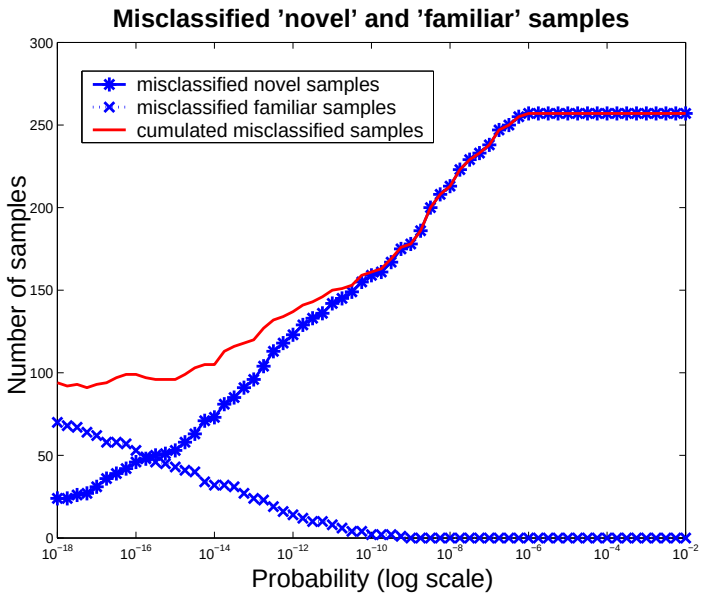

FIGURE 14. **Optimized novelty detection**
Misclassified samples vs. accepting threshold value.
Model parameters: Training set size:150 samples, Branching ratio:3, Depth of tree:5, "Familiar" dataset: Dat4, "Novelty" dataset: Dat5 (for generative parameters of artificial datasets see Table 11.3)

- Can express sets, i.e., can define the child elements to occur in any order
- Can specify element content as being unique (keys on content) and uniqueness within a region
- Can define multiple elements with the same name but different content
- Can define elements with nil content
- Can define substitutable elements - e.g., the "Book" element is substitutable for the "Publication" element.

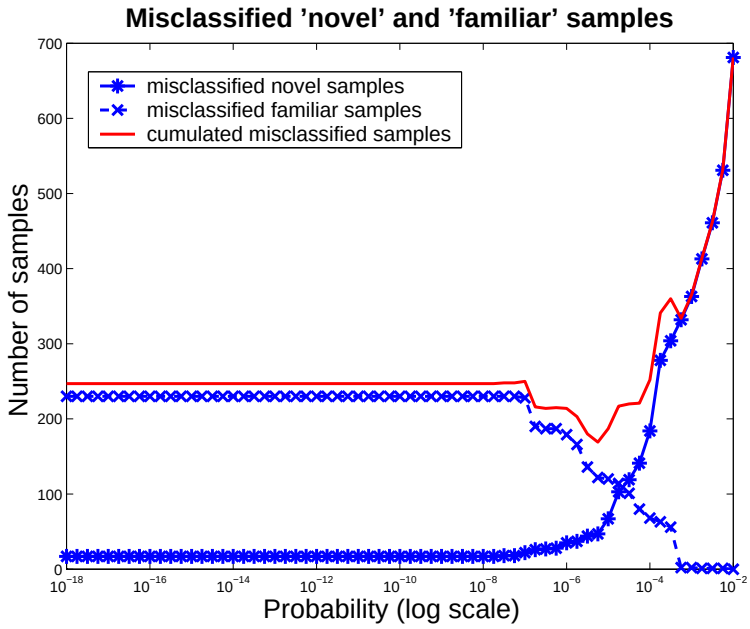

FIGURE 15. **Optimized novelty detection**
Misclassified samples vs. accepting threshold value.
Model parameters: Training set size:150 samples, Branching ratio:3, Depth of tree:4, "Familiar" dataset: Dat4, "Novelty" dataset: Garbage (for generative parameters of artificial datasets see Table 11.3)

Most of the other schema languages are very much like XML Schema. (see the comparative analysis done at UCLA by Wesley Chu and Dongwon Lee [75]). All are approaches to describe the document structure and data structures, except Schematron.

Schematron

The schemas discussed earlier (XSD, DTD) are constructed by defining parent-child and sibling relationships, as they are based on regular grammars. This means that there are still many constraints that are

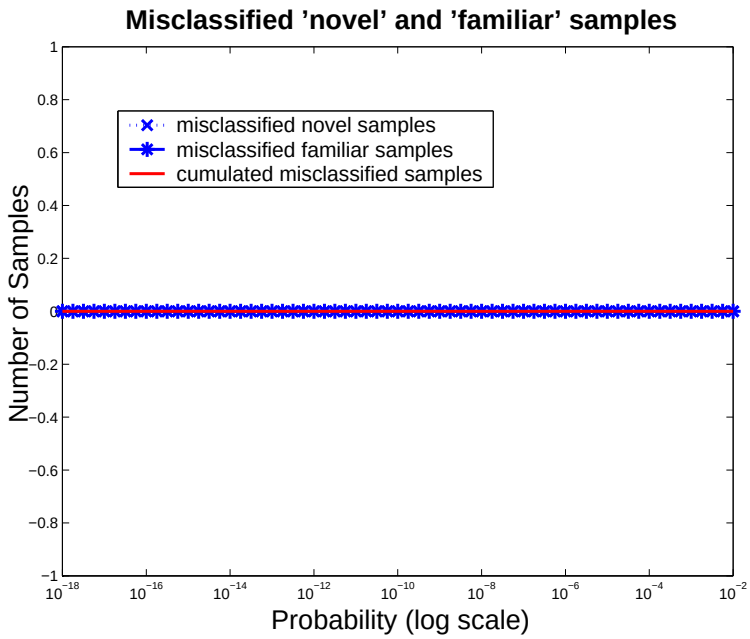

FIGURE 16. **Optimized novelty detection**
Misclassified samples vs. accepting threshold value.
Model parameters: Training set size:150 samples, Branching ratio:3, Depth of tree:5, "Familiar" dataset: ADC, "Novelty" dataset: Garbage (for generative parameters of artificial datasets see Table 11.3)

difficult or impossible to express with these 'content model-based' validation languages :

- Where attribute X has a value, attribute Y is also required
- Where the parent of element A is element B, it must have an attribute Y, otherwise an attribute Z

Schematron [3] uses a tree pattern based validation, which is characterised as a two step process of identification and then assertion. Both are done by XPath expressions (e.g. '//node' and '@value < 10'). Assertions and reports can be expressed and grouped into rules. A rule

T.S.	N.S.	B.R.	Depth	S.N.	Thrs	R.T.S	U.T.S	R.N.S	U.N.S.	O.M.	O.P.
Dat1	Dat2	3	4	2002	$1.78.10^{-8}$	959	42	1001	0	42	0.979
Dat2	Dat3	3	4	3002	$3.16.10^{-6}$	957	44	1001	0	44	0.985
Dat4	Dat2	3	4	1258	$1,78.10^{-6}$	212	45	1001	0	45	0.964
Dat4	Dat5	3	4	688	$1,78.10^{-5}$	185	72	423	8	80	0.883
Dat4	Dat5	3	5	688	$5.62.10^{-18}$	230	27	367	64	91	0.867
Dat2	Garbage	3	4	1801	$5.62.10^{-6}$	954	47	678	122	169	0.906
Cml	Garbage	3	5	1074	$5.62.10^{-18}$	246	1	798	2	3	0.997
Adc	Garbage	3	5	2983	0	2183	0	800	0	0	1

Abbrev	
T.S.	Training set. The dataset used to train a tree model.
N.S.	"Novelty set". The dataset that was shown to the model as novelty samples.
B.R.	Branching ratio of the covering tree.
Depth	Depth of the covering tree.
S.N.	Sample number. Overall number of samples in T.S. and N.S.
Thrs	Threshold. Optimal threshold of probabilities for accepting/rejecting samples.
R.T.S.	Recognized training samples.
U.T.S.	Unrecognized training samples
R.N.S.	Recognized novel samples
U.N.S.	Unrecognized novel samples
O.M.	Optimized Misclassification. U.T.S.+U.N.S. is minimal using the given threshold.
O.P.	Optimal performance (using the given threshold)

TABLE 3. **Novelty detection.** A set of samples was divided into two parts. The first subset was used for training a tree. The result was used for the estimation of probabilities from the other subset as well as for probability estimation of trees from different subsets. Optimal threshold (Thrs) values have been determined for the different datasets.

has a context attribute (an element, which it refers to). Then, patterns can be made from rules (these represent different kind of checks), and a schema can be created from patterns.

Schematron builds on existing technologies such as XSLT (a Schematron schema will be translated into an XSL stylesheet) and XPath. When the XQuery proposal is ready, it will be possible to use that language as a substitute for XPath in the identification of candidate

objects for validation. And extensions to the XSL technology can also improve Schematron.

In short, there are several aims which Rick Jelliffe which believed were important during the design and specification of Schematron:

- Promote natural language descriptions of validation failures, i.e. diagnose as well as reject
- Reject the binary valid/invalid distinction which is inherent in other schema languages
- Aim for a short learning curve by layering on existing tools (XPath and XSLT)
- Trivial to implement on top of XSLT
- Provide an architecture which lends itself to GUI development environments
- Support workflow by providing a system which understands the phases through which a document passes in its lifecycle

Target uses suggested by Jelliffe:

- Document validation in software engineering, through the provision of interlocking constraints
- Mining data graphs for academic research or information discovery. Constraints may be viewed as hypotheses which are tested against the available data

- Automatic creation of external markup through the detection of patterns in data, and generation of links
- Use as a schema language for "hard" markup languages such as RDF.
- Aid accessibility of documents, by allowing usage constraints to be applied to documents

XSL and its extensions

XSL [9] (with XPath) can also be used as a validation language ([57]). One can write macro functions (¡xsl:call-template¿), and use xsl's built-in comparison elements (e.g. ¡xsl:if¿). Some of the advantages of using XSL as a validation language are

- terse–the match patterns are very terse, like XML content models; declarative; simple
- fragment-friendly, since the interpretation of content models does not depend on the document context
- widely implemented

XSL can also be extended to allow user-defined functions to be called by the templates (JavaScript, Java functions, for example, using the ¡...xsl:script¿ element, or otherwise). However, this is implementation-dependant. Currently two different approaches were made to extend the functionality of XSL:

- EXSLT is an extension of XSLT which consists of a few modules of functions written in XSLT (Functions, Common, Math, Set)[1].
- FXPath were implemented on top of XPath [2]

XQuery

XQuery [8] is an XML query language. Its expressions are based on the XPath language, and its data model is like XML Schema's (XML Infoset working draft). The working group is making XPath more powerful (see Schematron above), and is creating a query language, which, like SQL, makes possible to extract information from XML documents. XQuery is not just a simple query language: it is also a data definition language (eg. XML element constructors are built-in, and any datatype which can be expressed in XML Schema can be used in XQuery), and user-defined functions can also be written.

Assertion Grammars

This document describes experimental work in progress at HP Labs - Bristol on formal techniques for describing combinations of modular tagsets for documents written in XML. The motivation is provided by the increasing diversity of web browsers, running on desktops, television, handhelds, cellphones or voice browsers.

The goal is to provide a means for document to be described in terms of an algebra operating over modules, which in turn are described as collections of assertions. It is hoped that this work will provide an interesting comparison with traditional approaches based upon Document Type Declarations, and more recent approaches, such as the drafts published by the W3C XML Schemas working group.

13.1. **Appendix A3: XML Comparison Tools. Comparing features in the IBM, Sun, DeltaXML and DOMMITT diff tools**

Feature	DOMMITT	IBM	Sun	DeltaXML
XML Parser used by tool	Oracle v2 Parser	IBM XML4J	Perl extension to expat	Unknown
Output format for diffs	New xmdiff nodes (same namespace as original doc)	XUL (XML Update Language) based on XPath	New diff nodes (same namespace as original doc)	New attributes (same namespace as original doc)
DTD generated for diff output	YES	NO (diffs not in XML format)	NO	DTD hand generated by company
Is output valid XML?	YES	NO	YES	NO

14. Appendix B: Used useful software tools

14.1. **Appendix B1: Tidy and Jtidy.** JTidy was used to convert html documents to XML form. This type of XML documents were used in the early phase of our studies. We learned that these type of documents have typically identical structures if those are downloaded from the same site. For example, all XML documents created from a movie database has the same XML structure. Different sites have very different XML structure not representing any challange for the probabilistic tree method[1].

Interestingly, this html based XML case could take advantage of tag, attribute and text based novelty detection.

14.2. **Appendix B2: Matlab and JAVA. List of Java compilers and virtual machines**

A list of compilers and VMs for the Java programming language, including short descriptions of price, supported platform and Java version. Includes native compilers (to convert class files to executables) as well.

http://www.mathtools.net/Java/Compilers

http://www.geocities.com/marcoschmidt.geo/java.html

[1]The case of many different sites should be, however different, but is, undoubtedly beyond our computational possibilities

Ptolemy II

The Ptolemy project studies modeling, simulation, and design of concurrent, real-time, embedded systems. Ptolemy II is a Java-based component assembly framework with a graphical user interface called Vergil.

http://www.mathtools.net/Java/Simulation

http://ptolemy.eecs.berkeley.edu/ptolemyII/index.htm

Hot Picks

Run Java programs and link to Java objects directly from MATLAB

http : *//www.mathworks.com/programs/javaprograms/index.html*

New MATLAB Add-in for Visual Studio automatically converts your C code to MATLAB-callable functions

http : *//www.mathworks.com/programs/matlab*$_a$*ddin/index.html*

MATLAB Database Toolbox lets exchange data with any ODBC/JDBC-compliant toolbox from MATLAB

http : *//www.mathworks.com/programs/databasetoolbox/index.html*

15. Appendix C: AI Tools

This section reviews the tree probabilistic technique. For the sake of completeness, a few selected AI methods that are considered relevant and novel from the point of view of combined probabilistic and rule

based identification and validation are also reviewed. The described AI techniques are as follows: novelty detection in general (subsection 15.2), fast intelligent restart point generation for satisfiability problems, learning parts, or subcomponents from examples to promote problem partitioning.

