THE 18TH EUROPEAN CONFERENCE ON MACHINE LEARNING
AND
THE 11TH EUROPEAN CONFERENCE ON PRINCIPLES AND PRACTICE
OF KNOWLEDGE DISCOVERY IN DATABASES

PROCEEDINGS OF THE WORKSHOPS:

PRIOR CONCEPTUAL KNOWLEDGE IN MACHINE LEARNING AND DATA MINING AND WEB MINING 2.0

PriCKL'07 & Web Mining 2.0

September 21, 2007

Warsaw, Poland

Editors:
Bettina Berendt
Institute of Information Systems, Humboldt University Berlin, Germany
Dunja Mladenič
J. Stefan Institute, Ljubliana, Slovenia
Giovanni Semeraro
Department of Informatics, University of Bari, Italy
Myra Spiliopoulou
Faculty of Computer Science, Otto-von-Guericke-Univ. Magdeburg, Germany
Gerd Stumme
Knowledge and Data Engineering Group, University of Kassel, Germany
Vojtěch Svátek
University of Economics, Prague, Czech Republic
Filip Železný
Czech Technical University, Prague, Czech Republic

Typesetting:
Bettina Berendt

Preface

Introduction: PriCKL and Web Mining 2.0

This proceedings volume comprises the papers of two workshops held at ECML/PKDD 2007: PriCKL (Prior Conceptual Knowledge in Machine Learning and Knowledge Discovery) and Web Mining 2.0. Our own prior work (including the joint proceedings volume *Semantics, Web and Mining, Joint International Workshops, EWMF 2005 and KDO 2005* published as Springer LNAI 4289) made us expect that there would be areas of joint interest in these two workshops. This year's accepted papers and the topics of the invited talks corroborated this expectation. We therefore decided to partially merge the workshops themselves (by a joint session) and to fully merge the proceedings.

PriCKL : There is general agreement that the quality of ML and KDD output strongly depends not only on the quality of source data and sophistication of learning algorithms, but also on additional, task/domain specific input provided by domain experts for the particular session. There is however less agreement on whether, when and how such input can and should effectively be formalised and reused as explicit prior knowledge. In this workshop, we aimed to investigate current developments and new insights on learning techniques that exploit prior knowledge and on promising application areas. With respect to application areas, we invited – and received – in particular papers on bioinformatics / medical and Web data environments.

The workshop is part of the activities of the "SEVENPRO – Semantic Virtual Engineering for Product Design" project of the European 6th Framework Programme.

Web Mining 2.0 : The workshop "Web Mining 2.0" has been motivated by the specification of Web 2.0. We observe Web 2.0 as a powerful means of promoting the Web as a social medium, stimulating interpersonal communication and fostering the sharing of content, information, semantics and knowledge among people. The workshop hosts research on the role of web mining in and for the Web 2.0.

The workshop is part of the activities of the working groups "Ubiquitous Data – Interaction and Data Collection" and "Human Computer Interaction and Cognitive Modelling" of the Coordination Action "KDubiq – Knowledge Discovery in Ubiquitous Environments" of the European 6th Framework Programme.

PriCKL sessions

The contributions to PriCKL fell into four groups. The first three were ILP/MRDM and an application focus on bioinformatics; the role of the human user; and investigations of fully automated methods of integrating background knowledge. The last group focused on the use of background knowledge for Web mining; these papers were presented in the joint session of PriCKL and WebMining 2.0.

An overview of both the application area bioinformatics and computational techniques used in it was given by our first Invited Speaker, Stephen Muggleton, in his

talk on *Using prior knowledge in biological pathway discovery*. An important group of techniques (not only) in this domain are ILP/MRDM methods. *On Ontologies as Prior Conceptual Knowledge in Inductive Logic Programming* by Francesca A. Lisi and Floriana Esposito provides an overview of Inductive Logic Programming attempts at using Ontologies as prior conceptual knowledge. Specifically, they compare the proposals CARIN-ALN and AL-log. *Using Taxonomic Background Knowledge in Propositionalization and Rule Learning* by Monika Žáková and Filip Železný exploit explicit term and predicate taxonomies to improve relational learning. They speed up the process of propositionalization of relational data substantially, by exploiting such ontologies through a novel refinement operator used in the construction of conjunctive relational features. Subsequent search is also shown to profit from the taxonomic background knowledge.

Two contributions emphasize the role of the human expert in contributing background knowledge for data mining quality: In *A Knowledge-Intensive Approach for Semi-Automatic Causal Subgroup Discovery*, Martin Atzmueller and Frank Puppe present a method for identifying causal relations between subgroups to form a network of links between subgroups. Background knowledge is used to add missing links in this network, correct directionality, and remove wrong links. Their approach is semi-automatic: the network and the relations are visualized to allow a user to accept them into a final causal model or not. An example case study illlustrates how data mining can help to identify risk factors for medical conditions. In *Evaluation of GUHA Mining with Background Knowledge*, Martin Ralbovský evaluates results of the GUHA method (General Unary Hypotheses Automaton), one of the oldest data mining methods for hypothesis generation, against rules formulated by human domain experts. The rules concern relationships between health-related behavioural variables. He concludes that the semantics of the quantifiers need to be worked on, and the default quantitative parameters adjusted to the domain.

Fully automated uses of background knoweldge and their advantages, including speed and accuracy, are investigated with respect to different types of machine learning in four contributions. *A study of the SEMINTEC approach to frequent pattern mining* by Joanna Józefowska, Agnieszka Lawrynowicz, and Tomasz Lukaszewski describes an experimental investigation of various settings under an approach to frequent pattern mining in description logics (DL) knowledge bases. Background knowledge is used to prune redundant (partial) patterns, which substantially speeds up pattern discovery. *Quantitative association rule mining in genomics using apriori knowledge* by Filip Karel and Jiří Kléma addresses the problem of mining high-dimensional, quantitative, and noisy data like transcriptomic data. The quantitative AR approach is based on simple arithmetic operations with variables and it outputs rules that are syntactically like classical association rules. They use prior knowledge (expressed, for example, in a gene similarity matrix) to find promising rule candidates, thus pruning the search space and reducing the number of derived rules. *Conceptual Clustering Applied to Ontologies by means of Semantic Discernability* by Floriana Esposito, Nicola Fanizzi, and Claudia d'Amato proposes a way of clustering objects described in a logical language. The clustering method relies on a semi-distance measure and combines bisecting k-means and medoids into a hierarchical extension of the PAM algorithm (Partition Around

Medoids). *Nonlinear knowledge in learning models* by Mario R. Guarracino, Danilo Abbate, and Roberto Prevete proposes a method to include nonlinear prior knowledge in a Generalized Eigenvalues Support Vector Machine. Prior knowledge here is expressed as additional terms of the cost function of the SVM optimization problem. This improves both algorithmic complexity and prediction accuracy, as shown in a medical case study.

Web Mining 2.0 sessions

The workshop accommodates four papers and one invited talk. In his invited talk *Using context models and models for contextually instantiated social relations for mobile social computing services*, George Groh will discuss services that combine models of social structures and context awareness.

Two of the papers are on the dissemination of semantics in the Web, one of them dealing with automated semantic annotation and the other with the extraction of information from Web documents. The first (*Using Term-Matching Algorithms for the Annotation of Geo-Services* by Grcar and Klien) involves the use of prior conceptual knowledge and is therefore part of the joint session. In the second paper, Raeymaekers and Bruynooghe propose *A Hybrid Approach Towards Wrapper Induction*. They elaborate on wrapper induction for the extraction of information from Web documents: They point out that "tree-based" approaches, which observe a Web document as a tree structure, require less training examples than "string-based" approaches, which treat a document as a string of tokens. To achieve the flexibility and fine granularity possible with string-based approaches, they extend a tree-based wrapper induction method to a hybrid one.

Two further papers of the Web Mining 2.0 workshop deal with the dissemination of preferences about content. The study of Baltrunas and Ricci on *Dynamic Item Weighting and Selection for Collaborative Filtering* investigates the use of item ratings for collaborative filtering in a recommendation engine. The authors study different methods for item weighting and item selection, with the intention to increase recommendation accuracy despite data sparsity and high dimensionality of the feature space. In *Mining Music Social Networks for Automating Social Music Services*, Baccigalupo and Plaza study sequences of music songs, as found in music social networks, and propose a method for the prediction of the most appropriate next song in a playlist.

Joint session PriCKL / Web Mining 2.0

Prior conceptual knowledge has a large importance for the Web. Approaches range from the use of background knowledge (or "semantics") to improve the results of mining Web resources, to the use of background knowledge in mining various other resources to improve the Web. The two contributed papers from PriCKL in the joint session illustrate these two forms.

The Ex Project:Web Information Extraction using Extraction Ontologies by Martin Labský, Vojtěch Svátek, Marek Nekvasil and Dušan Rak addresses the problem of using background knowledge for extracting knowledge from the Web. They use richly-structured extraction ontologies to aid the Information Extraction task. The system also

makes it possible to re-use third-party ontologies and the results of inductive learning for subtasks where pre-labelled data abound.

Dealing with Background Knowledge in the SEWEBAR Project by Jan Rauch and Milan Šimůnek illustrates the use of data mining with background knowledge for creating the Semantic Web. The goal is to generate, in a decentralized fashion, local analytical reports about a domain, and then to combine them, on the Semantic Web, into global analytical reports. Background knowledge is applied on a meta-level to guide the functioning of the mining algorithms themselves (in this case, GUHA). An example of background knowledge are useful category boundaries / granularity for quantitative attributes. The case study concerns relationships between health-related behavioural variables.

In *Using Term-Matching Algorithms for the Annotation of Geo-Services*, Grcar and Klien study semantic annotation of spatial objects: Their objective is to associate terms that describe the spatial objects with appropriate concepts from a domain ontology. Their method achieves this by associating terms with documents fetched from the Web and then assessing term similarity (and term/concept similarity) on the basis of (a) document similarity, (b) linguistic patterns and (c) Google distance.