15.1. **Appendix C1: Learning a Tree distribution.** The solution to the ML Learning Problem has been published in [28] in the broader context of finding the tree that best fits a given distribution P over the dataset D. The goodness of fit is evaluated by the KullbackLeibler (KL) divergence [73] between P and T :

$$KL(P||T) = \sum_{x \in D} P(x) log \frac{P(x)}{T(x)} \tag{1}$$

Let us start by examining (1). It is known [30] that for any two distributions P and Q, $KL(P||Q) \geq 0$ and that equality is attained only for $Q = P$. The KL divergence can be rewritten as

$$KL(P||Q) = \sum_{x} P(x)[\log P(x) - \log Q(x)] \tag{2}$$

and thus

$$KL(P||Q) = \sum_{x} P(x) \log P(x) - \sum_{x} P(x) \log Q(x) \tag{3}$$

Notice that the first term above does not depend on Q. Hence, minimizing the KL divergence w.r.t. Q is equivalent to maximizing the second term of (2) (called the crossentropy between P and Q) and we know that this is achieved for $Q = P$.

Now, let us return to our problem of fitting a tree to a fixed distribution P. Finding a tree distribution requires finding its structure (represented by the edge set E) and the corresponding parameters, i.e. the values of $T_{uv}(x_u, x_v)$ for all edges $(u, v) \in E$ and for all values x_u, x_v.

Assume first that the structure E is fixed and expand the righthand side of (1):

$$KL(P||T) = \sum_{x \in D} P(x)[\log P(x) - \log T(x)] \tag{4}$$

making use of the following steps

(5)

$$KL(P||T) = \sum_{x \in D} P(x) \log \frac{1}{P(x)} - \sum_{x \in D} P(x) \log[\prod_{v \in V} T_{v|pa(v)}(x_v|x_{pa(v)})]$$

$$KL(P||T) = -H(P) - \sum_{v \in V} \sum_{x \in D} P(x) * \log T_{v|pa(v)}(x_v, x_{pa(v)}) \tag{6}$$

(7)

$$KL(P||T) = -H(P) - \sum_{v \in V} \sum_{x_v, x_{pa(v)}} P_{v,pa(v)}(x_v, x_{pa(v)}) \log T_{v|pa(v)}(x_v, x_{pa(v)})$$

we have

(8)

$$KL(P||T) = -H(P) - \sum_{v \in V} \sum_{x_{pa(v)}} P_{pa(v)}(x_{pa(v)}) \sum_{x_v} P_{v|pa(v)}(x_v|x_{pa(v)}) \log T_{v|pa(v)}(x_v|x_{pa(v)})$$

In the above, $H(P)$ denotes the entropy of the distribution P , a quantity that does not depend on T , and P_{uv}, P_v represent respectively the marginals of $\{u, v\}$, v under P. The inner sums in the last two lines are taken over the domains of v and $pa(v)$ respectively. When v is a root node, $pa(v)$ is the void set and its corresponding range has, by convention, one value with a probability of $Ppa(v)(x_{pa(v)}) = 1$. Moreover, note that the terms that depend on T are of the form

(9)
$$-\sum_{x_v} P_{v|pa(v)}(x_v, x_{pa(v)}) \log T_{v|pa(v)}(x_v, x_{pa(v)})$$

which differs only by a constant independent of T from the KL divergence

(10)
$$KL(P_{v|pa(v)}||T_{v|pa(v)})$$

We know that the latter is minimized by

$$T_{v|pa(v)}(.|x_{pa(v)}) \equiv P_{v|pa(v)}(.|x_{pa(v)}) \quad \forall v \in V \tag{11}$$

Hence, for a fixed structure E, the best tree parameters in the sense of the minimum KL divergence are obtained by copying the corresponding values from the conditional distributions $P_{v|pa(v)}$. Let us make two remarks: first, the identity (11) can be achieved for all v and $x_{pa(v)}$ because the distributions $T_{v|pa(v)} = x_{pa(v)}$ are each parameterized by its own set of parameters. Second, from the identity (11) it follows that

$$T_{uv} \equiv P_{uv} \quad \forall (u,v) \in E \tag{12}$$

and subsequently, that the resulting distribution T is the same independently of the choice of the roots. For each structure E we denote by T^E the tree with edge set E and whose parameters satisfy equation (12). T^E achieves the optimum of (1) over all tree distributions conformal with (V, E).

Now, with the previous results in mind, we shall proceed to the minimization of $KL(P||T)$ over the tree structures. First, notice that this task is equivalent to maximizing the objective

(13) $$J(E) = XxDP(x)logTE(x).$$

over all structures E.

Expanding the above formula and using (12) we obtain successively:

(14) $$J(E) = \sum_{i=1}^{N} P(x^i) \log T^E(x^i)$$

(15)
$$J(E) = \sum_{i=1}^{N} P(x^i)[\sum_{(u,v)\in E} \log T^E_{uv}(x^i_u x^i_v) - \sum_{v\in V}(deg(v)-1)\log T^E_v(x^i_v)]$$

(16)
$$J(E) = \sum_{i=1}^{N} P(x^i)[\sum_{(u,v)\in E} \log P_{uv}(x^i_u x^i_v) - \sum_{v\in V}(deg(v)-1)\log P_v(x^i_v)]$$

(17)
$$J(E) = \sum_{(u,v)\in E}\sum_{i=1}^{N} P(x^i)[\log P_{uv}(x^i_u x^i_v) - \log P_u(x^i_u) - \log P_v(x^i_v)] + \sum_{v\in V}\sum_{i=1}^{N} P(x^i)\log P_v(x^i_v)$$

(18)
$$J(E) = \sum_{(u,v)\in E}\sum_{i=1}^{N} P(x^i)\log\frac{P_{uv}(x^i_u x^i_v)}{P_u(x^i_u)P_v(x^i_v)} + \sum_{v\in V}\sum_{i=1}^{N} P(x^i)\log P_v(x^i_v)$$

(19)

$$J(E) = \sum_{(u,v)\in E} \sum_{x_u x_v} P_{uv}(x_u x_v) \log \frac{P_{uv}(x_u x_v)}{P_u(x_u) P_v(x_v)} + \sum_{v\in V} \sum_{x_v} P_v(x_v) \log P_v(x_v)$$

(20)

$$J(E) = \sum_{(u,v)\in E} I_{uv} - \sum_{v\in v} H(P_v)$$

Equation (15) follows from the undirected tree representation of T^E, (16) is obtained from (15) by taking into account (12); equation (18) follows from (17) by performing a summation over all $x \in D$ that have the same x_u, x_v and using the definitions of $P - uv$ and P_v; finally in equation (19) the terms I_{uv} under the first sum sign represent the mutual information between the variables u and v under the distribution P:

(21)

$$I_{uv} = \sum_{x_u x_v} P_{uv}(x_u x_v) \log \frac{P_{uv}(x_u x_v)}{P_u(x_u) P_v(x_v)}$$

The mutual information between two variables is a quantity that is always nonnegative and equals 0 only when the variables are independent.

Remark two important facts about equation (20): first, the second sum does not depend on the structure E; second and more importantly, the dependence of $J(E)$ from E is additive w.r.t. to the elements of

the set E. In other words, each edge in $(u, v) \in E$ contributes a certain positive amount to $J(E)$ and this amount I_{uv} is always the same, independently of the presence or absence of other edges and of the size of their contributions!

In this situation, maximization of J over all structures can be performed efficiently via a Maximum Weight Spanning Tree (MWST) algorithm [113] with weights $W_{uv} = I_{uv}, u, v \in V$.

The MWST problem is formulated as follows: given a graph $(V, \overline{E})$ and a set of real numbers, called weights each corresponding to an edge of the graph, find a tree (V, E), $E \in \overline{E}$ for which the sum of the weights corresponding to its edges is maximized. This problem can be solved by a greedy algorithm that constructs the tree by adding one edge at a time, in decreasing order of the weights W_{uv}. There are several variants of the algorithm: the simplest one, called Kruskal's algorithm, runs in $\mathcal{O}(n^2 \log n)$ time. Note that if all the weights are strictly positive, a tree with the maximum number of edges and $p = 1$ connected components will result. If some of the weights $W_u v$ are zero, it is possible to obtain trees with more than one connected component. More sophisticated MWST algorithms exist (see for example [113, 112, 45, 43, 105]) and they improve on Kruskal's algorithm on both running time and memory requirements. However, the running time of all published algorithms

Algorithm TreeLearn

Input	Probability distribution P over domain V
	Procedure MWST(weights) that
	fits a maximum weight spanning tree over V.
1.	Compute marginals P_v, P_{uv} for $u, v \in V$
2.	Compute mutual information I_{uv} for $u, v \in V$
3.	Call MWST (I_{uv}) that outputs the edge set E
	of a tree distribution T.
4.	Set $T_{uv} \equiv P_{uv}$ for $(uv) \in E$.
Output	T

TABLE 4. **Pseudo-code for the tree learning algorithm.**

is at least proportional to the number of candidate edges ($\overline{E}$). In our case, this number is equal to $n(n-1)/2$ since all pairs of variables have to be considered. Hence the best running time achievable for a MWST algorithm will be $\mathcal{O}(n^2)$. Henceforth, we will assume that the MWST algorithm runs in $\mathcal{O}(n^2)$ time and will not further specify the implementation. All of the above are summarized in the **TreeLearn** algorithm 4

The algorithm takes as input a probability distribution P over a domain V and outputs a tree distribution T that minimizes the KL divergence $KL(P||T)$. The running times for the algorithm's steps are as follows:

steps 1,2. For steps 1. and 2. (computing the marginals and the mutual information for all pairs of variables) the running time is dependent on the representation of P . But, generally, it should be expected to

be $\mathcal{O}n^2$, since there are $n(n-1)/2$ mutual information values to be computed.

step 3. The MWST algorithm takes $\mathcal{O}(n^2)$ operations (or $\mathcal{O}(n^2 \log n)$ in Kruskal's variant).

step 4. This step comprises only $\mathcal{O}(n{r_{MAX}}^2)$ assignments (remember that $|E| < n$). Hence, the total running time of **TreeLearn** $\mathcal{O}(n^2 + n{r_{MAX}}^2)$ or $\mathcal{O}(n^2)$ if we consider r_{MAX} to be a constant.

15.2. **Appendix C2: Connectionist novelty detection. Generative networks**

We investigate generative auto-associative networks. In such networks, channel capacity constraints form the main obstacle for effective information transfer. Robust and fast information flow processing methods warranting efficient information transfer, e.g. grouping of inputs and information maximization principles need to be applied. For this reason, indepent component analysis on groups of patterns were conducted using (a) model labyrinth, (b) movies on highway traffic and (c) mixed acoustical signals. We found that in all cases 'familiar' inputs give rise to cumulated firing histograms close to exponential distributions, whereas 'novel' information are better described by broad, sometimes truncated Gaussian distributions. It can be shown that upon minimization of mutual information between processing channels,

noise can reveal itself locally. Therefore, we conjecture that novelty - as opposed to noise - can be recognized by means of the statistics of 'neuronal firing' in generative auto-associative architectures.

Generative networks work by bottom-up processing of the input providing an internal representation and then top-down processing of the internal representation by means of the long-term memory [52, 49, 99, 80, 81]. This means that sensory processing is not simply coding, but also involves decoding (i.e., reconstruction). Such connectionist systems are called reconstruction networks [48, 47]. The internal representation is used to generate the reconstructed (expected) input and error is produced between the expected input and the actual input. Finally, this error is used to correct the values of the internal representation. The main objection against such iterative schemes is that those may not be fast enough [68]. According to [68] speed favors feedforward networks and may represent a challenge for iterative schemes.

On the other hand, iteration is not an obstacle for processing data if decoding is perfect. This is the case when coding and decoding invert each other [81]. Reduced to triviality, the steps of the iteration are as follows:

- Network starts with zeroed internal representation and with zeroed reconstructed input.

- Input is provided.
- Error becomes equal to the input.
- Coding gives rise to the internal representation.
- Reconstruction inverts (decodes) the internal representation.
- Error disappears.

. In turn, feedforward networks and generative networks will have the same one-step delay in the forming of the internal representation.

If we consider generative schemes then the optimization information transfer is of immediate consequence. Loss of information in bottom-up processing gives rise to deteriorated reconstruction. Efficient use of the channel capacity in the bottom-up (coding) and top-down (decoding) processes is a necessity in reconstruction networks.

Efficient Coding for Generative Networks

Efficiency is related to the possible speed of the whole coding-decoding process and is influenced by the capacity of the channels. Channel capacity measures the maximum rate (pulse/s, bit/s) for the given channel. In most cases channel capacity forms a hard constraint (the increase of the number of channels is costly), so we need other tools to make information transfer efficient. According to Shannon [104, 30]

optimal coding can be achieved by grouping the atomic units of the inputs first and by coding these new blocks instead of coding the atomic units themselves.

We argue that concatenation of disjoint inputs may be a smart trick applied. For example, inputs from different modalities, or temporal sequences, or both, i.e., spatio-temporal sequences can be grouped to improve the efficiency of coding.

The efficiency can also be improved by minimization of the mutual information between the processing channels. It is well known (see, e.g., [30]) that for statistically independent components the maximization of information transfer induces the minimization of mutual information between processing channels. Thus, we ought to consider the minimization of mutual information between processing channels, and processing needs to be shaped by independent component (IC) analysis (ICA) [15, 59, 53]. IC analysis in the linear case corresponds to a matrix transformation. Given a set of inputs, the components of the transformed input set, i.e., the components of the bottom-up processed input, will be as independent as possible in statistical sense. It is worth noting that ICA can be formed by means of local (Hebbian) learning rules, which makes it appropriate for neuronal modeling [53, 65].

Modeling Methods

We shall examine how the bottom-up processed input looks like upon minimization of mutual information. We shall use temporal sequences as inputs. A neural network accepts vector-valued inputs. We interpret 'sequences' by concatenating a few (say k) temporally subsequent inputs into a (k times) longer input vector. The method will be called embedding and the depth of the concatenation (i.e., k) will be the embedding dimension. After embedding, we present this new vector to the network that performs independent component analysis. Rows of the derived matrix correspond to vectors in 'sequence space'. These rows will be called temporal independent components (TICs) and the algorithm will be called TICA . Computation of the outputs of the matrix for a given input data will be called 'testing'. We make the following distinction between 'familiar' and 'novel' inputs. Inputs (examples) are divided into categories. For each category, respective bottom-up processing matrices are developed by means of TICA. TIC analysis in each category is performed by means of the training samples, a randomly selected subset of the examples of the category. Testing is performed by means of the examples not used for training. To each bottom-up processing matrix, we have within-category testing samples that will be called 'familiar samples'. We can also test samples from other categories that will be called 'novel samples'.

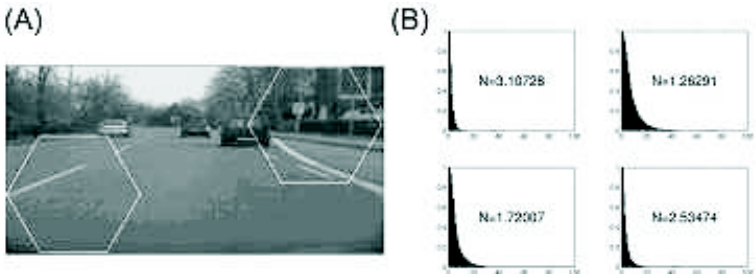

FIGURE 17. **Traffic example**
Hexagonal windows of 169 pixels in different positions (hexagonal areas in subfigure A) were used for creating the familiar and unfamiliar data sets. (B) Distributions of TICA outputs. 7 inputs were concatenated to form an embedded input. Diagonal (off-diagonal) histograms describe testing for familiar (novel) inputs. The negentropy (N) values [30] of the histograms (discretizations of the rectified distributions) are given within each subfigure.

In our computer studies, the FastICA [53] software package was used because FastICA has no adjustable parameters. It is of equal importance that FastICA makes use of negentropy, a quantity, which is invariant for invertible linear transformations. Upon minimization of mutual information, directions with roughly optimized negentropy values are provided. In larger simulations, PCA preprocessing was used for dimensionality reduction [46].

Results

TICA was performed on temporal sequences taken from different data sources. Two examples are shown in Figs. 17, 18.

In the first example a movie database was used (Figure 17). The movies were taken from a car in traffic. We used two input sets cut from

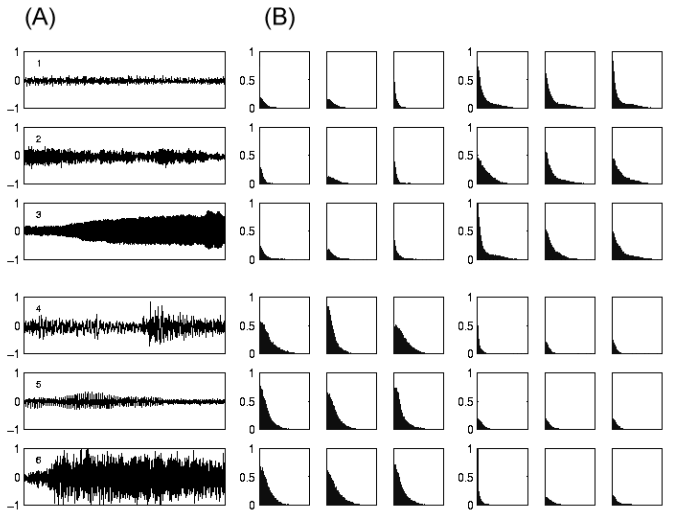

FIGURE 18. **The case of acoustic signals.**
(A) Samples are about 250 ms in length and are from different sources (e.g. music, sounds in a forest or sounds of a whale). (B) TIC output distributions. The training set for the TICA matrix was created from a mix of three samples out of the six signals. Mixing of the other three samples made the 'novel' inputs. Embedding depth is 16. The number of TICs is thus 3x16=48. Here - in contrast to Figure 17 - the histograms of randomly selected individual outputs are shown. Diagonal (off-diagonal) blocks of histograms represent familiar (novel) tests.

the same movie in different positions. Upon TIC analysis, histograms of familiar outputs are narrower (sparser) than histograms on novel outputs. This observation is numerically confirmed by the calculated negentropy values on the rectified distributions.

In the other example acoustic signals were used (Figure 18).

Discussion

In a hierarchical system, noise at one level may be grouped into novel information at another level. In fact, the optimization of information transfer requires that frequent symbols need short codes (low level in the hierarchy), whereas less frequent symbols need longer codes (higher levels in the hierarchy). A less frequent symbol may seem nothing but noise at low levels. ICA and TICA are optimal to make this distinction. Denoising (noise filtering) of IC outputs can be approximated by diminishing small amplitude outputs [56]. In turn, noise filtering is local. Reconstruction is performed by high activity IC outputs. Noise appears in the reconstruction error. This reconstruction error can be analyzed by higher levels, as suggested in [100]. Recognition of novelty is the task of a higher level and works on large reconstruction errors (large residuals) in a given area. Our simulations imply that the recognition of novelty is 'easy' and can be instantaneous in optimized generative networks. We found on different examples, that a novel (not yet seen, not yet optimized) input gives rise to distinctly different statistical properties at the level of internal representation.

Computational considerations, such as entropy minimization, factor analysis, or sparse coding lead to representational schemes that optimize information transfer [12, 14, 37, 41]. Factor analysis and ICA have

been considered as the main strategy of encoding sensory information for a long time.

We stress two things. The emphasis is on forming blocks of symbols, i.e., concatenating in time, grouping in space, and grouping between different modalities. Secondly, none of the test samples was used during training, but some represented similar, whereas others represented dissimilar statistics. ICA is working on the higher order moments of the activity distribution and tends to make this distribution narrower (sparser). The learned statistics is thus expected to be sparser. Statistical properties can form the basis of novelty recognition in optimized generative networks. This feature is not specific to generative schemes [98]. A feedforward network that optimizes information transfer will behave the same way.

15.3. **Appendix C3: Pattern completion with probability estimation.** In most pattern recognition problem noise filtering is of central issue. The separation of noise from data is, however, problem- dependent. In information theory, noise is considered structure-free, i.e., of maximal entropy. (For continuous variables, Gaussian distribution of unit variance has the maximal entropy). The recently introduced sparse code shrinkage (SCS) algorithm [54], aims to separate Gaussian noise from structured components. This novel approach can be seen as

a generalization of wavelet denoising [54]. The generalization concerns the process of the learning of the underlying basis set given an ensemble of inputs and then performing *denoising*, alike in wavelet bases. SCS performs well on inputs that are linearly mixed from independent sources. SCS originates from independent component analysis (ICA, also called blind source separation, or demixing), which has almost a ten year history [60, 29, 16, 11, 85]. The objective of the mathematical derivations of ICA algorithms is to optimize information transfer for linearly mixed inputs [55]. ICA removes higher order correlations from components. At the same time it is not known how the ICA algorithms perform in general information transfer optimization problems (i.e., without this independence assumption). ICA transformed information has limited power in pattern completion problems that assume correlations between components.

Decomposition of multivariate data into correlating sub-structures is needed in pattern completion problems, e.g., for occlusion. The objective of learning is to seek sub-parts in individual inputs of an ensemble to enable inferencing. A powerful recent technique is non-negative matrix factorization (NMF, [76, 77]), which aims to find sub-structures in a given set of inputs, with each input built from non-negative components.

Architecture

SCS is a bottom-up transformation that aims to recover the original sources given the mixed input. In case of independent sources covered by additive Gaussian noise, SCS can recover the original inputs. NMF, on the other hand, can be seen as a top-down generative algorithm that optimizes the internal representation in order to minimize the reconstruction error between input and generated (reconstructed) input. The two algorithms can be merged into a single auto-associative architecture as shown in Fig. 19(A). Here, $\mathbf{W}$ denotes the bottom-up ICA transformation that together with denoising produces the MMI components, whereas $\mathbf{Q}$ is the NMF generative matrix.

We investigated learning capabilities and working performance of the proposed architecture. It exhibited good parameter-free performance under the following conditions:

First, bottom-up learning The MMI components were developed using the non-linear SCS method (see Appendix). The bottom-up demixing matrix is learned in this phase. This is a single parameter procedure. Only non-zero values of this parameter, the kernel width d, can lower mutual information.

Second, top-down learning: Inputs are filtered by the bottom-up matrix $\mathbf{W}$ and the SCS non-linearity is not applied. The linear outputs

are multiplied by the pseudoinverse $\mathbf{W}^+$ of matrix $\mathbf{W}$ (which is equal to $\mathbf{W}^T$ for our case [54].) (In other words, the input is projected into the SCS subspace defined by the row vectors of matrix $\mathbf{W}$. Each projected input is then shifted and scaled (the lowest and the highest pixel values are made equal to 0 and 1, respectively). These bounded and non-negative inputs are subsequently used in batch mode to compute the NMF basis set (or NMF matrix, $\mathbf{Q}$).

Working phase: Inputs are projected into the SCS subspace as before. Hidden components $\mathbf{h}$ are computed via the NMF iteration procedure ($\mathbf{I}_{NMF}$) keeping the $\mathbf{Q}$ matrix fixed. The iteration minimized the mean square of the reconstruction error. Reconstructed input $\hat{\mathbf{x}}$ is computed by multiplying hidden vector $\mathbf{h}$ by NMF matrix $\mathbf{Q}$.

A hierarchy is built by combining hidden representations into blocks. Forming of blocks can take advantage of neighboring relations, e.g., prewired neighbors of topographical maps). A two layer architecture is shown in Fig. 19 (B). First approximation of the hidden values are computed by the first layer. These hidden values make up the input of the second layer. The reconstructed input of the second layer then overruns the hidden values of the first layers. This leads to corrected reconstructed inputs $\hat{\mathbf{x}}$ at the first layer.

The bar example

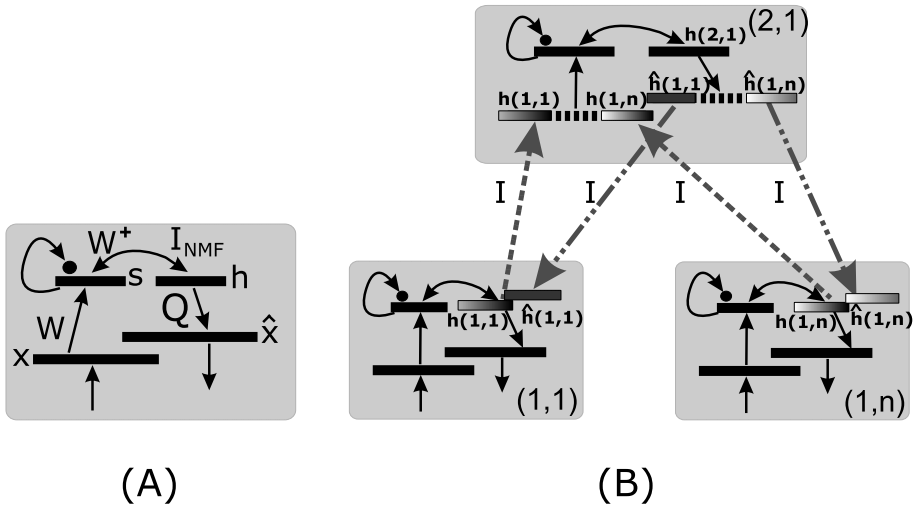

FIGURE 19. **Graphical representation of the algorithm.** *(A):* $\mathbf{x}, \mathbf{s}, \mathbf{h}$ and $\hat{\mathbf{x}}$: input, shrunk (denoised) ICA components, hidden variables, and reconstructed input, respectively. $\mathbf{W}$, $\mathbf{W}^+$, $\mathbf{Q}$ and $\mathbf{I}_{NMF}$ denote demixing matrix, pseudoinverse of the demixing matrix, NMF matrix and NMF iteration, respectively. Arrow: linear transformation, arrow with black dot: linear transformation with component-wise non-linearity (shrinkage kernel), lines with two arrow-heads: iteration. The algorithm was utilized in a two phase mode (see text for details).
(B): $\hat{\mathbf{h}}$: reconstructed input of second layer (i.e., hidden variables of first layer 'according to' the second layer). Two-layer hierarchy. Lower layers provide inputs (their hidden variables $\mathbf{h}$) for higher layers. Reconstructed input of second layer overruns hidden variables at first layers. Hidden variables are subject to NMF iteration. Grey arrows represent identity transformations (I).

The following two-bar example is used for demonstration purposes. The tasks are identification and reconstruction of bars on gray-scale images while (i) each input is composed of two or more orthogonal bars, (ii) noise is present in the sense that input is covered by additive Gaussian noise of zero mean and the variance as a free parameter, and (iii) in the hierarchical setting the bars may have missing parts. We studied the effect of the signal to noise ratio (SNR) on the quality

of the reconstruction. The 'original' inputs consisted of 2 or more white bars (represented by 1's for each vector component), whereas the background was black (0 value vector component). The inputs were corrupted by additive Gaussian noise with different standard deviations (STD). For our purposes, signal-to-noise ratio (SNR) will be defined as the ratio between the maximal pixel value of the true noise-free input (i.e., 1.0) and the STD of the additive Gaussian noise. An example of the noise-free perfect input togther with its noisy version are shown in Fig. 20(A) and (B). The reconstructions of the input are shown in Fig. 20 (C)-(E). See the figure caption for details.

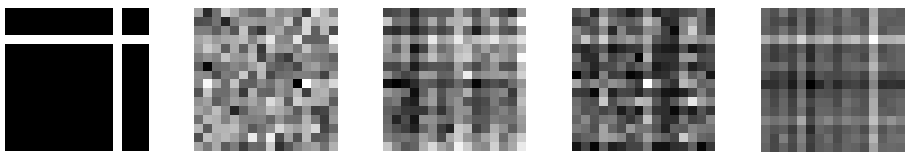

FIGURE 20. **Input and its reconstructed forms**
(A): perfect input without noise, (B): noise covered input, (C): reconstructed input (RI) with SCS, (D): RI with NMF, and (E): RI with combination of SCS and NMF. Signal-to-noise ratio is: 0.83. Note the improved reconstruction for the combined method compared to single SCS or single NMF algorithms.