Research on the Semantic Web in particular and semantic technologies in general continues to profit from the support of the European Union. In his Invited Talk, Stefano Bertolo gives an overview of *EU funding opportunities in research of intelligent content and semantics in Call 3 of Framework Programme 7*.

We thank our reviewers for their careful help in selecting and improving submissions, the ECML/PKDD organizers and especially the Workshops Chairs for their support, our projects SEVENPRO and KDubiq, and the PASCAL project and the Czech Society for Cybernetics and Informatics for sponsoring.

August 2007

The Workshop Chairs:
Bettina Berendt
Dunja Mladenič
Myra Spiliopoulou
Gerd Stumme
Giovanni Semeraro
Vojtěch Svátek
Filip Železný

Workshop Organization

Workshop Chairs

PriCKL'07

Bettina Berendt (Institute of Information Systems, Humboldt University Berlin, Germany)
Vojtěch Svátek (University of Economics, Prague, Czech Republic)
Filip Železný (Czech Technical University, Prague, Czech Republic)

Web Mining 2.0

Bettina Berendt (Institute of Information Systems, Humboldt University Berlin, Germany)
Dunja Mladenič (J. Stefan Institute, Ljubliana, Slovenia)
Giovanni Semeraro (Department of Informatics, University of Bari, Italy)
Myra Spiliopoulou (Faculty of Computer Science, Otto-von-Guericke-Univ. Magdeburg, Germany)
Gerd Stumme (Knowledge and Data Engineering Group, University of Kassel, Germany)

ECML/PKDD Workshop Chair

Marzena Kryszkiewicz (Warsaw University of Technology)

Workshop Program Committees

PriCKL'07

Sarabjot Singh Anand
Martin Atzmueller
Laurent Brisson
Mario Cannataro
Martine Collard
Nicola Fanizzi
Peter Flach
Aldo Gangemi
Marko Grobelnik
Alipio Jorge
Nada Lavrac
Francesca Lisi
Bernardo Magnini
Stan Matwin
Dunja Mladenic
Bamshad Mobasher
Jan Rauch
Massimo Ruffolo
Myra Spiliopoulou
Steffen Staab
York Sure

Web Mining 2.0

Andreas Hotho
Maarten van Someren
Ernestina Menasalvas
Janez Brank
Michelangelo Ceci
Marco de Gemmis
Natalie Glance
Marko Grobelnik
Matthew Hurst
Pasquale Lops
Ion Muslea
Nicolas Nicolov
George Paliouras
Sarabjot Anand

PriCKL'07 Sponsoring Institutions

Czech Society for Cybernetics and Informatics

Web Mining 2.0 Sponsoring Institutions

EU Network of Excellence PASCAL
Pattern Analysis, Statistical Modelling, and Computational Learning

Table of Contents

II Web Mining 2.0 Papers

Part I

PricKL'07 Papers

A Knowledge-Intensive Approach for Semi-Automatic Causal Subgroup Discovery

Martin Atzmueller and Frank Puppe

University of Würzburg,
Department of Computer Science VI
Am Hubland, 97074 Würzburg, Germany
{atzmueller, puppe}@informatik.uni-wuerzburg.de

Abstract. In this paper, we present a methodological approach for knowledge-intensive causal subgroup discovery. We show how to identify causal relations between subgroups by generating an extended causal subgroup network utilizing background knowledge. Using the links within the network we can identify true causal relations, but also relations that are potentially confounded and/or effect-modified by external (confounding) factors. In a semi-automatic approach, the network and the discovered relations are then presented to the user as an intuitive visualization. The applicability and benefit of the presented technique is illustrated by examples from a case-study in the medical domain.

1 Introduction

Subgroup discovery (e.g., [1–4]) is a powerful approach for explorative and descriptive data mining to obtain an overview of the interesting dependencies between a specific target (dependent) variable and usually many explaining (independent) variables, for example, the risk of coronary heart disease (target variable) is significantly higher in the subgroup of smokers with a positive family history than in the general population.

When interpreting and applying the discovered relations, it is often necessary to consider the patterns in a causal context. However, considering an association with a causal interpretation can often lead to incorrect results, due to the basic tenet of statistical analysis that association does not imply causation (cf., [5]): A subgroup may not be causal for the target group, and thus can be suppressed by other causal groups. Then, the suppressed subgroup itself is not interesting, but the other subgroups are better suited for characterizing the target concept. Furthermore, the estimated *effect*, i.e., the quality of the subgroup may be due to associations with other *confounding* factors that were not considered in the quality computation. For instance, the quality of a subgroup may be confounded by other variables that are associated with the independent variables, and are a direct cause of the (dependent) target variable. Then, it is necessary to identify potential confounders, and to measure or to control their influence concerning the subgroup and the target concept. Let us assume, for example, that ice cream consumption and murder rates are highly correlated. However, this does not necessarily mean that ice cream incites murder or that murder increases the demand for ice cream. Instead, both ice cream and murder rates might be joint effects of a common cause or confounding factor, namely, hot weather.

In this paper, we present a methodological approach for the semi-automatic detection of (true) causal subgroups and potentially confounded and/or effect-modified relations. We apply known subgroup patterns in a knowledge-intensive process as background knowledge that can be incrementally refined: The applied patterns represent subgroups that are *acausal*, i.e., have no causes, and subgroup patterns that are known to be directly causally related to other (target) subgroups. Additionally, both concepts can be combined, for example, in the medical domain certain variables such as *Sex* have no causes, and it is known that they are causal risk factors for certain diseases. Using the patterns contained in the background knowledge, and a set of subgroups for analysis, we can construct a causal net containing relations between the subgroups. In a semi-automatic process, this network can be interactively inspected and analyzed by the user: It directly provides a visualization of the causal relations between the subgroups, and also provides for a possible explanation of these. By traversing the relations in the network, we can then identify causal relations, potential confounding and/or effect-modification.

The rest of the paper is organized as follows: First, we discuss the background of subgroup discovery, the concept of confounding, and basic constraint-based causal analysis methods in Section 2. After that we present the knowledge-intensive causal discovery approach for detecting causal and confounding/effect-modified relations in Section 3. Exemplary results of the application of the presented approach are given in Section 4 using data from a fielded system in the medical domain. Finally, Section 5 concludes the paper with a discussion of the presented work.

2 Background

In this section, we first introduce the necessary notions concerning the used knowledge representation, before we define the setting for subgroup discovery. After that, we introduce the concept of confounding, criteria for its identification, and describe basic constraint-based techniques for causal subgroup analysis.

2.1 Basic Definitions

Let Ω_A be the set of all attributes. For each attribute $a \in \Omega_A$ a range $dom(a)$ of values is defined; $\mathcal{V}_A$ is assumed to be the (universal) set of attribute values of the form $(a = v)$, where $a \in \Omega_A$ is an attribute and $v \in dom(a)$ is an assignable value. We consider nominal attributes only so that numeric attributes need to be discretized accordingly. Let CB be the case base (data set) containing all available cases (instances): A case $c \in CB$ is given by the n-tuple $c = ((a_1 = v_1), (a_2 = v_2), \ldots, (a_n = v_n))$ of $n = |\Omega_A|$ attribute values, $v_i \in dom(a_i)$ for each a_i.

2.2 Subgroup Discovery

The main application areas of subgroup discovery (e.g., [1–4]) are exploration and descriptive induction, to obtain an overview of the relations between a (dependent) target variable and a set of explaining (independent) variables. As in the *MIDOS* approach [1],

we consider subgroups that are, e.g., as large as possible, and have the most unusual (distributional) characteristics with respect to the concept of interest given by a binary target variable. Therefore, not necessarily complete relations but also partial relations, i.e., (small) subgroups with "interesting" characteristics can be sufficient.

Subgroup discovery mainly relies on the subgroup description language, the quality function, and the search strategy. Often, heuristic methods (e.g., [3]) but also efficient exhaustive algorithms (e.g., the SD-Map algorithm [4]) are applied. The description language specifies the individuals belonging to the subgroup. For a common single-relational propositional language a subgroup description can be defined as follows:

Definition 1 (Subgroup Description). *A subgroup description* $sd = \{e_1, e_2, \ldots, e_n\}$ *is defined by the conjunction of a set of selectors* $e_i = (a_i, V_i)$*: Each of these are selections on domains of attributes,* $a_i \in \Omega_A, V_i \subseteq dom(a_i)$*. We define* Ω_E *as the set of all selectors and* Ω_{sd} *as the set of all possible subgroup descriptions.*

A quality function measures the interestingness of the subgroup and is used to rank these. Typical quality criteria include the difference in the distribution of the target variable concerning the subgroup and the general population, and the subgroup size.

Definition 2 (Quality Function). *Given a particular target variable* $t \in \Omega_E$*, a quality function* $q : \Omega_{sd} \times \Omega_E \rightarrow R$ *is used in order to evaluate a subgroup description* $sd \in \Omega_{sd}$*, and to rank the discovered subgroups during search.*

Several quality functions were proposed (cf., [1–4]), e.g., the functions q_{BT} and q_{RG}:

$$q_{BT} = \frac{(p - p_0) \cdot \sqrt{n}}{\sqrt{p_0 \cdot (1 - p_0)}} \cdot \sqrt{\frac{N}{N - n}}, \quad q_{RG} = \frac{p - p_0}{p_0 \cdot (1 - p_0)}, n \geq \mathcal{T}_{Supp},$$

where p is the relative frequency of the target variable in the subgroup, p_0 is the relative frequency of the target variable in the total population, $N = |CB|$ is the size of the total population, and n denotes the size of the subgroup.