Details of the SCS procedure are presented in the Appendix. In the NMF iteration the Euclidean cost function was used [76].

Simulation results are shown in Figs. 20, 21 and in Fig. 23. Figure 20 (B) depicts one of the inputs used for training and testing. The SCS, NMF, and combined SCS and NMF reconstructed inputs are shown

in Fig. 20(C)-(E). NMF is not capable to work with such high noise content, reconstructed input is mostly noise. Instead, the combined method 'predicts' higher amplitudes for pixel values corresponding to the 1's of the original (clean) input.

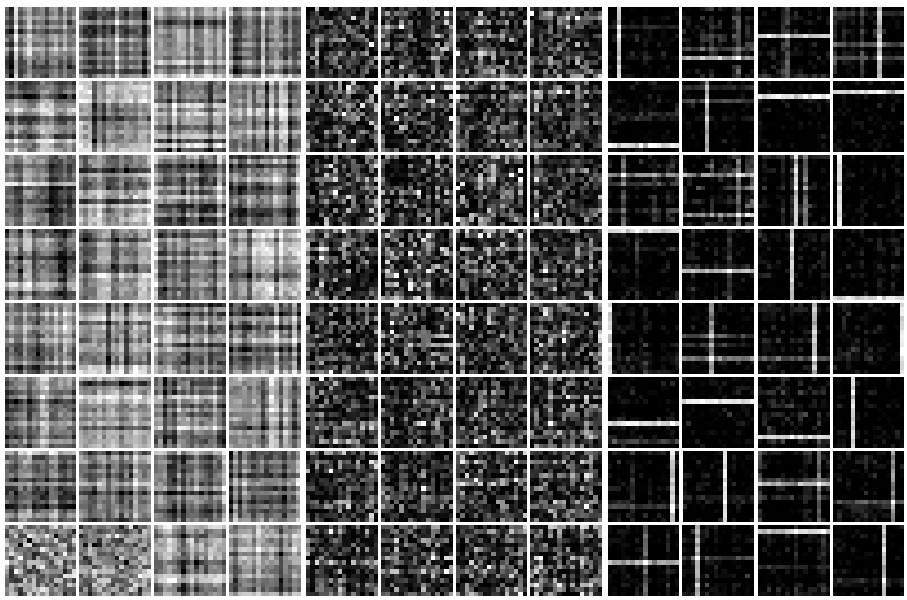

FIGURE 21. **Single layer basis sets for SCS alone (A), NMF alone (B) and for the NMF using SCS outputs in linear mode (C).**
Inputs have 256 (= 16 × 16) dimensions. Number of filters: 32 (= 2 × 16). For each method, all 32 filters are depicted. SNR=0.67

Basis sets for the three methods are shown in Fig. 21. According to the figure, NMF is not capable to overcome the high noise content, whereas SCS discovers the high-order line-like correlations in the input but is not capable to decompose the input into subcomponents for the

chosen noise parameters, whereas the combined method shows robustness against noise and is able to extract subcomponents (single vertical and horizontal lines).

Quantitative dependencies on noise and shrinkage parameter (kernel width of the SCS, see Appendix) are shown in Fig. 22. Noise resistivity is depicted in Fig. 22 (A). The figure shows the root mean square (RMS) error between perfect (noise-free) double-bar inputs and reconstructed inputs for the different methods. Results are superior for the combined method. SCS performs badly, because SCS could not discover the underlying structure and the SCS basis vectors can not cover all the possible inputs. Performance of NMF is better in spite of the fact that NMF basis vectors can not represent the underlying structure for noisy inputs. Another test concerned performance on perfect (noise-free) single-bar inputs. RMS reconstruction error versus noise is depicted for SCS, NMF and for the combined method in Fig. 22 (B). RMS error is small but non-zero, because the filters developed on noisy inputs are not perfect. As before, the combined method is again the best in this case. Fig. 22 (C) shows the dependence of the RMS on the kernel parameter d for different signal-to-noise levels (see Appendix). For all cases, the combined method has significant advantages because of two reasons: It filters out the noise and completes the reconstructed

input according to the information encompassed by the NMF matrix and used in the NMF iteration. In the working phase the sparsifying

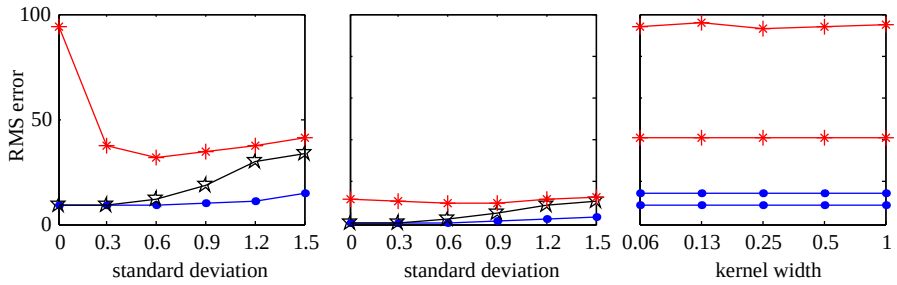

FIGURE 22. **Parameter dependences.**
Colors denoted different methods. SCS: cross, NMF: star, SCS and NMF: dot. (A) Dependence of RMS reconstruction error on standard deviation (STD) of the noise, (double-bar inputs) (B) RMS reconstruction error for noise-free perfect single bar inputs versus STD of noise during training phase (single-bar inputs), and (C) RMS reconstruction error for SCS (crosses) and for the combined method (dots) versus kernel width for the noise-free case (black lines) and for noisy input (grey lines). STD=1.5. Note the remarkable independece on the kernel parameter, the single adjustable parameter for a given architecture.. Kernel width for (A) and (B): 0.06

non-linearity of the SCS transformation was not used, the SCS stage worked in linear mode. This operation was chosen for reproducibility reasons: we found that the non-linearity (i) does not improve results significantly, but (ii) introduces irregular parameter dependences (not to be detailed here). SCS developed different filters for different kernel width values. These linear filters (that were formed with different kernel width values) together with the NMF iteration reduced the noise and showed robust insensitivity to the kernel width. This is the combined result of the low dimensionality of the sub-structures (32) relative

to the dimension of the input (256), and the excellent filtering capabilities of the NMF iteration.

In completion tasks we compared the efficiency of the combined method in a hierarchy. We studied the case of a two layer hierarchy (Fig. 23), with the first layer consisting of several compartments as shown in Fig. 19 (A), each working on a localized subarea of the input, and the second layer collecting the results from the first layer over the entire input range as shown in Fig. 19 (B). Two types of inputs were generated. The upper row shows an example when the original (noise-free) input has missing pixel components. The lower row depicts the case when full layer 1 subcomponents are missing. Noise-free, incomplete inputs are shown in column (A). Visual examination reveals that input reconstruction improves both for the missing pixel case as well as for the missing subcomponent case.

Mathematical details

The combined algorithm can be summarized as follows. Let $\mathbf{x} \in \mathbb{R}^n$ denote the input to the system. $\mathbf{W}$ denotes the SCS (denoising) filter and $\mathbf{Q}$ denotes the NMF filter. Tuning equation for the SCS filter is as follows:

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \lambda(k)\mathbf{q}(\mathbf{W}(k)\mathbf{x})\mathbf{x}^T + \frac{1}{2}(I - \mathbf{W}(k)\mathbf{W}(k)^T)\mathbf{W}(k) \tag{22}$$

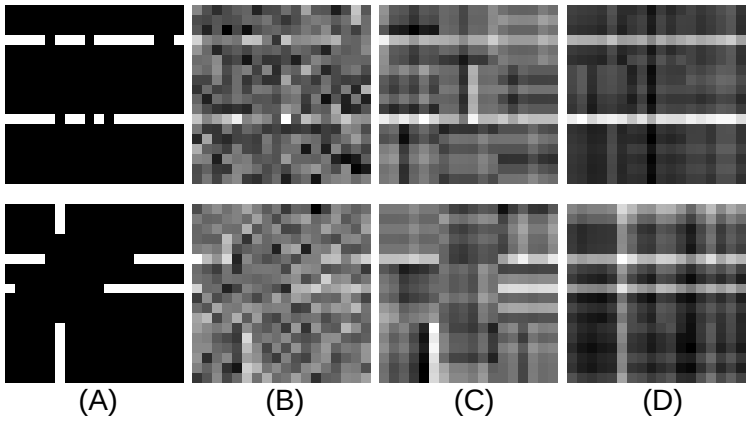

FIGURE 23. **Improved noise filtering and pattern completion in the hierarchy.**
Subarchitectures of the first layer have $6 \times 6 = 36$ input dimension. Dimension of NMF hidden vectors of first layer units is 12. First layer is made of $3 \times 3 = 9$ units. Input of the second layer has 108 dimensions. Dimension of the hidden vector of the second layer is 36. *Upper row:* Pixels of the inputs are missing. *Bottom row:* Pixels *and* sub-components of the inputs are missing. (A) Original input with missing pixels and sub-components, (B) input to the architecture, (C) reconstructed input using first layer reconstructions only, (D) reconstructed input using the full hierarchy. SNR is 0.5.

where k is the number of the training session, $\lambda(k)$ is the learning rate, and $\mathbf{q}(.)$ denotes a component-wise non-linear function of the following form: $q_i(\mathbf{u}) = -u_i \exp(-u_i^2/d^2)$, that is the non-linearity is applied separately to each component of the vector argument. Assuming that $\lambda(k)$ satisfies the Robbins-Monro condition, this set of equations has a single parameter, the kernel width, d [54].

For NMF the multiplicative update rules (a dual iteration) were used to minimize the Euclidean distance based cost function. These iterative

equations were used in batch mode for learning:

$$\mathbf{H}_{ij}(k+1) = \mathbf{H}_{ij}(k)\frac{(\mathbf{Q}^T(k)\mathbf{V})_{ij}}{(\mathbf{Q}^T(k)\mathbf{Q}(k)\mathbf{H}(k))_{ij}} \tag{23}$$

$$\mathbf{Q}_{ij}(k+1) = \mathbf{Q}_{ij}(k)\frac{(\mathbf{V}\mathbf{H}^T(k))_{ij}}{(\mathbf{Q}(k)\mathbf{H}(k)\mathbf{H}^T(k))_{ij}} \tag{24}$$

where columns of matrix $\mathbf{H}$ represent the hidden representation vectors, $i = 1, \ldots, n$ (n is the dimension of the hidden vector), $j = 1, \ldots, m$ (m is the number of inputs). Matrix $\mathbf{H}$ is formed by iteration 23. This iteration makes use of the pseudoinverse $\mathbf{W}^+$ of $\mathbf{W}$: $\mathbf{V}_j = \mathbf{W}^+\mathbf{S}_j$, where $\mathbf{S}_j$ denote the j^{th} output of the *linear* SCS transformation. In the learning phase both equations 23 and 24 are used for the NMF iteration. In the working phase only the first equation is used for single inputs and the SCS and NMF matrices ($\mathbf{W}$ and $\mathbf{Q}$) are not modified.

15.4. **Appendix C4: Fast solution to the satisfiability problem.**

One of the most fundamental activities in an engineer's life is to find good, probably the best (i.e., *optimal*) choice of different alternatives in situations offering many alternatives while respecting certain limitations. Given a *problem* one usually *evaluates* the *possibilities* according to some *strategy* and finally selects one, which seems optimal for him.

In engineering the problem of finding the best possible answer from a large space of possible answers is called *global optimization*. Formally speaking a global optimization task consists of the following tuples: a *state space* X together with a scalar *objective function* $Obj : X \to \mathbb{R}$ used for assessing the states. Our goal is to find the "best" state $x^* \in X$, which minimizes[2] Obj, that is, $Obj(x^*) \leq Obj(x)\ \forall x \in X$.

If the state space is small, then x^* can be trivially obtained by exhaustive search, otherwise special knowledge or some heuristics have to be utilized for performing effective partial search [17]. Many practical *combinatorial optimization problems* in engineering where X is finite, but enormous in size fall into the class of *NP-Hard* problems [31], [112], to which no efficient exact solution algorithm is known.

One possibly adapts the solution algorithm according to his best knowledge in order to incorporate his special "insights" regarding the problem. The main problem comes from the fact that the true nature of the problem can be very well hidden behind the objective function, therefore searching only by objective function can be very inefficient.

In the theory of Reinforcement Learning [110] the fundamental instrument in decision making is the so called evaluation function, or shortly *value function*. It maps *features* of a given state to a single

[2]In the followings when we speak about optimization, we will always assume minimization, unless explicitly noted.

scalar value that gives us the "promise of the state" with respect to the solution to the problem [20]. Obtaining this mapping is not a straightforward task. The good news is that evaluation functions can be *learnt* automatically [110], [20], [22].

One of the recent advances in this field is the algorithm STAGE [20]. It is a local search (LS) based optimisation tool, that falls into the class of multiple-restart local search algorithms [17]. Its novelty is the use of value function approximation to find promising restart state such that let the chance of finding a global optimum during the next LS be the highest possible. To achieve this STAGE maintains an *approximation* of the *value function* of the states. The state with the best value function is the candidate for the next restart. This algorithm performs surprisingly well on many classical optimisation problems, such as bin-packing, VLSI routing, geographic cartogram design, binary satisfiability, etc [20], [21], [22].

STAGE (a function approximator based problem solver) was applied to different problems. First, results on a benchmark CNF database (ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/benchmarks/cnf/) are shown and discussed. See results below: Figure 24 provides a view why sweeping (longer and longer) randomly restarted search instants are suggested by the Cornell group. If good result can be found soon

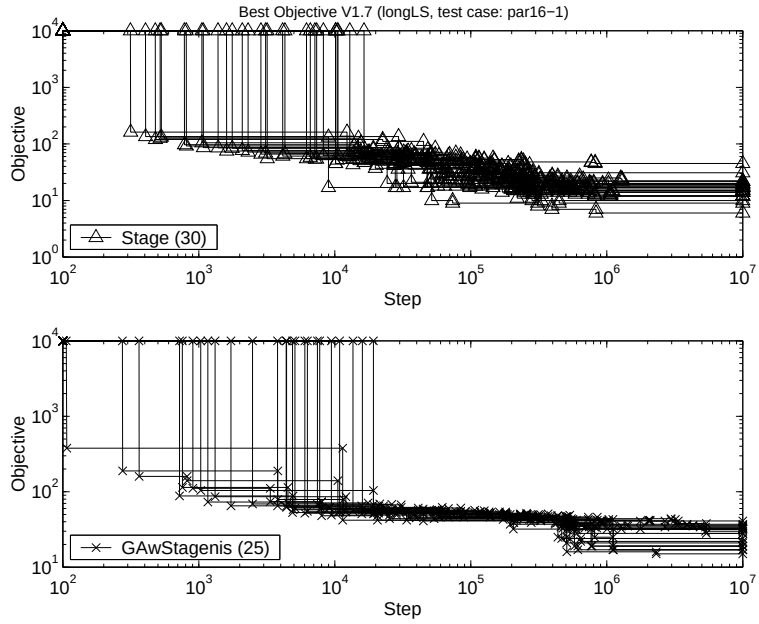

FIGURE 24. **Results with STAGE and it parallel form on a parity problem from benchmark from DIMACS**
Upper sub-figure: Stage alone. **Lower sub-figure**: Up to improvement step number 4×10^5 the runs are equivalent to 100 parallel Walksats. If those Walksats were run parallel then the results would be the same upon two orders of magnitude shorter computation time. In fact, Walksat stops improving at around 2×10^4, i.e., much sooner than step no. 4×10^5. At step $4x10^5$ STAGE is turned on and intelligent restarts are generated.

– then go for it! However, random restarts (RR) correspond to gambling. If there is a structure in the problem, RR has no strategy to find it. Function approximator (FAPP) methods combined with RL technology can discover and make use of the structure. The figure depicts the results of a quadratic FAPP that with an approximate and fast RL method (STAGE) may improve the results considerably. Database: ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/benchmarks/cnf/

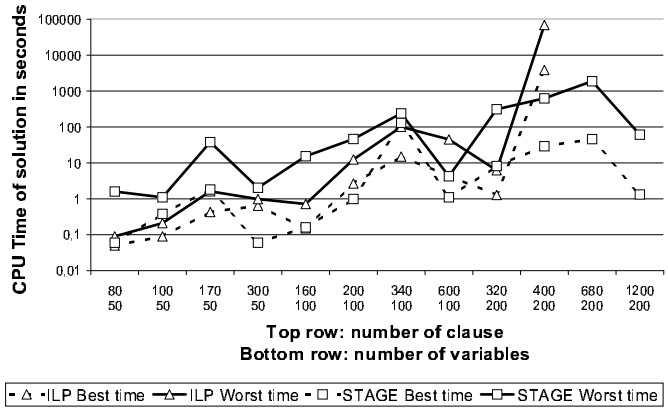

FIGURE 25. **Results on aim-x-y-z-'text' problems**
Comparisons with CPLEX, a commercially available Integer Linear Programming routine.

Problems: par16-1, par32-1-c. Larger dataset was tried on the 'aim' problems from database. See Figure 25. Results on aim-x-y-z-'text' where x,y and z are number. x is the number of clauses, y is the number of the variables, z is the number of the problem within category are depicted in Figure 2. We used clauses with 'text'=yes1, which means that the problem can be solved. The figure shows the best and the worst CPU times for 4 CNFs randomly selected for each category. CPLEX shows faster performance for smaller problems, whereas there is a clear advantage for STAGE on larger problems.

One may ask the following question: 'What types of data deems appropriate for this methodology?' Also, what data would be inappropriate?

In my view, no definite limitation can be given. 'Hard problems' can be clearly defined (see later). Reinforcement learning (RL) equipped FAPP is a general methodology for continuous variables and it suits – as demonstrated – satisfiability problems, too. It has the capability of adding heuristics. Heuristics means:

- deterministic state-action pairs, i.e., rules that can not be modified,
- deterministic state-action pairs, i.e., rules that can be modified,
- stochastic state-action pairs, i.e., rules that can be modified in risk sensitive way,
- model, including differential equations - spatio-temporal priors.

'**The hard part of the problem**' is to learn features. A feature is a (non-linear) combination of variables that decreases search space in an efficient manner. The method can include such features, these features can be 'wired-in' and can be, possibly, learned. No method has been given for the automatic discovery of features. In the examples given, we used no feature beyond the objective, the number of clauses that

have been satisfied. Demonstrations, in turn, are clean comparisons and are not corrupted by 'pre-processing'.

Given the importance of feature extraction, I shall elaborate on this issue in Appendix refss:Acomplete

16. Appendix D: Internet crawler

The number of documents on the world-wide web is way over 1 billion [35]. The number of new documents is over 1 million per day. The number of documents that change on a daily basis, e.g., documents about news, business, and entertainment, could be much-much larger. This ever increasing growth presents a considerable problem for finding, gathering, ordering the information on the web. The only search engine that may still warrant that the information it provides is not older than 1 month is AltaVista[3]. However, the number of indexed pages on Altavista is about 250 million documents. Google[4], on the other hand, is indexing about 1,300 million pages, but Google does not warrant any refresh rate on those documents.

The problem is complex: These search engines are not up-to-date and information gathering is not efficient with these engines. Search engines may offer too many documents; sometimes on the order of

[3]http://www.altavista.com
[4]http://www.google.com

hundreds or many thousands. Many of the pages form traps, e.g., by making use of particular (sometimes fake) keywords, or being simply collections of documents indexed in every 'dimension' of the web.

Specialized crawlers, possibly personalized crawlers are in need. This problem represents a real challenge for methods of artificial intelligence and has been tried by several research groups [27, 33, 24, 87, 71, 74, 88, 95, 96]. One of the first attempts in this direction was made by Chakrabarti et al. [25] who put forth the idea of *focused crawling*. To understand the idea, let us consider crawling in general. Assume that you are at a node of the web. This node has been analyzed and you have what to do next. It is very possible that relevant information can be found in the immediate neighborhood of this node. In turn, you download all the documents next to you and start to analyze those documents. Doing so, you may found relevant documents or may not. When you are done you have the option to download all the documents that are two steps away from you and to analyze those documents. This procedure is well known in the AI literature and is called breadth first technique. Although the number of nodes and the number of documents on the world-wide web is huge, still, taking another viewpoint, the world-wide web is *'small'*: It is small in terms of the shortest paths between nodes. Distance between can be defined

by The shortest path between two nodes can be defined as a kind of distance between them. Given this distance, one may talk about the diameter of the WWW. The WWW had about 800 million nodes in 1999 and the diameter of the web was around 19 [10]. In turn, breadth first search has an enormous burden and should be avoided. If computation power, or communication bandwidth, or both are limited, then at one point breadth first search needs to be abandoned and a decision is to be made about where to move next. To make a move to a node the value of that node needs to be estimated from the point of view of the search problem. In the focused crawling approach, an attempt is made to classify the content of the document from the point of view of the search topic. If the document falls into the search category then the document is downloaded and the links of the documents are followed.

It has been recognized by Diligenti et al. [35] that searched information on the web is typically *hidden*. One of the reasons for this hidden property is that we need to search (the location of the information is not apparent for us) and our search is influenced, sometimes derailed by direct and indirect advertisements. Another reason is that sites of particular interest may have a lower number of directed links then sites of general interest. In turn, we might face the 'needle in the haystack'

problem with the haystack being sites on general interest. The hidden property is thus the implicit consequence of our particular interest.

Let us consider sites dealing with support vector machines (SVMs). Sites about SVMs are not typical on the web. Not all sites dealing with SVM are linked. In turn, focused crawling could be rather inefficient and this direct search method for SVM sites might fail. On the other hand, most of the SVM sites are within (i.e., linked to) academic environments, sites dealing with information technology and these topics are much more general and might have much more links and a much higher 'visibility'. In turn, searching for the environment of SVM sites, could be much more efficient. The environment of the document, i.e., documents that are one step away, documents that are two steps away, etc., form the 'context' of the document. The idea is that when we search for a document we shall encounter the environment of the document first. In turn, we might as well search for the environment of the document first. This is the idea behind 'context focused crawling' (CFC) [35]. CFC – to our best knowledge – is superior to other crawling techniques to date. The CFC idea, which is trivial for worlds of small *effective* dimension (i.e., for graphs with small branching ratio), could be criticized for the case of small worlds (i.e., worlds of high effective dimension) when documents – on average – are about as far as

the environment of the document. However, the question is intriguing, because the visibility (as defined above) could be much less for searched documents than that of the environments of the searched documents.

The problem is that environments may differ. Good performance on the estimation of the value of one environment could be misleading on the estimation of another. In turn, the estimation whether to stay and download at a given site or to do not download but move further can be seriously jeopardized without adaptation. Given the high branching ratios at most of the sites, such adaptation could be performed at any of the sites, provided that the learning procedure is fast. Although value estimation has not been found particularly efficient for searching the world-wide web [101], this is not against value estimation in general, but calls for the improvement of the feature extraction utilized. Here we show combine machine learning technologies for better feature extraction on the web.

Preprocessing of texts

There is a large variety of methods that try to classify texts [89, 70, 19, 39, 66, 13, 23, 50, 69, 94, 90, 91, 51, 97, 61, 116, 117, 58, 118, 36]. Most of these methods are based on special dimension reduction. First, the occurrence, or sometimes the frequency of selected words is measured. The subset of all possible words ('bag of words' (BoW)) is

selected by means of probabilistic measures. Different methods are used for the selection of the 'most important' subset. The occurences (0's and 1's) or the frequencies of the selected words of the subsets are used to characterize all documents. This low – typically 100 – dimensional vector is supposed to encompass most of the information about the type of the document. Different methods are used to derive 'closeness measures' between documents in the low dimensional spaces of occurence or frequency vectors. The method can be used both for classification, i.e., the computation of decision surfaces between documents of different 'labels' [19, 39, 58, 97] and clustering, a more careful way of deriving closeness (or similarity) measures when no labels are provided [89, 66, 51, 61, 117].

We tried several BoW based classifiers and found that the tried methods can not classify web documents over 50% for the 'Call for Papers' (CfP) problem[5]. CfP is considered a benchmark classification problem of documents: The ratio of correctly classified and misclassified documents can be automated easily by checking whether the document has this three word phrase (i.e., call for paper) within the document or

[5]The CfP problem is defined by deleting the phrase 'call for paper' from the document, executing search on the internet and considering each document that contains the phrase 'call for paper' a 'hit'.

not. We found, in agreement with published results [39], that supervised SVM classification method is simple, fast and is somewhat better than Bayes classification.

Classification

The SVM classifier is a form of classifiers that has favorable generalizing capabilities [115, 107, 67]. Examples had been collected for training. Other classifiers could be used, too. The SVM classifier was chosen because of its superior generalizing capabilities and because it has no adjustable parameters and, in turn, parameter adjustments are simplified for SVM. The trained SVM was used soft mode. That is, the output of the SVM was not a decision (yes, or no), but instead the output could take continuous values between 0 and 1. A saturating sigmoid function[6] was used for this purpose. In turn, (i) the non-linearity of the decision surface was not sharp, (ii) at around the decision surface each classifier provides a linear output proportional to the 'distance' from the decision surface, and (iii) this distance is bounded by the sigmoid function. These distances can be considered as distinct yardsticks working on distinct features. A subsequent algorithm, the *value* of each yardstick can be estimated.

Value estimation

[6] $\text{output} = \frac{1}{1+\exp(-\lambda * \text{input})}$

There is a history of value estimation methods, called reinforcement learning [72, 93, 108, 119, 103, 82, 32, 62, 102, 79, 114, 18, 84, 110, 111, 109, 106, 34]. In our approach, value estimation plays a central role. Value estimation works on states (s) and provides a real number, the *value*, that belongs to that state: $V(s) \in \mathbb{R}$. Value estimation is based on the *immediate reward* (e.g., the number of hits) that could be gained at the given state by executing different actions (e.g., download or move). The idea is to estimate long-term cumulative reward based on an action–to–probability mapping (called policy) that may differ for each state and that determines the probability distribution of possible action choices in each state. Policy improvement and the finding of the optimal policy are central issues of RL. The problem can be simplified if all possible future states are available and can be evaluated. This is our case: We do not need to evaluate the policy, nor to improve the policy. Instead, we evaluate all neighboring states of the actual state and move to (or download) the most promising one. We follow all of the links of a given document and evaluate the documents at the links. The action that we shall execute is the most 'promising' action; we shall move to the most promising next link. If the link is a document then we download that document. If the document contains the phrase 'call for papers' then the learning system incurs immediate reward of

1. If a downloaded document does not contain this phrase then there is cost (the immediate reward is negative) of -0.01. These numbers are rather arbitrary. The relative ratio between reward and cost and the magnitude of the parameter of the sigmoid function matters. These parameters influence learning capabilities. From this point of view our results may not be the best possible that can be achieved by this hybrid technology.