In contrast to the quality function q_{BT} (the classic binomial test), the quality function q_{RG} only compares the target shares of the subgroup and the total population measuring the *relative gain*. Therefore, a support threshold $\mathcal{T}_{Supp}$ is necessary to discover significant subgroups.

The result of subgroup discovery is a set of subgroups. Since subgroup discovery methods are not necessarily covering algorithms the discovered subgroups can overlap significantly and their estimated quality (effect) might be confounded by external variables. In order to reduce the redundancy of the subgroups and to identify potential confounding factors, methods for causal analysis can then be applied.

2.3 The Concept of Confounding

Confounding can be described as a bias in the estimation of the effect of the subgroup on the target concept due to attributes affecting the target concept that are not contained in the subgroup description [6]. Thus, confounding is caused by a lack of comparability between subgroup and complementary group due to a difference in the distribution of the target concept caused by other factors.

An extreme case for confounding is presented by *Simpson's Paradox*: The (positive) effect (association) between a given variable X and a variable T is countered by a negative association given a third factor F, i.e., X and T are negatively correlated in the subpopulations defined by the values of F. For binary variables X, T, F this can be formulated as

$$P(T|X) > P(T|\neg X)\,, P(T|X,F) < P(T|\neg X,F)\,, P(T|X,\neg F) < P(T|\neg X,\neg F)\,,$$

i.e., the event X increases the probability of T in a given population while it decreases the probability of T in the subpopulations given by the restrictions on F and $\neg F$. For the example shown in Figure 1, let us assume that there is a positive correlation between the event X that describes *people that do not consume soft drinks* and T specifying the diagnosis *diabetes*. This association implies that people not consuming soft drinks are affected more often by diabetes (50% non-soft-drinkers vs. 40% soft-drinkers). However, this is due to age, if older people (given by F) consume soft drinks less often than younger people, and if diabetes occurs more often for older people, inverting the effect.

Combined	T	¬T	∑	Rate (T)
X	25	25	50	50%
¬X	20	30	50	40%
∑	45	55	100	

Restricted on F	T	¬T	∑	Rate (T)
X	24	16	40	60%
¬X	8	2	10	80%
∑	32	18	50	

Restricted on ¬F	T	¬T	∑	Rate (T)
X	1	9	10	10%
¬X	12	28	40	30%
∑	13	37	50	

Fig. 1. Example: Simpson's Paradox

There are three criteria that can be used to identify a **confounding factor** F [6], given the factors X contained in a subgroup description and a target concept T:

1. A confounding factor F must be a *cause* for the target concept T, e.g., an independent risk factor for a certain disease.
2. The factor F must be associated/correlated with the subgroup (factors) X.
3. A confounding factor F must *not* be affected by the subgroup (factors) X.

However, these criteria are only necessary but not sufficient to identify confounders. If purely automatic methods are applied for detecting confounding, then such approaches may label some variables as confounders incorrectly, e.g., if the real confounders have not been measured, or if their contributions cancel out. Thus, user interaction is rather important for validating confounded relations. Furthermore, the identification of confounding requires causal (background) knowledge since confounding is itself a causal concept [7, Chapter 6.2].

Proxy Factors and Effect Modification There are two phenomena that are closely related to confounding. First, a factor may only be associated with the subgroup but may be the real cause for the target concept. Then, the subgroup is only a **proxy factor**. Another situation is given by **effect modification**: Then, a third factor F does not necessarily need to be associated with the subgroup described by the factors X; F can be an additional factor that increases the effect of X in a certain subpopulation only, pointing to new subgroup descriptions that are interesting by themselves.

2.4 Constraint-Based Methods for Causal Subgroup Analysis

In general, the philosophical concept of causality refers to the set of all particular 'cause-and-effect' or 'causal' relations. A subgroup is causal for the target group, if in an ideal experiment [5] the probability of an object not belonging to the subgroup to be a member of the target group increases or decreases when the characteristics of the object are changed such that the object becomes a member of the subgroup. For example, the probability that a patient survives (target group) increases if the patient received a special treatment (subgroup). Then, a redundant subgroup, that is, e.g., conditionally independent from the target group given another subgroup, can be suppressed.

For causal analysis the subgroups are represented by binary variables that are true for an object (case) if it is contained in the subgroup, and false otherwise. For constructing a causal subgroup network, constraint-based methods are particularly suitable because of scalability reasons (cf., [5, 8]).

These methods make several assumptions (c.f. [5]) w.r.t. the data and the correctness of the statistical tests. The crucial condition is the *Markov condition* (c.f. [5]) depending on the assumption that the data can be expressed by a Bayesian network: Let X be a node in a causal Bayesian network, and let Y be any node that is not a descendant of X in the causal network. Then, the Markov condition holds if X and Y are independent conditioned on the parents of X.

The CCC and CCU rules [8] described below constrain the possible causal models by applying simple statistical tests: For subgroups s_1, s_2, s_3 represented by binary variables the χ^2-test for independence is utilized for testing their independence $ID(s_1, s_2)$, dependence $D(s_1, s_2)$ and conditional independence $CondID(s_1, s_2|s_3)$, shown below (for the tests user-selectable thresholds are applied, e.g., $\mathcal{T}_1 = 1, \mathcal{T}_2 = 3.84$, or higher):

$$ID(s_1, s_2) \longleftrightarrow \chi^2(s_1, s_2) < \mathcal{T}_1\,, \qquad D(s_1, s_2) \longleftrightarrow \chi^2(s_1, s_2) > \mathcal{T}_2\,,$$
$$CondID(s_1, s_2|s_3) \longleftrightarrow \chi^2(s_1, s_2|s_3 = 0) + \chi^2(s_1, s_2|s_3 = 1) < 2 \cdot \mathcal{T}_1$$

Thus, the decision of (conditional) (in-)dependence is threshold-based, which is a problem causing potential errors if very many tests are performed. Therefore, we propose a semi-automatic approach featuring interactive analysis of the inferred relations.

Definition 3 (CCC Rule). *Let X, Y, Z denote three variables that are pairwise dependent, i.e., $D(X, Y), D(X, Z), D(Y, Z)$; let X and Z become independent when conditioned on Y. In the absence of hidden and confounding variables we may infer that one of the following causal relations exists between X, Y and Z: $X \rightarrow Y \rightarrow Z$, $X \leftarrow Y \rightarrow Z$, $X \leftarrow Y \leftarrow Z$. However, if X has no causes, then the first relation is the only one possible, even in the presence of hidden and confounding variables.*

Definition 4 (CCU Rule). *Let X, Y, Z denote three variables: X and Y are dependent ($D(X, Y)$), Y and Z are dependent ($D(Y, Z)$), X and Z are independent ($ID(X, Z)$), but X and Z become dependent when conditioned on Y ($CondD(X, Z|Y)$). In the absence of hidden and confounding variables, we may infer that X and Z cause Y.*

3 Semi-Automatic Causal Subgroup Discovery and Analysis

The approach for knowledge-intensive causal subgroup discovery is embedded in an incremental semi-automatic process. In the following, we first introduce the process model for causal subgroup discovery and analysis. After that, we present the necessary background knowledge for effective causal analysis. Next, we describe a method for constructing an extended causal subgroup network, and discuss how to identify confounded and effect-modified relations.

3.1 Process Model for Causal Subgroup Discovery and Analysis

The steps of the process for semi-automatic causal subgroup discovery are shown in Figure 2:

1. First, the user applies a standard *subgroup discovery* approach, e.g., [3, 4]. The result of this step is a set of the most interesting subgroups. Optionally, background knowledge contained in the knowledge base can also be applied during subgroup discovery (e.g., as discussed in [9]).
2. Next, we apply *causal analysis* using appropriate background knowledge for a detailed analysis of the discovered associations. Using constraint-based techniques for causal analysis, a (partial) causal subgroup network is constructed.
3. In the *evaluation and validation* phase, the user assesses the (partial) causal network: Since the relations contained in the network can be wrong due to various statistical errors, inspection and validation of the causal relations is essential in order to obtain valid results. Then, the final results are obtained.
4. The user can extend and/or tune the applied background knowledge during the *knowledge extraction* step: Then, the knowledge base can be updated in incremental fashion by including further background knowledge, based upon the discovery results.

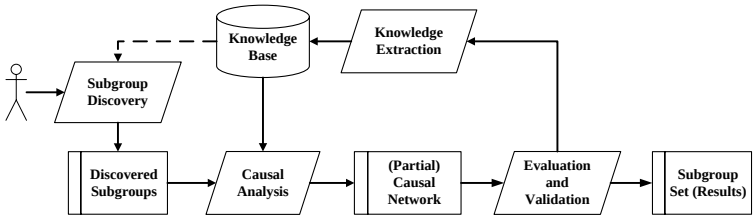


Fig. 2. Process Model for Knowledge-Intensive Causal Subgroup Analysis

3.2 Extended Causal Subgroup Analysis

Detecting causal relations, i.e., (true) causal associations, confounding, and effect modification using causal subgroup analysis consists of two main steps that can be iteratively applied:

1. First, we generate a causal subgroup network considering a target group T, a user-selected set of subgroups U, a set of confirmed (causal) and potentially confounding factors C for any included group, a set of unconfirmed potentially confounding factors P given by subgroups significantly dependent with the target group, and additional background knowledge described below. In addition to causal links, the generated network also contains (undirected) associations between the variables.
2. In the next step, we traverse the network and mark the potential confounded and/or effect-modified relations. The causal network and the proposed relations are then presented to the user for subsequent interpretation and analysis. After confounding factors have been confirmed, the background knowledge can then be extended.

In the following sections we discuss these steps in detail. First, we describe the elements of the background knowledge that are utilized for causal analysis.