Value estimation makes use of the following upgrade

$$V^{+}(s_t) = V(s_t) + \alpha * (r_{t+1} + \gamma * V(s_{t+1}) - V(s_t)) \tag{25}$$

where α is the learning rate, $r_{t+1} \in \mathbb{R}$ is the immediate reward, $0 < \gamma < 1$ is the discount factor, and subscripts $t = 1, 2, \ldots$ indicate action number (i.e., time). An excellent introduction to value estimation by means of parameterized function approximators can be found in [110]. Concerning technicalities of RL, (a) we used eligibility traces, (b) we did not use of explorative steps because the environments can be very different, and (c) approximated the value function as

$$V(s) \approx \sum_{i=1}^{n} w_i \, \sigma(\text{output}_i) \tag{26}$$

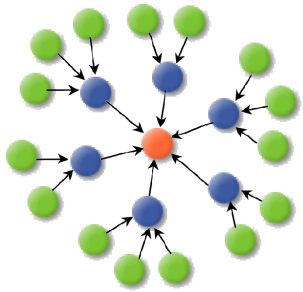

FIGURE 26. ***Context* of the document**
Document and its first and second 'neighbors'.

where $\sigma(.)$ denotes the sigmoid function acting on the linear outputs of the SVM classifiers, w_i is the weight (or relevance) of the i^{th} classifier determined by upgrade Eq. 25. If the quality of the upgrade is measured by the mean square error of the estimations then the following approximate weight upgrade can be derived by using Eq. 25 (see, e.g., [110] for details):

$$\Delta w_i = \alpha * (r_{t+1} + \gamma * V(s_{t+1}) - V(s_t)) * \sigma(\text{output}_i). \tag{27}$$

This approximate upgrade was used in our RL engine.

Features and learning to search

A target document and its environment are illustrated in Fig. 26. The basic assumption is that the environment of one one target document may provide good yardsticks for other target documents. A hand-waving argument can be given as follows. Documents are linked

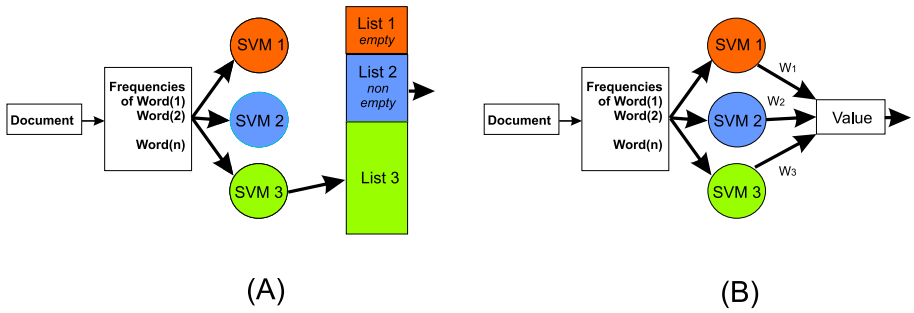

FIGURE 27. **SVM based document classifiers** (a) Classification of distance from document using SVM classifiers, (b) Value estimation based on SVM classifiers.

to each other. Links are made by those, for whom the document has value; who can make use of the target document. These links form the one-step *context* of the document. The one-step context, in turn, may be characteristic to the document. This is the idea behind CFC. The CFC [35] method develops classifiers of the environment. The k^{th} classifier is trained on documents k-steps away from known documents and it is supposed to say 'yes' if – according to its decision surface – there is a target document k-steps away from the actual site/document. If more than one classifier outputs 'yes' signal then the best classifier is considered. The CFC idea with SVM classifiers is shown in Fig. 27(a). The CFC maintains a list of visited links ordered according to the SVM classification. One of the links belonging to the best non-empty classifier is visited next.

CFC and RL: Learning to search

The CFC method has a prewired property, namely the strict ordering of contexts. This prewiring can be criticized on the following ground: (i) The number of documents that we have at our disposal could be small for training the CFC classifiers. (ii) The context might vary drastically from site to site and from time to time. In turn, classification could be weak.

Reinforcement learning offers a key here. If this prewiring is perfect, still, we could learn it. There is nothing to loose here, provided that learning is fast. Utilizing RL, we gain the adaptive property. If the prewiring is imperfect then proper weights will be derived by the learning algorithm.

The outputs of the SVMs can be saved. These outputs can be used to estimate the value of a document at any instant. In turn, the value based ordering of the documents requires minor computation and can be made at each step. This continuous reordering of the documents replaces the prewired ordering of the documents of the CFC method. The new architecture is shown in Fig. 27(b). In our studies 5 SVMs were used to classify up to the 5^{th} neighborhood. This number was limited to 5 because of the high branching ratio on the web. Smaller numbers are not needed for RL: RL can determine if a feature (the output of the SVM) is irrelevant or not. If it is irrelevant the weight

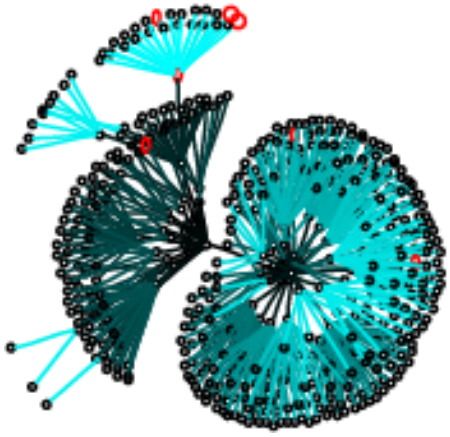

FIGURE 28. **Search pattern for breadth first crawler.** Search was launched from neutral site. A site is called neutral if there is very few target document in its environment. (For further details, see text.) Diameter of open circles is proportional to the number of target documents downloaded.

assigned by RL becomes zero. This effect can be seen in many cases in our computer studies.

Results and discussion

The CfP problem has been studied. Search pattern at the initial phase for the breadth first method is shown in Fig. 28.

Search patterns for the context focused crawler and the crawler using RL based value estimation are shown in Fig. 16 and Fig. 16. The launching site of these searches was a 'neutral site', a relatively large site containing few CfP documents (`http://www.inf.elte.hu`). We consider this type of launching important for web crawling, it simulates the case when mail lists are not

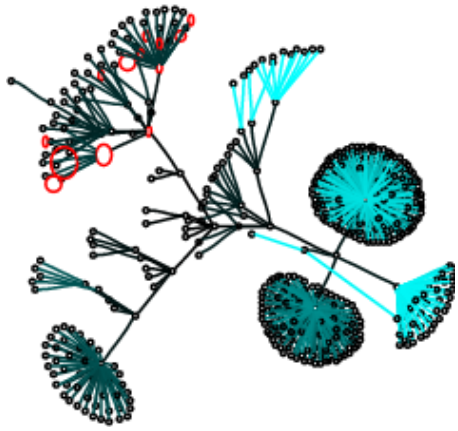

FIGURE 29. **Search pattern for context focused crawler.**
Search was launched from neutral site. Diameter of open circles is proportional to the number of target documents downloaded.

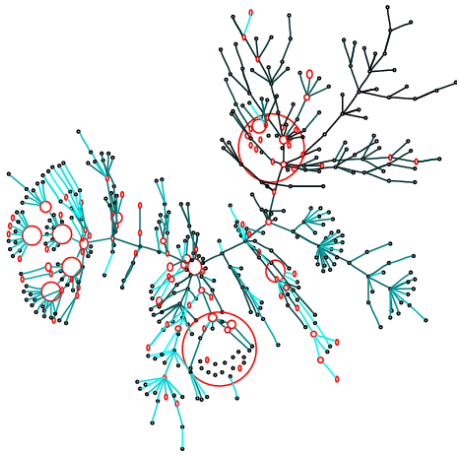

FIGURE 30. **Search pattern for CFC *and* reinforcement learning**
Search was launched from neutral site. Diameter of open circles is proportional to the number of target documents downloaded.

available (to us) and traditional search engines are not satisfactory. Results indicate that yes, this site has a minor number of target documents in the neighborhood.

'Scale' of Figs. (16,16) differs from that of Fig. 28. 'True surfed scale' would be reflected by normalizing to edge thickness. Radius of open circles is proportional to the number of downloaded target documents. The CFC is somewhat better in the initial phase than the breadth first method. Later portions of search history show that CFC becomes considerably better than the breadth first method.

Quantitative comparisons are shown in Fig. 16. According to the figure, upon downloading 20,000 documents, the number of hits were about 50, 200, and 1000 for the breadth first, the CFC and CFC based RL crawlers, respectively. These launches were conducted about the same time. We shall demonstrate that the large difference between CFC and CFC based RL method is mostly due to the adaptive properties of the RL crawler.

There are two site types that have been investigated. The first site type is the neutral site that has been described before, the second site is a mail server on conferences. Target documents were rewarded by '1 unit', whereas downloaded and non-target documents were punished by 0.01 unit. Also, for some examples there are runs separated by one month (March, 2001). Most summer conferences made announcements during this month. It is important to make a distinction between the

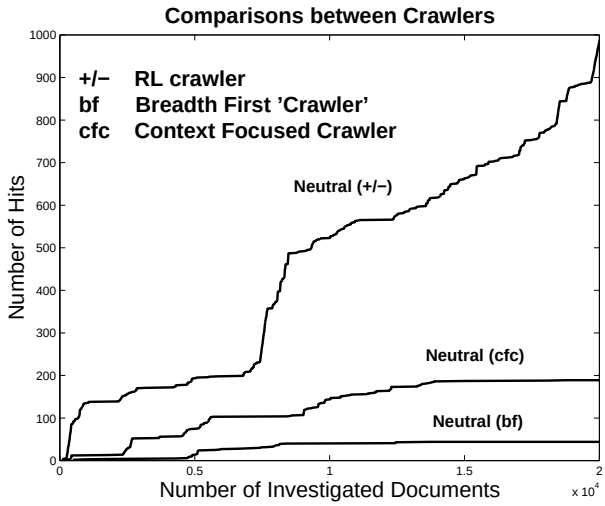

FIGURE 31. **Results of breadth first, CFC and CFC based RL methods.**

initial phase of the search and the later phases. The initial phase of the search (the first 200 downloaded documents) are shown in Fig. 32.

According to this figure downloading is very efficient from the mail server site in each occasion. The non-adapting crawler utilizing averaged weights is superior to all the other crawlers that are learning – almost all downloaded documents are hits. Close to this site there are many relevant documents and the 'breadth first crawler' is also efficient here. Nevertheless, the non-adapting CFC crawler outperforms the breadth first crawler in this domain. Launching from neutral sites is inefficient at this early phase. Breadth first method finds no hit close to the neutral site (not shown in the figure). Middle phase of the search is shown in Fig. 33.

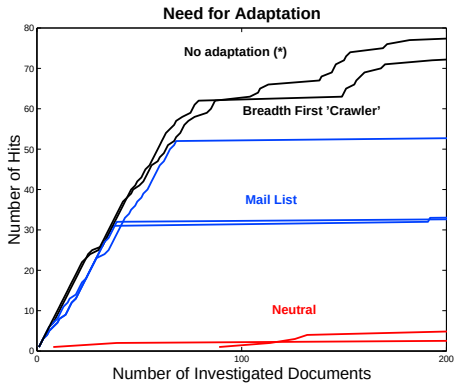

FIGURE 32. **Comparisons between 'neutral' and mail server sites in the initial phase.** Reward and punishment are given in the legend of the figure. Differences between similar types are due to differences in launching time. The largest time difference between similar types is one month. Neutral site (thin lines): `http://www.inf.elte.hu`. Mail list (thick lines): `http://www.newcastle.research.ec.org/cabernet/events/msg00043.html`. Search with 'no adaptation' (dotted line) was launched from mail list and used average weights from another search that was launched from the same place.

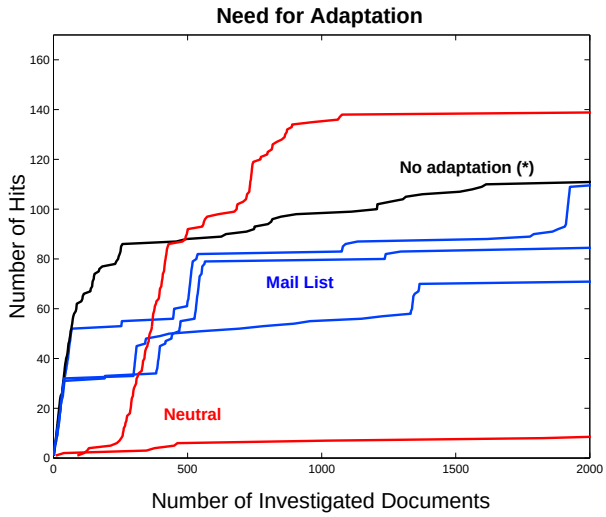

FIGURE 33. **Comparisons between 'neutral' and mail server sites up to 2000 documents.** Same conditions as in Fig. 32

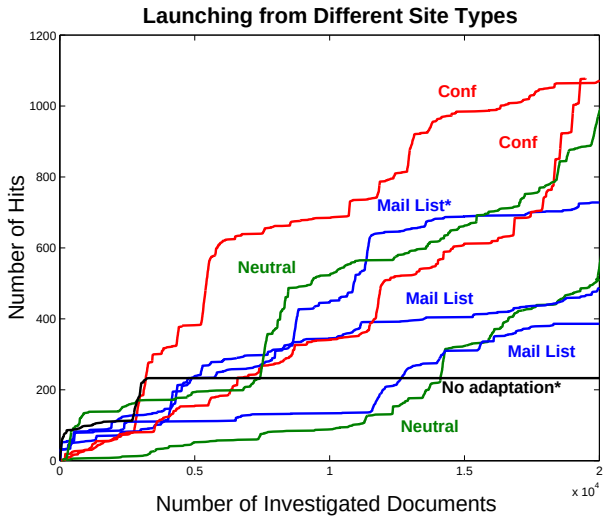

FIGURE 34. **Comparisons between different sites up to 20,000 documents.** Same conditions as in Figs. 32 and 33. Search with 'no adaptation' used average weights from another search that was launched from the same place (denoted by $*$)

Middle phase has a somewhat different message. Sometimes, launches from the neutral site may become very successful, and search without adaptation is still competitive. Launches from the mail list look about the same.

Search results up to 20,000 documents are shown in Fig. 34. We note, that the 2,000 to 20,000 range can be a typical one for the number of documents found by search engines.

This graph contains results from a subset of the runs that we have executed. These runs were launched from different sites; the neutral site and the mail list, as well as a third type, the 'conference' site: `http:// www.informatik.uni-freiburg.de/ index.en.html`.

This site is known to be involved in organizing conferences. Adapting crawlers collected a large number of documents from all site types and during the whole time region. The time region covered by these examples was one month. The rate of collection was between 2%-5%. In contrast, the collection rate is close to 100% for the non-adaptive CFC launched from the mail list site and it is better than the breadth first method here. Lack of adaptation, however, prohibits this crawler to find new target documents in cca. 17,000 downloads in spite of the fact that in the early phase this crawler was the most successful and that other documents exist on the web. Let us summarize these points:

(1) Identical conditions give rise to very different results one month later.

(2) Starting from a neutral site can be as effective as starting from a mailing list.

(3) The lack of adaptation is a serious drawback even if the crawler is launched from a mailing list.

The importance of adaptation is also demonstrated by the RL weights assigned during search. These weights are shown in the following figures. Figure 35 depicts the weights belonging to the different SVMs. At the beginning of the search the weights are almost perfectly ordered; the largest weight is given to the SVM that predicts relevant document

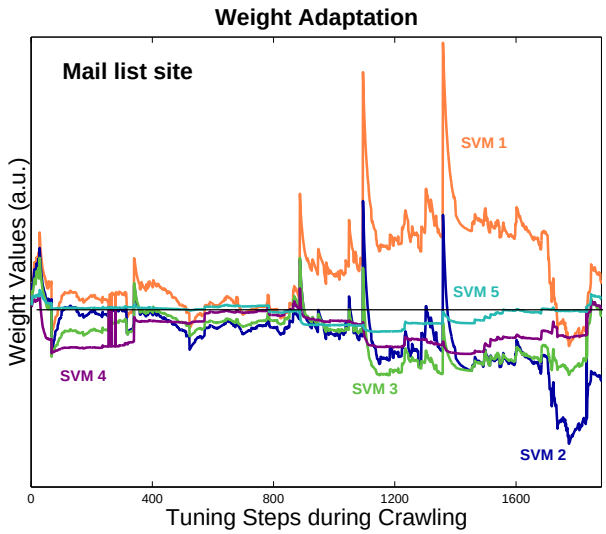

FIGURE 35. **Change of weights of SVMs in value estimation for mail site.**

'one step away' whereas the 4^{th} and the 5^{th} SVMs have the smallest weights. That is, RL 'pays attention' to the first SVM and pays less attention to the others. This order changes as time goes on. There are regions (around 800 on the horizontal axis) where most attention is paid to the fifth SVM and smaller attention is paid to the others. The order of importance changes again when a rich region is found; the importance of the first SVM recovers quickly and, in turn, crawling is dominated by the weight of the first SVM.

'Weight history' is different at the neutral site (Fig. 36). Up to about 100 downloads very few relevant documents were found at this site. The value of weight of the 5^{th} SVM is slightly positive, whereas that

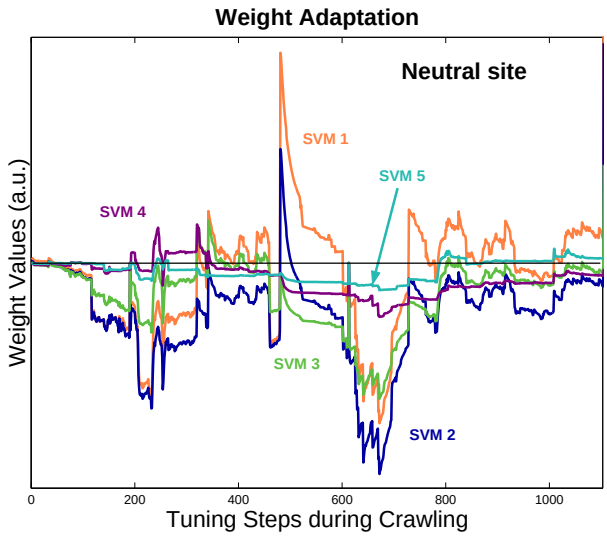

FIGURE 36. **Change of weights of SVMs in value estimation for 'neutral' site.**

of the others are negative. The 1^{st} and the 2^{nd} SVMs are considered the 'worst', the weights belonging to these classifiers are large negative numbers. Situation changes quickly when a rich region is found. It is typical that the weight of the 5^{th} SVM is ranked second. That is, the ordering of the neighborhoods – which was used for the training of the SVMs – can be misleading in other parts of the internet. The information contained by the 'context' (the environment) of the nodes can be used, if the values for these classifiers are estimated and corrected on-line.

Conclusions

We have suggested a novel method for web search. The method makes use of combinations of two popular AI techniques, support vector machines (SVM) and reinforcement learning (RL). The method has a few adapting parameters that can be optimized during the search. This parameterization helps the crawler that can adapt to different parts of the web. The value estimation used the SVM classifiers are in linear mode: The outputs of the SVMs, together, formed a set of yardsticks for the estimation of the distance of the actual site from target documents. Preliminary studies indicate that SVM with CfP BoWs performs well on very different problems, like searching for documents published recently about books on Harry Potter. The point is that the few RL parameters can be trained by rewarding for target documents quickly. There are many ways one may try to improve our method. RL has many different formulations all of which could be applied here. Most promising are the approaches that can take into account (many) different criteria in the search objective [42, 44, 38]. Alas, RL methods are capable of extracting features [114] that may improve the quality of the search during the 'flight'.

17. Appendix E: List of software components

The software have several components from two different groups. The first group is made of the Matlab software. This group is either from the internet (the Bayes Network Toolbox) or has been developed by us.

(1) Bayes Network Toolbox: Kevin Murphy's toolbox for Matlab. It must be in the Matlab's path.
(2) Learntree: Our Probabilistic Tree Model and Mixture of Trees software. It must be in the Matlab's path.
(3) Tests: Matlab software used to produce the results in Tech Report
(4) Data: The datasets used in our tests

The other group contains JAVA components.

(1) TreeBuilder JAVA class
(2) Probability JAVA class
(3) XML2Tree class

These components can be found on the CD-ROM. Both components are commented by their own comment system, Matlab has its help system using the header of the files, whereas we used Javadoc for commenting JAVA routines.

References

1. *Exslt*, http://www.jenitennison.com/xslt/exslt/.
2. *Fxpath*, http://www.pantor.com/fxpath/.
3. *Schematron*, http://www.ascc.net/xml/resource/schematron/schematron.html.
4. *Xml schema datatypes*, http://www.w3.org/TR/xmlschema-2/.
5. *Xml schema primer*, http://www.w3.org/TR/xmlschema-0/.
6. *Xml schema structures*, http://www.w3.org/TR/xmlschema-1/.
7. *Xml schema xerces-j*, Xerces-J:http://xml.apache.org/xerces-j/schema.html,.
8. *Xquery*, http://www.w3.org/TR/xquery/.
9. *Xsl and xpath*, http://www.w3.org/TR/xslt, http://www.w3.org/TR/xpath.
10. R. Albert, H. Jeong, and A.-L. Barabási, *Diameter of the world-wide web*, Nature **401** (1999), 130–131.
11. S.L Amari, A. Cichocki, and H.H. Yang, *A new learning algorithm for blind signal separation.*, Advances in Neural Information Processing Systems, Morgan Kaufmann, San Mateo, CA, 1996, pp. 757–763.
12. F. Attneave, *Some informational aspects of visual perception*, Psychological Review **61** (1954), 183–193.
13. D. Baker, T. Hofmann, A. McCallum, and Y. Yang, *A hierarchical probabilistic model for novelty detection in text*, ICML-99, 1999, http: //www.cs.cmu.edu /People /mccallum /papers /tdt-icml99s.ps.
14. H.B. Barlow, *Sensory communication*, w.a. rosenblith ed., pp. 217–234, MIT Press, Cambridge, MA, 1961.
15. A.J. Bell and T.J. Sejnowski, *An information-maximization approach to blind separation and blind deconvolution*, Neural Computation **7** (1995), 1129–1159.
16. ———, *An information-maximization approach to blind separation and blind deconvolution*, Neural Computation **7** (1995), 1129–1159.

17. D. P. Bertsekas, *Network optimization: Continuous and discrete models*, Optimization and Neural Computation Series, Athena Scientific, Belmont, Massachusetts, 1998.

18. D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-dynamic programming*, Athena Scietific, Belmont, MA, 1996.

19. A. Blum and J. Mitchell, *Combining labeled and unlabeled data with co-training*, COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers, 1998, `http: //www.cs.cmu.edu /afs/cs.cmu.edu /project/theo-11 /www/wwkb /colt98_final.ps`.

20. J. A. Boyan, *Learning evaluation functions for global optimization*, Ph.D. thesis, School of Computer Science Carnagie Mellon University, Pittsburgh PA 15213, 1998.

21. J. A. Boyan and A. W. Moore, *Learning evaluation function for global optimization and boolean satisfiability*, In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI), 1998, p. 14.

22. J.A. Boyan and A.W. Moore, *Learning evaluation functions to improve optimization by local search*, J. of Machine Learning Journal **1** (2000), 77–122.

23. S. Chakrabarti, B.E. Dom, and P. Indyk, *Enhanced hypertext categorization using hyperlinks*, Proc. SIGMOD-98, ACM Int. Conf. on Management of Data (Seattle, US) (Laura M. Haas and Ashutosh Tiwary, eds.), ACM Press, New York, US, 1998, `http://www.almaden.ibm.com/cs/k53/irpapers/sigmod98.ps`, pp. 307–318.

24. S. Chakrabarti, D.A. Gibson, and K.S. McCurley, *Surfing the Web backwards*, 8th World Wide Web Conference (Toronto, Canada), 1999, `http: //www8.org/w8-papers /5b-hypertext-media /surfing/`.

25. S. Chakrabarti, M. van der Berg, and B. Dom, *Focused crawling: a new approach to topic-specific Web resource discovery*, 8th International World Wide Web Conference (WWW8), 1999, `http: //www.cs.berkeley.edu / soumen/doc /www1999f/pdf /www1999f.pdf`.

26. D. Charles and C. Fyfe, *Modelling multiple cause structure using rectification constraints*, Network: Computations in Neural Systems **9** (1998), 167–182.

27. J. Cho, H. García-Molina, and L. Page, *Efficient crawling through URL ordering*, Computer Networks and ISDN Systems **30** (1998), no. 1–7, 161–172, `http://www-db.stanford.edu/pub/papers/efficient-crawling.ps`.

28. C. K. Chow and C. N. Liu, *Approximating discrete probability distributions with dependence trees*, IEEE Transactions on Information Theory **IT-14(3)** (1968), 462–467.

29. P. Comon, *Independent component analysis - A new concept?*, Signal Processing **36** (1994), 287–314.

30. T. Cover and J. Thomas, *Elements of information theory*, John Wiley and Sons, New York, USA, 1991.

31. editor D. S. Hochbaum, *Approximation algorithms for np-hard problems*, PWS Publishing Co., Boston, 1995.

32. P. Dayan and G. E. Hinton, *Feudal reinforcement learning*, Advances in Neural Information Processing Systems (San Mateo, CA), vol. 5, Morgan Kaufmann, 1993, pp. 271–278.

33. J. Dean and M.R. Henzinger, *Finding related pages in the world wide web*, WWW8 / Computer Networks **31** (1999), no. 11-16, 1467–1479, `http://www.research.compaq.com/SRC/personal/monika/papers/monika-www8-1.ps.gz`.

34. T. G. Dietterich, *Hierarchical reinforcement learning with the MAXQ value function decomposition*, Journal of Artificial Intelligence Research **13** (2000), 227–303.

35. M. Diligenti, F. Coetzee, S. Lawrence, C. Lee Giles, and M. Gori, *Focused crawling using context graphs*, 26th International Conference on Very Large Databases, VLDB 2000 (Cairo, Egypt), 10–14 September 2000, `http://www.neci.nec.com/lawrence/papers/focus-vldb00/focus-vldb00.ps.gz`.

36. S. Dominich, *A unified mathematical definition of classical information retrieval*, Journal of the American Society for Information Science **51** (2000), no. 7, 614–624.

37. D.W. Dong and J.J. Atick, *Temporal decorrelation: a theory of lagged and nonlagged responses in the lateral geniculate nucleus*, Network **6** (1995), 159–178.

38. D. Dubois, M. Grabisch, F. Modave, and H. Prade, *Relating decision under uncertainty and multicriteria decision making models*, International Journal of Intelligent Systems **15** (2000), 967–979.

39. S. Dumais, J. Platt, D. Heckerman, and M. Sahami, *Inductive learning algorithms and representations for text categorization*, 7th International Conference on Information and Knowledge Management, ICIKM'98, ACM, 1998, `http://robotics.stanford.edu/users/sahami/papers-dir/cikm98.pdf`.

40. M. Feder and N. Merhav, *Hierarchical universal coding*, IEEE Transactions in Information Theory **IT-42** (1996), 1354–1364.

41. D.J. Field, *What is the goal of sensory coding?*, Neural Computation **6** (1994), 559–601.

42. N. M. Fraser and J. W. Hauge, *Multicriteria approval: application of approval voting concepts to mcdm problems*, Journal of Multi-Criteria Decision Analysis **7** (1998), 263–273.

43. Michael L. Fredman and Robert Endre Tarjan, *Fibonacci heaps and their uses in improved network optimization algorithms*, Journal of the Association for Computing Machinery **34** (1987), 596–615.

44. Z. Gábor, Z. Kalmár, and C. Szepesvári, *Multi-criteria reinforcement learning*, Proceedings of the Fifteenth International Conference on Machine Learning, 1998.

45. T. Spencer H. N. Gabow, Z. Galil and Robert Endre Tarjan, *Efficient algorithms for finding minimum spanning trees in undirected and directed graphs*, Combinatorica **6** (1986), 109–122.

46. S. Haykin, *Neural networks: A comprehensive foundation*, Prentice Hall, New Jersey, USA, 1999.

47. G.E. Hinton, P. Dayan, B.J. Frey, and R.M. Neal, *The "wake-sleep" algorithm for unsupervised neural networks*, Science **268** (1995), 1158–1161.

48. G.E. Hinton and Z. Ghahramani, *Generative models for discovering sparse distributed representations*, Philosophical Transactions of the Royal Society B **352** (1997), 1177–1190.