3.3 Background Knowledge for Causal Analysis

For the causal analysis step we first need to generate an extended causal network capturing the (causal) relations between the subgroups represented by binary variables. In order to effectively generate the network, we need to include background knowledge provided by the user, e.g., by a domain specialist. In the medical domain, for example, a lot of background knowledge is already available and can be directly integrated in the analysis phase. Examples include (causal) relations between diseases, and the relations between diseases and their respective findings.
The applicable background consists of two elements:

1. Acausal factors: These include factors represented by subgroups that have no causes; in the medical domain, e.g., the subgroup $Age \geq 70$ or the subgroup *Sex = male* have no causes, while the subgroups *BMI = underweight* has certain causes.
2. (Direct) causal relations: Such relations include subgroups that are not only dependent but (directly) causal for the target group/target variable and/or other subgroups. In the medical domain, for example, the subgroup *Body-Mass-Index (BMI)=overweight* is directly causal for the subgroup *Gallstones=probable*.

Depending on the domain, it is often easier to provide acausal information. Direct and indirect causal relations are often also easy to acquire, and can be acquired 'on-the-fly' when applying the presented process. However, in some domains, e.g., in the medical domains, it is often difficult to provide non-ambiguous directed relationships between certain variables: One disease can cause another disease and vice versa, under different circumstances. In such cases, the relations should be formalized with respect to both directions, and can still be exploited in the method discussed below. Then, the interpretation performed by the user is crucial in order to obtain valid and ultimately interesting results.

3.4 Constructing an Extended Causal Subgroup Network

Algorithm 1 summarizes how to construct an extended causal subgroup network, based on a technique for basic causal subgroup analysis described in [2, 10]. When applying the algorithm, the relations contained in the network can be wrong due to various statistical errors (cf., [5]), especially for the CCU rule (cf., [8]). Therefore, after applying the algorithm, the resulting causal net is presented to the user for interactive analysis.

The first step (lines 1-5) of the algorithm determines for each subgroup pair (including the target group) whether they are independent, based on the inductive principle that the dependence of subgroups is necessary for their causality.

In the next step (lines 6-10) we determine for any pair of subgroups whether the first subgroup s_1 is suppressed by a second subgroup s_2, i.e., if s_1 is conditionally independent from the target group T given s_2. The χ^2-measure for the target group and s_1 is calculated both for the restriction on s_2 and its complementary subgroup. If the sum of the two test-values is below a threshold, then we can conclude that subgroup s_1 is conditionally independent from the target group. Conditional independence is a sufficient criterion, since the target distribution of s_1 can be explained by the target distribution in s_2, i.e., by the intersection. Since very similar subgroups could symmetrically suppress each other, the subgroups are ordered according to their quality, and then subgroups with a nearly identical extension (and a lower quality) can be eliminated.

The next two steps (lines 11-18) check conditional independence between each pair of subgroups given the target group or a third subgroup, respectively. For each pair of conditionally independent groups, the separating (conditioning) group is noted. Then, this separator information is exploited in the next steps, i.e., independencies or conditional independencies for pairs of groups derived in the first steps are used to exclude any causal links between the groups. The conditioning steps (lines 6-18) can optionally be iterated in order to condition on combinations of variables (pairs, triples). However, the decisions taken further (in the CCU and CCC rules) may become statistically weaker justified due to smaller counts in the considered contingency tables (e.g., [5, 10]).

Direct causal links (line 19) are added based on background knowledge, i.e., given subgroup patterns that are directly causal for specific subgroups. In the last step (lines 24-26) we also add conditional associations for dependent subgroups that are not conditionally independent and thus not suppressed by any other subgroups. Such links can later be useful in order to detect the (true) associations considering a confounding factor.

Extending CCC and CCU using Background Knowledge The CCU and CCC steps (lines 20-23) derive the directions of the causal links between subgroups, based on information derived in the previous steps. In the context of the presented knowledge-intensive approach, we extend the basic CCC and CCU rules including background knowledge both for the derivation of additional links, and for inhibiting links that contradict the background knowledge. We introduce associations instead of causal directions if these are wrong, or if not enough information is available in order to derive the causal directions. The rationale behind this principle is given by the intuition that we want to exploit and provide as much information as possible considering the generated causal net. When identifying potentially confounded relations, we can also often utilize weaker associative information.

Algorithm 1 Constructing a causal subgroup net

Require: Target group T, user-selected set of subgroups U, potentially confounding groups P, background knowledge B containing acausal subgroup information, and known subgroup patterns $C \subseteq B$ that are directly causal for other subgroups. Define $\mathcal{S} = U \cup P \cup C$

for all $s_i, s_j \in \mathcal{S} \cup T, s_i \neq s_j$ **do**
  if $approxEqual(s_i, s_j)$ **then**
    Exclude any causalities for the subgroup with smaller correlation to T
  if $ID(s_i, s_j)$ **then**
    Exclude causality: $ex(s_i, s_j) = true$
for all $s_i, s_j \in \mathcal{S}, s_i \neq s_j$ **do**
  if $\neg ex(s_i, T)$, $\neg ex(s_j, T)$, or $\neg ex(s_i, s_j)$ **then**
    if $CondID(s_i, T|s_j)$ **then**
      Exclude causality: $ex(s_i, T) = true$, and include s_j into $separators(s_i, T)$
      If conditional independencies are symmetric, then select the strongest relation
for all $s_i, s_j \in \mathcal{S}, i < j$ **do**
  if $\neg ex(s_i, T)$, $\neg ex(s_j, T)$, or $\neg ex(s_i, s_j)$ **then**
    if $CondID(s_i, s_j|T)$ **then**
      Exclude causality: $ex(s_i, s_j) = true$, and include T into $separators(s_i, s_j)$
for all $s_i, s_j, s_k \in \mathcal{S}, i < j, i \neq k, j \neq k$ **do**
  if $\neg ex(s_i, s_j)$, $\neg ex(s_j, s_k)$, or $\neg ex(s_i, s_k)$ **then**
    if $CondID(s_i, s_j|s_k)$ **then**
      Exclude causality: $ex(s_i, s_j) = true$, and include s_k into $separators(s_i, s_j)$
Integrate direct causal links that are not conditionally excluded considering the sets C and B
for all $s_i, s_j, s_k \in \mathcal{S}$ **do**
  Apply the extended CCU rule, using background knowledge
for all $s_i, s_j, s_k \in \mathcal{S}$ **do**
  Apply the extended CCC rule, using background knowledge
for all $s_i, s_j, s_k \in \mathcal{S} \cup \{T\}, i < j, i \neq k, j \neq k$ **do**
  if $\neg CondIDs_i, s_j|s_k$ **then**
    Integrate association between dependent s_i and s_j that are not conditionally excluded

For the **extended** CCU **rule** we use background knowledge for inhibiting acausal directions, since the CCU rule can be disturbed by confounding and hidden variables. The causal or associative links do not necessarily indicate direct associations/causal links but can also point to relations enabled by hidden or confounding variables [8].

For the **extended** CCC **rule**, we can use the relations inferred by the extended CCU rule for disambiguating between the causal relations, if the CCU rule is applied in all possible ways: The non-separating condition (conditional dependency) of the relation identified by the CCU rule is not only a sufficient but a necessary condition [8], i.e., for $X \rightarrow Y \leftarrow Z$, with $CondID(X, Z|Y)$. Additionally, we can utilize background knowledge for distinguishing between the causal relations. So, for three variables X, Y, Z with $D(X, Y), D(X, Z), D(Y, Z)$, and $CondID(X, Z|Y)$, if there exists an (inferred) causal link $X \rightarrow Y$ between X and Y, we may identify the relation $X \rightarrow Y \rightarrow Z$ as the true relation. Otherwise, if Y or Z have no causes, then we select the respective relation, e.g., $X \leftarrow Y \rightarrow Z$ for an acausal variable Y.

3.5 Identifying Confounded Relations

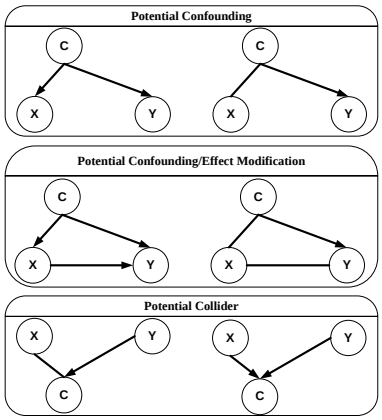


Fig. 3. Examples for Confounded Relations

A popular method for controlling confounding factors is given by *stratification* [6]: For example, in the medical domain a typical confounding factor is the attribute *age*: We can stratify on age groups such as $age < 30$, $age\ 30 - 69$, and $age \geq 70$. Then, the subgroup – target relations are measured within the different strata, and compared to the (crude) unstratified measure.
It is easy to see, that in the context of the presented approach stratification for a binary variables is equivalent to conditioning on them: If we assess a conditional subgroup – target relation and the subgroup factors become independent (or dependent), then this indicates potential confounding. After constructing a causal net, we can easily identify such relations. Since the causal directions derived by the extended CCC and CCU rules may be ambiguous, user interaction is crucial: In some domains, e.g., in the medical domains, it is often difficult to provide non-ambiguous directed relationships between certain variables: One disease can cause another disease and vice versa, under different circumstances. The network then also provides an intuitive visualization for the analysis.

In order to identify potentially confounded relations and the corresponding variables, as shown in Figure 3, and described below, we just need to traverse the network:

- **Potential Confounding**: If there is an association between two variables X and Y, and the network contains the relations $C \rightarrow X$, $C \rightarrow Y$, and there is no link between X and Y, i.e., they are conditionally independent given C, then C is a confounder that inhibits the relation between X and Y. This is also true if there is no causal link between C and X but instead an association.
- **Potential Confounding/Effect Modification**: If the network contains the relations $C \rightarrow X$, $C \rightarrow Y$, and there is also either an association or a causal link between X and Y, then this points to confounding and possible effect modification of the relation between the variables X and Y.
- **Potential Collider (or Confounder)**: If there is no (unconditioned) association between two variables X and Y and the network contains the relations $X \rightarrow C$ and $Y \rightarrow C$, then C is a potential collider: X and Y become dependent by conditioning on C. The variable C is then no confounder in the classical sense, if the (derived) causal relations are indeed true. However, such a relation as inferred by the CCU rule can itself be distorted by confounded and hidden variables. The causal directions could also be inverted, if the association between X and Y is just not strong enough as estimated by the statistical tests. In this case, C is a potential confounder. Therefore, manual inspection is crucial in order to detect the true causal relation.

4 Examples

We applied a case base containing about 8600 cases taken from the SONOCONSULT system [11] – a medical documentation and consultation system for sonography. The system is in routine use in the DRK-hospital in Berlin/Köpenick, and the collected cases contain detailed descriptions of findings of the examination(s), together with the inferred diagnoses (binary attributes). The experiments were performed using the VIKAMINE system [12] implementing the presented approach.

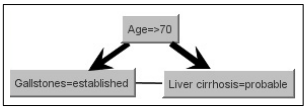


Fig. 4. Confounded Relation: Gallstones and Liver cirrhosis

In the following, we provide some (simplified) examples considering the diagnosis *Gallstones=established* as the target variable. After applying a subgroup discovery method, several subgroups were selected by the user in order to derive a causal subgroup network, and to check the relations w.r.t. possible confounders. These selected subgroups included, for example, the subgroup *Fatty liver=probable or possible* and the subgroup *Liver cirrhosis=probable*.

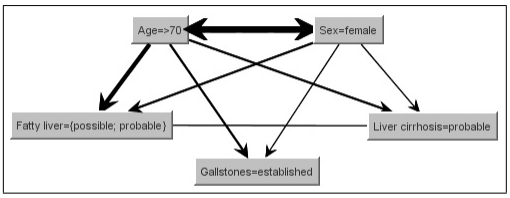


Fig. 5. Confounded Relations: Gallstones, Liver cirrhosis and Fatty Liver

A first result is shown in Figure 4: In this network, the subgroup *Liver cirrhosis=probable* is confounded by the variable $Age \geq 70$. However, there is still an influence on the target variable considering the subgroup *Liver cirrhosis=probable* shown by the association between the subgroups. This first result indicates confounding and effect modification (the strengths of the association between the nodes is also visualized by the widths of the links). A more detailed result is shown in Figure 5: In this network another potential confounder, i.e., *Sex=female* is included. Then, it becomes clear, that both the subgroup *Fatty liver=probable or possible* and the subgroup *Liver cirrhosis=probable* are confounded by the variables *Sex* and *Age*, and the association (shown in Figure 4) between the subgroup *Liver cirrhosis=probable* and the target group is then no longer present (In this example the removal of the gall-bladder was not considered which might have an additional effect concerning a medical interpretation).

It is easy to see that the generated causal subgroup network becomes harder to interpret, if many variables are included, and if the number of connections between the nodes increases. Therefore, we provide filters, e.g., in order to exclude (non-causal) associations. The nodes and the edges of the network can also be color-coded in order to increase their interpretability: Based on the available background knowledge, causal subgroup nodes, and (confirmed) causal directions can be marked. Since the network is first traversed and the potentially confounded relations are reported to the user, the analysis can also be focused towards the respective variables, as a further filtering condition. The user can then analyze selected parts of the network in more detail.

5 Conclusion

In this paper, we have presented a methodological approach for knowledge-intensive causal subgroup discovery: We utilize background knowledge for establishing an extended causal model of the domain. The constructed network can then be used to identify potential (true) causal relations, and confounded/effect-modified associations. In a semi-automatic approach, these can then be evaluated and validated by the user. Furthermore, the available background knowledge can be incrementally refined and extended.

In the future, we are planning to consider an efficient approach for detecting confounding that is directly embedded in the subgroup discovery method. Related work in that direction was described, for example, in [13]. Another interesting direction for future work is given by considering further background knowledge for causal analysis.

Acknowledgements

This work has been partially supported by the German Research Council (DFG) under grant Pu 129/8-1.

References

1. Wrobel, S.: An Algorithm for Multi-Relational Discovery of Subgroups. In: Proc. 1st Europ. Symp. Principles of Data Mining and Knowledge Discovery, Berlin, Springer (1997) 78–87
2. Klösgen, W.: 16.3: Subgroup Discovery. In: Handbook of Data Mining and Knowledge Discovery. Oxford University Press, New York (2002)
3. Lavrac, N., Kavsek, B., Flach, P., Todorovski, L.: Subgroup Discovery with CN2-SD. Journal of Machine Learning Research **5** (2004) 153–188
4. Atzmueller, M., Puppe, F.: SD-Map - A Fast Algorithm for Exhaustive Subgroup Discovery. In: Proc. 10th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD 2006), Berlin, Springer (2006) 6–17
5. Cooper, G.F.: A Simple Constraint-Based Algorithm for Efficiently Mining Observational Databases for Causal Relationships. Data Min. Knowl. Discov. **1**(2) (1997) 203–224
6. McNamee, R.: Confounding and Confounders. Occup. Environ. Med. **60** (2003) 227–234
7. Pearl, J.: Causality: Models, Reasoning and Inference. Cambridge University Press (2000)
8. Silverstein, C., Brin, S., Motwani, R., Ullman, J.D.: Scalable Techniques for Mining Causal Structures. Data Mining and Knowledge Discovery **4**(2/3) (2000) 163–192
9. Atzmueller, M., Puppe, F., Buscher, H.P.: Exploiting Background Knowledge for Knowledge-Intensive Subgroup Discovery. In: Proc. 19th Intl. Joint Conference on Artificial Intelligence (IJCAI-05), Edinburgh, Scotland (2005) 647–652
10. Kloesgen, W., May, M.: Database Integration of Multirelational Causal Subgroup Mining. Technical report, Fraunhofer Institute AIS, Sankt Augustin, Germany (2002)
11. Huettig, M., Buscher, G., Menzel, T., Scheppach, W., Puppe, F., Buscher, H.P.: A Diagnostic Expert System for Structured Reports, Quality Assessment, and Training of Residents in Sonography. Medizinische Klinik **99**(3) (2004) 117–122
12. Atzmueller, M., Puppe, F.: Semi-Automatic Visual Subgroup Mining using VIKAMINE. Journal of Universal Computer Science **11**(11) (2005) 1752–1765
13. Fabris, C.C., Freitas, A.A.: Discovering Surprising Patterns by Detecting Occurrences of Simpson's Paradox. In: Research and Development in Intelligent Systems XVI, Berlin, Springer (1999) 148–160

EU funding opportunities in research of intelligent content and semantics in Call 3 of Framework Programme 7 *(Invited Talk)*

Stefano Bertolo

European Commission, Brussels, Belgium
Stefano.BERTOLO@ec.europa.eu

Conceptual Clustering Applied to Ontologies by means of Semantic Discernability

Floriana Esposito, Nicola Fanizzi, and Claudia d'Amato

Dipartimento di Informatica – Università degli Studi di Bari
Campus Universitario, Via Orabona 4, 70125 Bari, Italy
{esposito|fanizzi|claudia.damato}@di.uniba.it

Abstract. A clustering method is presented which can be applied to relational knowledge bases to discover interesting groupings of resources through their annotations expressed in the standard languages of the Semantic Web. The method exploits a simple (yet effective and language-independent) semi-distance measure for individuals, that is based on the semantics of the resources w.r.t. a number of dimensions corresponding to a set of concept descriptions (discriminating features). The algorithm adapts the classic BISECTING K-MEANS to work with medoids. A final experiment demonstrates the validity of the approach using absolute quality indices.

1 Introduction

In the inherently distributed applications related to the *Semantic Web* (henceforth SW) there is an extreme need of automatizing those activities which are more burdensome for the knowledge engineer, such as ontology construction, matching and evolution. Such an automation may be assisted by crafting supervised or unsupervised methods for the specific representations of the SW field (RDF through OWL) founded in *Description Logics* (DLs).

In this work, we investigate on unsupervised learning for knowledge bases expressed in such standard concept languages. In particular, we focus on the problem of conceptual clustering of semantically annotated resources. The benefits of *conceptual clustering* [16] in the context of semantically annotated knowledge bases are manifold. Clustering annotated resources enables the definition of new emerging concepts (*concept formation*) on the grounds of the primitive concepts asserted in a knowledge base; supervised methods can exploit these clusters to induce new concept definitions or to refining existing ones *ontology evolution*; intensionally defined groupings may speed-up the task of search and *discovery*; a hierarchical clustering also suggests criteria for *ranking* the retrieved resources.

Essentially, many existing clustering methods are based on the application of similarity (or density) measures defined over a fixed set of attributes of the domain objects. Classes of objects are taken as collections that exhibit low interclass similarity (density) and high intraclass similarity (density). These methods rarely take into account forms of *background knowledge* to characterize object configurations by means

of global concepts and semantic relationships. This hinders the interpretation of the outcomes of these methods which is crucial in the SW perspective which foresees sharing and reusing knowledge in order to enable semantic interoperability.

Thus, conceptual clustering methods have been developed to define groups of objects through conjunctive descriptions based on selected attributes [16]. Related works in the literature have been dedicated to similarity measures for clausal spaces [14], instance based classification [6] and clustering [12], yet these representation are known to be incomparable w.r.t. DLs [3]. A theoretical problem is posed by the *Open World Assumption* (OWA) that is generally made on the language semantics, differently from the *Closed World Assumption* (CWA) which is standard in machine learning or query-answering. As pointed out in a seminal paper on similarity measures for DLs [4], most of the existing measures focus on the similarity of atomic concepts within hierarchies or simple ontologies. Moreover, they have been conceived for assessing *concept* similarity, whereas, for other tasks, a notion of similarity between *individuals* is required.