49. G.E. Hinton and R.S. Zemel, *Autoencoders, minimum description length and Helmholtz free energy*, Advances in Neural Processing Systems (J.D. Cowan, G. Tesauro, and J. Alspector, eds.), vol. 6, Morgan Kaufmann, San Mateo, CA, 1994, pp. 3–10.

50. T. Hofmann, *Probabilistic latent semantic analysis*, `http: //www.icsi.berkeley.edu /~hofmann /Papers /Hofmann-UAI99.ps`, 1999.

51. ———, *Learning the similarity of documents: An information-geometric approach to document retrieval and categorization*, Neural Information Processing Systems, vol. 12, MIT Press, 2000, pp. 914–920.

52. B.K.P. Horn, *Understanding image intensities*, Artificial Intelligence **8** (1977), 201–231.

53. A. Hyvärinen, *Independent component analysis for time-dependent stochastic processes*, Proc. Int. Conf. on Artificial Neural Networks (ICANN'98) (Skvde, Sweden), 1998, pp. 541–546.

54. ———, *Sparse code shrinkage: Denoising of nongaussian data by maximum likelihood estimation*, Neural Computation **11** (1999), 1739–1768.

55. ———, *Survey on independent component analysis*, Neural Computing Surveys **2** (1999), 94–128.

56. A. Hyvärinen, P. Hoyer, and E. Oja, *Sparse code shrinkage: Denoising by nonlinear maximum likelihood estimation*, Advances in Neural Information Processing Systems 11 (NIPS*98), MIT Press, 1999, pp. 1739–1768.

57. Rick Jelliffe, *Using xslt as a validation language*, `http://www.ascc.net/xml/en/utf-8/XSLvalidation.html`.

58. T. Joachims, *Estimating the generalization performance of an SVM efficiently*, Proc. 17th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA, 2000, `http:// www-ai.informatik.uni-dortmund.de /DOKUMENTE /joachims_99e.ps.gz`, pp. 431–438.

59. C. Jutten and J. Herault, *Blind separation of sources, Part I: An adaptive algorithm based on neuromimetic architecture*, Signal Processing **24** (1991), 1–10.

60. ———, *Blind separation of sources, Part I: An adaptive algorithm based on neuromimetic architecture*, Signal Processing **24** (1991), 1–10.

61. A. Kabán and M. Girolami, *Clustering of text documents by skewness maximisation*, 2'nd International Workshop on Independent Component Analysis and Blind Source Separation, ICA'2000 (Helsinki), 2000, pp. 435 – 440.

62. L. P. Kaelbling, *Hierarchical learning in stochastic domains: Preliminary results.*, Proceedings of the Tenth International Conference on Machine Learning (San Mateo, CA), Morgan Kaufmann, 1993.

63. Z. Kalmár, C. Szepesvári, and A. Lőrincz, *Generalized dynamic concept model as a route to construct adaptive autonomous agents*, Neural Network World **5** (1995), 353–360.

64. Z. Kalmár, C. Szepesvári, and A. Lőrincz, *Module-based reinforcement learning: Experiments with a real robot*, Machine Learning **31** (1998), 55–85.

65. J. Karhunen, E. Oja, L. Wang, R. Vigario, and J. Joutsensalo, *A class of neural networks for independent component analysis*, IEEE Trans. on Neural Networks **8** (1997), 487–504.

66. S. Kaski, *Dimensionality reduction by random mapping: Fast similarity computation for clustering*, Proc. of Int. Joint Conf. on Neural Networks, vol. 1, IEEE Service Center, Piscataway, NJ, 1998, `http://websom.hut.fi/websom/doc/ps/kaski98ijcnn.ps.gz`, pp. 413–418.

67. S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murthy, *Improvements to Platt's SMO Algorithm for SVM Classifier Design*, Tech. Report CD-99-14, Dept. of Mechanical and Production Engineering, National University of Singapore, 1999, `http://guppy.mpe.nus.edu.sg /~mpessk/smo_mod.ps.gz`.

68. C. Koch and T. Poggio, *Predicting the visual world: Silence is golden*, Nature Neuroscience **2** (1999), 9–10.

69. T. Kolenda, L. Hansen, and S. Sigurdsson, *Independent components in text*, Ed.: M. Girolami, `http: //eivind.imm.dtu.dk /publications /1999 /kolenda.nips99.ps.gz`, 2000.

70. D. Koller and M. Sahami, *Hierarchically classifying documents using very few words*, 14th International Conference on Machine Learning ICML97, 1997, `http://www-diglib.stanford.edu/diglib/WP/PUBLIC/DOC130.ps`, pp. 170–178.

71. R.K. Kolluri, N. Mittal, R.P. Ventakachalam, and N. Widjaja, *Focussed crawling*, 2000, `http: //www.cs.utexas.edu /users /ramki /datamining /fcrawl.ps.gz`.

72. R. E. Korf, *Learning to solve problems by searching for macro-operators*, Pitman Publishers, Boston, 1985.

73. S. Kullbach and R. A. Leibler, *On information and sufficiency*, Ann. Math. Stat. **22** (1951), 79–86.

74. S. Lawrence, *Context in web search*, IEEE Data Engineering Bulletin **23** (2000), no. 3, 25–32.

75. D. Lee and W. Chu, *Comparative analysis of six xml schema languages*, `http://www.cobase.cs.ucla.edu/tech-docs/dongwon/ucla-200008.html`.

76. D.D. Lee and H.S. Seung, *Learning the parts of objects by non-negative matrix factorization*, Nature **401** (1999), 788–791.

77. ———, *Algorithms for non-negative matrix factorization*, Advances in Neural Processing Systems, vol. 13, Morgan Kaufmann, San Mateo, CA, 2001, accepted.

78. A. Lőrincz, I. Pólik, and I. Szita, *Event-learning and robust policy heuristics*, Tech. Report NIPG-ELU-15-05-2001, ELTE, 2001, http://people.inf.elte.hu/lorincz/Files/NIPG-ELU-14-05-2001.pdf.

79. M. L. Littman, A. Cassandra, and L. P. Kaelbling, *Learning policies for partially observable environments: Scaling up*, Proceedings of the Twelfth International Conference on Machine Learning (A. Prieditis and S. Russell, eds.), Morgan Kaufmann, 1995.

80. A. Lőrincz, *Forming independent components via temporal locking of reconstruction architectures: A functional model of the hippocampus*, Biological Cybernetics **79** (1998), 263–275.

81. A. Lőrincz and Gy. Buzsáki, *The parahippocampal region: Implications for neurological and psychiatric dieseases*, Annals of the New York Academy of Sciences (H.E. Scharfman, M.P. Witter, and R. Schwarz, eds.), vol. 911, New York Academy of Sciences, New York, 2000, pp. 83–111.

82. S. Mahadevan and J. Connell, *Automatic programming of behavior-based robots using reinforcement learning*, Artificial Intelligence **55** (1992), 311–365.

83. Marina-Meila-Predoviciu, *Learning with mixtures of trees*, Ph.D. thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, 1999.

84. M.J. Mataric, *Behavior-based control: Examples from navigation, learning, and group behavior*, J. of Experimental and Theoretical Artificial Intelligence **9** (1997), 2–3.

85. S.-I. Amari, *Natural gradient works efficiently in learning*, Neural Computation **10** (1998), 251–276.

86. Zs. Palotai, T. Kandár, Z. Mohr, T. Visegrády, G. Ziegler, P. Arató, and A. Lőrincz, *Value prediction in hls allocation problems using intellectual properties*, Applied Artificial Intelligence, accepted for publication.

87. A. McCallum, K. Nigam, J. Rennie, and K. Seymore, *Building domain-specific search engines with machine learning techniques*, AAAI-99 Spring Symposium on Intelligent Agents in Cyberspace, 1999, http: //www.cs.cmu.edu/~mccallum /papers /cora-aaaiss98.ps.

88. ———, *Automating the construction of internet portals with machine learning*, Information Retrieval **3** (2000), no. 2, 127–163, http: //www.cs.cmu.edu /afs /cs /user /kseymore /html /papers / cora-journal.ps.gz.

89. A.K. McCallum, *Bow: A toolkit for statistical language modelling, text retrieval, classification and clustering*, http://www.cs.cmu.edu/~mccallum/bow, 1996.

90. ———, *Multi-label text classification with a mixture model trained by em*, 1999, http: //www.cs.cmu.edu/~mccallum /papers /multilabel-nips99s.ps.gz.

91. A.K. McCallum, K. Nigam, and L.H. Ungar, *Efficient clustering of high-dimensional data sets with application to reference matching*, Knowledge Discovery and Data Mining, 2000, http: //www.cs.cmu.edu /People /knigam /papers /canopy-kdd00.pdf, pp. 169–178.

92. N. Merhav and M. Feder, *Universal prediction*, IEEE Trans. Inform. Theory. **IT-44** (1998), 2124–2147.

93. S. Minton, *Learning search control knowledge: An explanation based approach*, Kluwer Academic, 1988.

94. T. Mitchell, *The role of unlabeled data in supervised learning*, Proceedings of the Sixth International Colloquium on Cognitive Science, 1999, http://www.ri.cmu.edu/pub_files/pub1/mitchell_tom_1999_2/ mitchell_tom_1999_2.pdf.

95. S. Mukherjea, *WTMS: A system for collecting and analyzing topic-specific web information*, 9th World Wide Web Conference, 2000, http://www9.org/w9cdrom/293/293.html.

96. J.W. Murdock and A.K. Goel, *Towards adaptive web agents*, 14th IEEE International Conference on Automated Software Engineering, October 1999, http: //www.cc.gatech.edu/morale /papers/ase99.ps.

97. K. Nigam, A.K. Mccallum, S. Thrun, and T. Mitchell, *Text classification from labeled and unlabeled documents using EM*, Machine Learning **39** (2000), no. 2/3, 103–134, http://www.cs.cmu.edu/~knigam/papers/emcat-mlj99.ps.gz.

98. L. Parra, G. Deco, and S. Miesbach, *Statistical independence and novelty detection with information preserving nonlinear maps*, Neural Computation **8** (1995), no. 2, 260–269.

99. R.P.N. Rao and D.H. Ballard, *Dynamic model of visual recognition predicts neural response properties in the visual cortex*, Neural Computation **9** (1997), 721–763.

100. ———, *Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects*, Nature Neuroscience **2** (1999), 79–87.

101. J. Rennie, K. Nigam, and A. McCallum, *Using reinforcement learning to spider the web efficiently*, 16th International Conference on Machine Learning, Morgan Kaufmann, San Francisco, CA, 1999, `http: //www.cs.cmu.edu /~mccallum /papers /rlspider-icml99s.ps.gz`, pp. 335–343.

102. G. A. Rummery and M. Niranjan, *On-line q-learning using connectionist systems*, Tech. report, Cambridge University Engineering Department, 1996.

103. J. Schmidhuber, *Neural sequence chunkers*, Technical Report FKI-148-91, Technische Universitat Munchen, Munchen, Germany, 1991.

104. C.E. Shannon, *A mathematical theory of communication*, AT & T Bell Labs. Tech. J. **27** (1948), 397–423.

105. R. Sibson, *Slink: an optimally efficient algorithm for the single link cluster method*, The Computer Journal (1973), 16.

106. S. Singh, T. Jaakkola, M. L. Littman, and Cs. Szepesvári, *Convergence results for single-step on-policy reinforcement-learning algorithms*, Machine Learning **38** (2000), 287–303.

107. Alex J. Smola and Bernhard Schlkopf, *A tutorial on Support Vector Regression*, Tech. Report NC2-TR-1998-030, NeuroCOLT2, 1998.

108. R. Sutton, *Learning to predict by the method of temporal differences*, Machine Learning **3** (1988), 9–44.

109. R. Sutton, S. Singh, D. Precup, and B. Ravindran, *Improved switching among temporally abstract actions*, Advances in Neural Information Processing Systems **11** (1999), 1066–1072.

110. R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.

111. Cs. Szepesvári, *Static and dynamic aspects of optimal sequential decision making*, Phd thesis, Attila József University, Bolyai Institute of Mathematics, 1998.

112. Robert Endre Tarjan, *Data structures nad network algorithms*, Society for Industrial and Applied Mathematics (1983).

113. Charles E. Leiserson Thomas H. Cormen and Ronald R. Rivest, *Introduction to algorithms*, MIT Press, 1990.

114. T. Thrun and A. Schwartz, *Finding structure in reinforcement learning*, Advances in Neural Information Processing Systems (San Mateo, CA), vol. 7, Morgan Kaufmann, 1995.

115. V. Vapnik, *The nature of statistical learning theory*, Springer Verlag, New York, 1995.

116. A. Vinokourov and M. Girolami, *A probabilistic hierarchical clustering method for organizing collections of text documents*, 15th Int. Conf. on Pattern Recognition, vol. 2, IEEE Computer Press, 2000, `http://cis.paisley.ac.uk/vino-ci0/vinokourov_ICPR00.ps`, pp. 182–185.

117. ———, *A probabilistic hierarchical clustering method for organizing collections of text documents*, Tech. Report ISSN 1461-6122, University of Paisley, Department of Computing and Information Systems, Paisley, PA1 2BE, UK, March 2000, `http://cis.paisley.ac.uk/vino-ci0/hplsa_kais.ps`.

118. ———, *Symmetric and asymmetric hierarchical latent class analysis for organizing collections of text documents*, 2000.

119. C.J.C.H. Watkins, *Learning from delayed rewards*, Phd thesis, King's College, Cambridge, UK, 1989.

120. Q. Xie and A.R. Barron, *Asymptotic mimimax regret for data compression, gambling, and prediction*, IEEE Transactions in Information Theory **IT-46** (2000), 431–445.

121. G. Ziegler, Zs. Palotai, T. Cinkler, P. Arató, and A. Lőrincz, *Value prediction in engineering applications*, Engineering of Intelligent Systems. Proceedings of AIE/IEA 2001, Budapest Hungary (L. Monostori, J. Váncza, and M. Ali, eds.), LNAI, vol. 2070, Springer, June 4-7 2001, ISBN 3-540-42219-6, pp. 25–34.