Recently, dissimilarity measures for specific DLs have been proposed [5]. Although they turned out to be quite effective for the inductive tasks, they are still partly based on structural criteria which makes them fail to fully grasp the underlying semantics and hardly scale to any standard ontology language. Therefore, we have devised a family of dissimilarity measures for semantically annotated resources, which can overcome the aforementioned limitations. Following the criterion of semantic discernibility of individuals, we present a new family of measures that is suitable a wide range of languages since it is merely based on the discernibility of the input individuals with respect to a fixed set of features (henceforth a *committee*) represented by concept definitions. As such the new measures are not absolute, yet they depend on the knowledge base they are applied to. Thus, also the choice of the optimal feature sets deserves a preliminary feature construction phase, which may be performed by means of a randomized search procedure based on *simulated annealing*.

In this perspective, the expressiveness of the language adopted for describing objects and clusters is equally important. Alternative approaches, for terminological representations, pursued a different way for attacking the problem, devising logic-based methods for specific languages [11, 7]. Yet it has been pointed out that these methods may suffer from noise in the data. This motivates our investigation on similarity-based clustering methods which can be more noise-tolerant, and as language-independent as possible. Specifically we propose a multi-relational extension of effective clustering techniques, which is tailored for the SW context. It is intended for grouping similar resources w.r.t. a semantic dissimilarity measure. Instead of the notion of *means* that characterizes the algorithms descending from (BISECTING) K-MEANS [9] originally developed for numeric or ordinal features, we recur to the notion of *medoids* [10] as central individuals in a cluster. Hence we propose a BISECTING AROUND MEDOIDS algorithm, which exploits the aforementioned measures to work on DLs. An initial evaluation of the method applied to real ontologies is also presented based on internal validity indices such as the silhouette measure [10].

The paper is organized as follows. Sect. 2 recalls the basics of the representation and the distance measure adopted. The clustering algorithm is presented and discussed in Sect. 3. Conclusions are finally examined in Sect. 5.

2 Semantic Distance Measures

One of the strong points of our method is that it does not rely on a particular language for semantic annotations. Hence, in the following, we assume that resources, concepts and their relationship may be defined in terms of a generic ontology language that may be mapped to some DL language with the standard model-theoretic semantics (see the handbook [1] for a thorough reference).

In this context, a *knowledge base* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ contains a *TBox* $\mathcal{T}$ and an *ABox* $\mathcal{A}$. $\mathcal{T}$ is a set of concept definitions. $\mathcal{A}$ contains assertions (facts, data) concerning the world state. Moreover, normally the *unique names assumption* is made on the ABox individuals[1] therein. The set of the individuals occurring in $\mathcal{A}$ will be denoted with $\mathsf{Ind}(\mathcal{A})$. As regards the inference services, like all other instance-based methods, our procedure may require performing *instance-checking*, which amounts to determining whether an individual, say a, belongs to a concept extension, i.e. whether $C(a)$ holds for a certain concept C.

2.1 A Semantic Semi-Distance for Individuals

For our purposes, a function for measuring the similarity of individuals rather than concepts is needed. It can be observed that individuals do not have a syntactic structure that can be compared. This has led to lifting them to the concept description level before comparing them (recurring to the approximation of the *most specific concept* of an individual w.r.t. the ABox).

We have developed a new measure whose definition totally depends on semantic aspects of the individuals in the knowledge base. On a semantic level, similar individuals should behave similarly with respect to the same concepts. We introduce a novel measure for assessing the similarity of individuals in a knowledge base, which is based on the idea of comparing their semantics along a number of dimensions represented by a committee of concept descriptions. Following the ideas borrowed from ILP [15] and *multi-dimensional scaling*, we propose the definition of totally semantic distance measures for individuals in the context of a knowledge base.

The rationale of the new measure is to compare them on the grounds of their behavior w.r.t. a given set of hypotheses, that is a collection of concept descriptions, say $\mathsf{F} = \{F_1, F_2, \ldots, F_m\}$, which stands as a group of discriminating *features* expressed in the language taken into account.

In its simple formulation, a family of distance functions for individuals inspired to Minkowski's distances can be defined as follows:

Definition 2.1 (family of measures). *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a knowledge base. Given set of concept descriptions $\mathsf{F} = \{F_1, F_2, \ldots, F_m\}$, a family of functions*

$$d_p^{\mathsf{F}} : \mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A}) \mapsto \mathbb{R}$$

defined as follows:

$$\forall a, b \in \mathsf{Ind}(\mathcal{A}) \quad d_p^{\mathsf{F}}(a, b) := \frac{1}{m} \left(\sum_{i=1}^{m} \mid \pi_i(a) - \pi_i(b) \mid^p \right)^{1/p}$$

[1] Individuals can be assumed to be identified by their own URI.

where $p > 0$ *and* $\forall i \in \{1, \ldots, m\}$ *the* projection function π_i *is defined by:*

$$\forall a \in \mathsf{Ind}(\mathcal{A}) \quad \pi_i(a) = \begin{cases} 1 & \mathcal{K} \models F_i(x) \\ 0 & \mathcal{K} \models \neg F_i(x) \\ 1/2 & otherwise \end{cases}$$

It is easy to prove that these functions have the standard properties for semi-distances:

Proposition 2.1 (semi-distance). *For a fixed feature set and* $p > 0$*, given any three instances* $a, b, c \in \mathsf{Ind}(\mathcal{A})$*. it holds that:*

1. $d_p(a, b) > 0$
2. $d_p(a, b) = d_p(b, a)$
3. $d_p(a, c) \leq d_p(a, b) + d_p(b, c)$

Proof. 1. and 2. are trivial. As for 3., noted that

$$\begin{aligned}(d_p(a,c))^p &= \frac{1}{m^p}\sum_{i=1}^{m} \mid \pi_i(a) - \pi_i(c) \mid^p = \frac{1}{m^p}\sum_{i=1}^{m} \mid \pi_i(a) - \pi_i(b) + \pi_i(b) - \pi_i(c) \mid^p \\ &\leq \frac{1}{m^p}\sum_{i=1}^{m} \mid \pi_i(a) - \pi_i(b) \mid^p + \frac{1}{m^p}\sum_{i=1}^{m} \mid \pi_i(b) - \pi_i(c) \mid^p \\ &\leq (d_p(a,b))^p + (d_p(b,c))^p \leq (d_p(a,b) + d_p(b,c))^p\end{aligned}$$

then the property follows for the monotonicity of the power function.

It cannot be proved that $d_p(a, b) = 0$ iff $a = b$. This is the case of *indiscernible* individuals with respect to the given set of hypotheses F.

Compared to other proposed distance (or dissimilarity) measures [4], the presented function does not depend on the constructors of a specific language, rather it requires only retrieval or instance-checking service used for deciding whether an individual is asserted in the knowledge base to belong to a concept extension (or, alternatively, if this could be derived as a logical consequence).

Note that the π_i functions ($\forall i = 1, \ldots, m$) for the training instances, that contribute to determine the measure with respect to new ones, can be computed in advance thus determining a speed-up in the actual computation of the measure. This is very important for the measure integration in algorithms which massively use this distance, such as all instance-based methods.

The underlying idea for the measure is that similar individuals should exhibit the same behavior w.r.t. the concepts in F. Here, we make the assumption that the feature-set F represents a sufficient number of (possibly redundant) features that are able to discriminate really different individuals.

2.2 Feature Set Optimization

Experimentally, we could obtain good results by using the very set of both primitive and defined concepts found in the ontology (see Sect. 4). The choice of the concepts to be included – *feature selection* – may be crucial, for a good committee may discern

the individuals better and a possibly smaller committee yields more efficiency when computing the distance.

We have devised a specific optimization algorithms founded in *genetic programming* and *simulated annealing* (whose presentation goes beyond the scope of this work) which are able to find optimal choices of discriminating concept committees. Namely, since the function is very dependent on the concepts included in the committee of features F, two immediate heuristics can be derived: 1) control the number of concepts of the committee, including especially those that are endowed with a real discriminating power; 2) finding optimal sets of discriminating features, by allowing also their composition employing the specific constructors made available by the representation language of choice.

Both these objectives can be accomplished by means of machine learning techniques especially when knowledge bases with large sets of individuals are available. Namely, part of the entire data can be hold out in order to learn optimal F sets, in advance with respect to the successive usage for all other purposes.

We have been experimenting the usage of genetic programming for constructing an optimal set of features. Yet these methods may suffer from being possibly caught in local minima. An alternative is employing a different probabilistic search procedure which aims at a global optimization, such as taboo search or random walk algorithms. For simplicity, we devised a simulated annealing search, whose algorithm is depicted in Fig. 1.

The algorithm searches the space of all possible feature committees starting from an initial guess (determined by MAKEINITIALFS($\mathcal{K}$)) based on the concepts (both primitive and defined) currently referenced in the knowledge base. The loop controlling the search is repeated for a number of times that depends on the temperature which gradually decays to 0, when the current committee can be returned. The current feature set is iteratively refined calling a suitable procedure RANDOMSUCCESSOR(). Then the fitness of the new feature set is compared to that of the previous one determining the increment of energy ΔE. If this is non-null then the computed committee replaces the old one. Otherwise it will be replaced with a probability that depends on ΔE.

As regards the FITNESSVALUE(F), it can be computed as the *discernibility factor* of the feature set. For example given the whole set of individuals $IS = \mathsf{Ind}(\mathcal{A})$ (or just a sample to be used to induce an optimal measure) the fitness function may be:

$$\text{FITNESSVALUE}(\mathsf{F}) = \sum_{(a,b) \in IS^2} \sum_{i=1}^{|\mathsf{F}|} \mid \pi_i(a) - \pi_i(b) \mid$$

As concerns finding candidates to replace the current committee (RANDOMSUCCESSOR()), the function was implemented by recurring to simple transformations of a feature set:

- adding (resp. removing) a concept C: $\mathsf{nextFS} \leftarrow \mathsf{currentFS} \cup \{C\}$ (resp. $\mathsf{nextFS} \leftarrow \mathsf{currentFS} \setminus \{C\}$)
- randomly choosing one of the current concepts from $\mathsf{currentFS}$, say C; replacing it with one of its refinements $C' \in \text{REF}(C)$

```
FeatureSet OPTIMIZEFEATURESET(𝒦, ΔT)
input  𝒦: Knowledge base
       ΔT: function controlling the decrease of temperature
output FeatureSet
static currentFS: current Feature Set
       nextFS: next Feature Set
       Temperature: controlling the probability of downward steps
begin
currentFS ← MAKEINITIALFS(𝒦)
for t ← 1 to ∞ do
       Temperature ← Temperature − ΔT(t)
       if (Temperature = 0)
           return currentFS
       nextFS ← RANDOMSUCCESSOR(currentFS,𝒦)
       ΔE ← FITNESSVALUE(nextFS) − FITNESSVALUE(currentFS)
       if (ΔE > 0)
           currentFS ← nextFS
       else  // replace FS with given probability
           REPLACE(currentFS, nextFS, e^(ΔE/Temperature))
end
```

Fig. 1. Feature Set optimization based on a Simulated Annealing procedure.

Refinement of concept description is language specific. E.g. for the case of $\mathcal{ALC}$ logic, refinement operators have been proposed in [13, 8].

3 Grouping Individuals by Hierarchical Clustering

The conceptual clustering procedure that we propose can be ascribed to the category of the heuristic partitioning algorithms such as K-MEANS and EM [9]. For the categorical nature of the assertions on individuals the notion of mean is replaced by the one of medoid, as in PAM (*Partition Around Medoids* [10]). Besides it is crafted to work iteratively to produce a hierarchical clustering.

Indeed, the algorithm implements a top-down bisecting method, starting with one universal cluster grouping all instances. Iteratively, it creates two new clusters by bisecting an existing one and this continues until the desired number of clusters is reached. This algorithm can be thought as levelwise producing a dendrogram: the number of levels coincides with the number of clusters.

Each cluster is represented by one of its individuals. As mentioned above, we consider the notion of medoid as representing a cluster center since our distance measure works on a categorical feature-space. The medoid of a group of individuals is the individual that has the lowest distance w.r.t. the others. Formally. given a cluster $C = \{a_1, a_2, \ldots, a_n\}$, the medoid is defined:

$$m = \text{medoid}(C) = \operatorname*{argmin}_{a \in C} \sum_{j=1}^{n} d(a, a_j)$$

The proposed method can be considered as a hierarchical extension of PAM. A bi-partition is repeated level-wise producing a dendrogram. Fig. 2 reports a sketch of our algorithm. It essentially consists of two nested loops: the outer one computes a new level of the resulting dendrogram and it is repeated until the desired number of clusters is obtained (which corresponds to the final level; the inner loop consists of a run of the PAM algorithm at the current level.

Per each level, the next worst cluster is selected (SELECTWORSTCLUSTER() function) on the grounds of its quality, e.g. the one endowed with the least average inner similarity (or cohesiveness [16]). This cluster is candidate to being splitted. The partition is constructed around two medoids initially chosen (SELECTMOSTDISSIMILAR() function) as the most dissimilar elements in the cluster and then iteratively adjusted in the inner loop. In the end, the candidate cluster is replaced by the newly found parts at the next level of the dendrogram.

The inner loop basically resembles to a 2-MEANS algorithm, where medoids are considered instead of means that can hardly be defined in symbolic computations. Then, the standard two steps are performed iteratively:

distribution given the current medoids, distribute the other individuals to either partition on the grounds of their distance w.r.t. the respective medoid;

center re-computation given the bipartition obtained by DISTRIBUTE(), compute the new medoids for either cluster.

The medoid tend to change at each iteration until eventually they converge to a stable couple (or when a maximum number of iterations have been performed).

3.1 From Clusters to Concepts

Each node of the tree (a cluster) may be labeled with an intensional concept definition which characterizes the individuals in the given cluster while discriminating those in the twin cluster at the same level. Labeling the tree-nodes with concepts can be regarded as solving a number of supervised learning problems in the specific multi-relational representation targeted in our setting. As such it deserves specific solutions that are suitable for the DL languages employed.

A straightforward solution may be found, for DLs that allow for the computation of (an approximation of) the *most specific concept* (msc) and *least common subsumer* (lcs) [1], such as $\mathcal{ALN}$, $\mathcal{ALE}$ or $\mathcal{ALC}$. This may involve the following steps:
given a cluster of individuals node_j

- **for each** individual $a_i \in \mathsf{node}_j$ **do**
  compute $M_i \leftarrow \mathsf{msc}(a_i)$ w.r.t. $\mathcal{A}$;
- **let** $\mathsf{MSCs}_j \leftarrow \{M_i \mid a_i \in \mathsf{node}_j\}$;
- **return** $\mathsf{lcs}(\mathsf{MSCs}_j)$

As an alternative, algorithms for learning concept descriptions expressed in DLs may be employed [13, 8]. Indeed, concept formation can be cast as a supervised learning problem: once the two clusters at a certain level have been found, where the members of a cluster are considered as positive examples and the members of the dual cluster as negative ones. Then any concept learning method which can deal with these representations may be utilized for this new task.

```
clusterVector HIERARCHICALBISECTINGAROUNDMEDOIDS(allIndividuals, k, maxIterations)
input  allIndividuals: set of individuals
       k: number of clusters;
       maxIterations: max number of inner iterations;
output clusterVector: array [1..k] of sets of clusters

begin
level ← 0;
clusterVector[1] ← allIndividuals;
repeat
       ++level;
       cluster2split ← SELECTWORSTCLUSTER(clusterVector[level]);
       iterCount ← 0;
       stableConfiguration ← false;
       (newMedoid1,newMedoid2) ← SELECTMOSTDISSIMILAR(cluster2split);
       repeat
           ++iterCount;
           (medoid1,medoid2) ← (newMedoid1,newMedoid2);
           (cluster1,cluster2) ← DISTRIBUTE(cluster2split,medoid1,medoid2);
           newMedoid1 ← MEDOID(cluster1);
           newMedoid2 ← MEDOID(cluster2);
           stableConfiguration ← (medoid1 = newMedoid1) ∧ (medoid2 = newMedoid2);
       until stableConfiguration ∨ (iterCount = maxIterations);
       clusterVector[level+1] ← REPLACE(cluster2split,cluster1,cluster2,clusterVector[level]);
until (level = k);
end
```

Fig. 2. The HIERARCHICAL BISECTING AROUND MEDOIDS Algorithm.

3.2 Discussion

An adaptation of a PAM algorithm has several favorable properties. Since it performs clustering with respect to any specified metric, it allows a flexible definition of similarity. This flexibility is particularly important in biological applications where researchers may be interested, for example, in grouping correlated or possibly also anti-correlated elements. Many clustering algorithms do not allow for a flexible definition of similarity, but allow only Euclidean distance in current implementations.

In addition to allowing a flexible distance metric, a PAM algorithm has the advantage of identifying clusters by the medoids. Medoids are robust representations of the cluster centers that are less sensitive to outliers than other cluster profiles, such as the cluster means of K-MEANS. This robustness is particularly important in the common context that many elements do not belong exactly to any cluster, which may be the case of the membership in DL knowledge bases, which may be not ascertained given the OWA.

The representation of centers by means of medoids has two advantages. First, it presents no limitations on attributes types, and, second, the choice of medoids is dictated by the location of a predominant fraction of points inside a cluster and, therefore, it is

Table 1. Ontologies employed in the experiments.

ontology	DL	#concepts	#obj. prop.	#data prop.	#individuals
FSM	$\mathcal{SOF}(D)$	20	10	7	37
S.-W.-M.	$\mathcal{ALCOF}(D)$	19	9	1	115
TRANSPORTATION	$\mathcal{ALC}$	44	7	0	250
FINANCIAL	$\mathcal{ALCIF}$	60	17	0	652
NTN	$\mathcal{SHIF}(D)$	47	27	8	676

lesser sensitive to the presence of outliers. In K-MEANS case a cluster is represented by its centroid, which is a mean (usually weighted average) of points within a cluster. This works conveniently only with numerical attributes and can be negatively affected by a single outlier.

4 Experimental Validation

An experimental session was planned in order to prove the method feasible. It could not be a comparative experimentation since, to the best of our knowledge no other hierarchical clustering method has been proposed which is able to cope with DLs representations (on a semantic level) except [11, 7] which are language-dependent and produce non-hierarchical clusterings.

For the experiments, a number of different ontologies represented in OWL were selected, namely: FSM, SURFACE-WATER-MODEL, TRANSPORTATION and NEWTESTAMENTNAMES from the Protégé library[2], the FINANCIAL ontology[3] employed as a testbed for the PELLET reasoner. Table 1 summarizes important details concerning the ontologies employed in the experimentation. For each individual, a variable number of assertions was available in the KB.

As pointed out in several surveys on clustering, it is better to use a different criterion for clustering (e.g. for choosing the candidate cluster to bisection) and for assessing the quality of a cluster. For the evaluation we employed standard validity measures for clustering: the mean square error (WSS, a measure of cohesion) and the *silhouette* measure [10]. Besides, we propose a the extension of Dunn's validity index for clusterings produced by the hierarchical algorithm [2]. Namely, we propose a modified version of Dunn's index to deal with medoids. Let $P = \{C_1, \ldots, C_k\}$ be a possible clustering of n individuals in k clusters. The index can be defined:

$$V_{GD}(P) = \min_{1 \leq i \leq k} \left\{ \min_{\substack{1 \leq j \leq k \\ i \neq j}} \left\{ \frac{\delta_p(C_i, C_j)}{\max_{1 \leq h \leq k} \{\Delta_p(C_h)\}} \right\} \right\}$$

[2] `http://protege.stanford.edu/plugins/owl/owl-library`

[3] `http://www.cs.put.poznan.pl/alawrynowicz/financial.owl`

where δ_p is the Hausdorff distance for clusters[4] derived from d_p and the cluster diameter measure Δ_p is defined:

$$\Delta_p(C_h) = \frac{2}{|C_h|}\left(\sum_{c \in C_h} d_p(c, m_h)\right)$$

which is more noise-tolerant w.r.t. other standard measures.

For each populated ontology, the experiments have been repeated for varying numbers k of clusters (5 through 20). In the computation of the distances between individuals (the most time-consuming operation) all concepts in the ontology have been used for the committee of features, thus guaranteeing meaningful measures with high redundance. The PELLET reasoner[5] was employed to compute the projections. An overall experimentation of 16 repetitions on a dataset took from a few minutes to 1.5 hours on a 2.5GhZ (512Mb RAM) Linux Machine.

The outcomes of the experiments are reported in Fig. 3. For each ontology, we report the graph for Dunn's, Silhouette and WSS indexes, respectively, at increasing values of k (number of clusters searched, which determines the stopping condition).

Particularly, the decay of Dunn's index may be exploited as a hint on possible cut points (the *knees*) in the hierarchical clusterings (i.e. optimal values of k).

It is also possible to note that the silhouette values, as absolute clustering quality measures, are quite stably close to the top of the scale (1). This gives a way to assess the effectiveness of our algorithms w.r.t. others, although applied to different representations.

Conversely, the cohesion coefficient WSS may vary a lot, indicating that for some level the clustering found by the algorithm, which proceeds by bisection of the worst cluster in the previous level, is not the natural one, and thus is likely to be discarded.

5 Conclusions

This work has presented a clustering for (multi-)relational representations which are standard in the SW field. Namely, it can be used to discover interesting groupings of semantically annotated resources in a wide range of concept languages.

The method exploits a novel dissimilarity measure, that is based on the resource semantics w.r.t. a number of dimensions corresponding to a committee of features represented by a group of concept descriptions (discriminating features). The algorithm, is an adaptation of the classic bisecting k-means to complex representations typical of the ontology in the SW.

Currently we are investigating evolutionary clustering methods both for performing the optimization of the feature committee and for clustering individuals automatically discovering an optimal number of clusters.

[4] The metric δ_p is defined, given any couple of clusters (C_i, C_j), $\delta(C_i, C_j) = \max\{d_p(C_i, C_j), d_p(C_j, C_i)\}$, where $d_p(C_i, C_j) = \max_{a \in C_i}\{\min_{b \in C_j}\{d_p(a, b)\}\}$.

[5] `http://pellet.owldl.com`

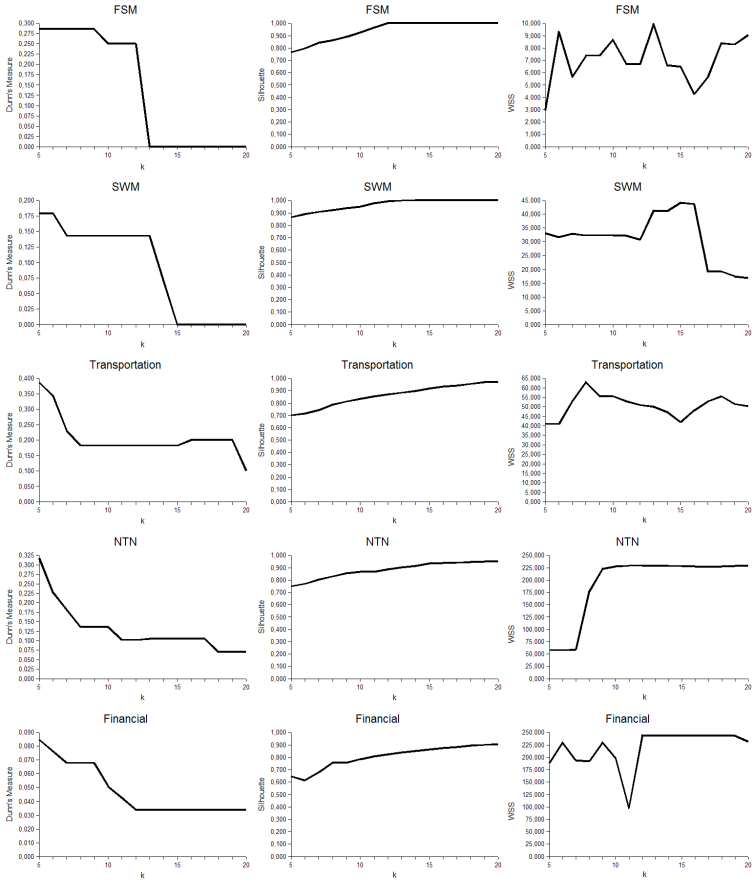


Fig. 3. Outcomes of the experiments: Dunn's, Silhouette, and WSS index graphs.

References

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[2] J.C. Bezdek and N.R. Pal. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics*, 28(3):301–315, 1998.

[3] A. Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1-2).

[4] A. Borgida, T.J. Walsh, and H. Hirsh. Towards measuring similarity in description logics. In I. Horrocks, U. Sattler, and F. Wolter, editors, *Working Notes of the International Description Logics Workshop*, volume 147 of *CEUR Workshop Proceedings*, Edinburgh, UK, 2005.

[5] C. d'Amato, N. Fanizzi, and F. Esposito. Reasoning by analogy in description logics through instance-based learning. In G. Tummarello, P. Bouquet, and O. Signore, editors, *Proceedings of Semantic Web Applications and Perspectives, 3rd Italian Semantic Web Workshop, SWAP2006*, volume 201 of *CEUR Workshop Proceedings*, Pisa, Italy, 2006.

[6] W. Emde and D. Wettschereck. Relational instance-based learning. In L. Saitta, editor, *Proceedings of the 13th International Conference on Machine Learning, ICML96*, pages 122–130. Morgan Kaufmann, 1996.

[7] N. Fanizzi, L. Iannone, I. Palmisano, and G. Semeraro. Concept formation in expressive description logics. In J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, editors, *Proceedings of the 15th European Conference on Machine Learning, ECML2004*, volume 3201 of *LNAI*, pages 99–113. Springer, 2004.

[8] L. Iannone, I. Palmisano, and N. Fanizzi. An algorithm based on counterfactuals for concept learning in the semantic web. *Applied Intelligence*, 26(2):139–159, 2007.

[9] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[10] L. Kaufman and Rousseeuw. P.J. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[11] J.-U. Kietz and K. Morik. A polynomial approach to the constructive induction of structural knowledge. *Machine Learning*, 14(2):193–218, 1994.

[12] M. Kirsten and S. Wrobel. Relational distance-based clustering. In David Page, editor, *Proceedings of the 8th International Workshop, ILP98*, volume 1446 of *LNCS*, pages 261–270. Springer, 1998.

[13] J. Lehmann. Concept learning in description logics. Master's thesis, Dresden University of Technology, 2006.

[14] S.-H. Nienhuys-Cheng. Distances and limits on herbrand interpretations. In D. Page, editor, *Proceedings of the 8th International Workshop on Inductive Logic Programming, ILP98*, volume 1446 of *LNAI*, pages 250–260. Springer, 1998.

[15] M. Sebag. Distance induction in first order logic. In S. Džeroski and N. Lavrač, editors, *Proceedings of the 7th International Workshop on Inductive Logic Programming, ILP97*, volume 1297 of *LNAI*, pages 264–272. Springer, 1997.

[16] R. E. Stepp and R. S. Michalski. Conceptual clustering of structured objects: A goal-oriented approach. *Artificial Intelligence*, 28(1):43–69, Feb. 1986.

Nonlinear knowledge in learning models

Mario R. Guarracino[1], Danilo Abbate[1], and Roberto Prevete[2]

[1] High Performance Computing and Networking Institute
{mario.guarracino,danilo.abbate}@na.icar.cnr.it
National Research Council, Italy
[2] University of Naples Federico II
prevete@na.infn.it

Abstract. For most real life problems it is difficult to find a classifier with optimal accuracy. This motivates the rush towards new classifiers that can take advantage of the experience of an expert. In this paper we propose a method to include nonlinear prior knowledge in Generalized Eigenvalues Support Vector Machine. The expression of nonlinear kernels and nonlinear knowledge as a set of linear constraints allows us to have a nonlinear classifier which has a lower complexity and halves the misclassification error with respect to the original generalized eigenvalues method. The Wisconsin Prognostic Breast Cancer data set is used as a case study to analyze the performance of our approach, comparing our results with state of the art SVM classifiers. Sensitivity and specificity results for some publicly available data sets well compare with the other considered methods.

1 Introduction

Given a set A of data and two classes, -1 and $+1$, the purpose of a binary classifier is to divide such a set in two partitions, so that each data can be assigned to the correct class, according to some discriminant features. A tasks which typically involves the use of binary classification is medical diagnosis, to verify, for example, whether a patient has a given disease or not. In this case the class label is related to the presence/absence of a disease. From a mathematical point of view, given a set of points $\Gamma \subset \mathbb{R}^n$ a binary classifier is a function $f(x) : \mathbb{R}^n \to \mathbb{R}$, whose sign represents the class the point belongs to.

Examples of classifiers are neural networks [2], decision trees and support vector machines (SVM) [3]. The performance of a binary classifier can be evaluated through *misclassification error*, *sensitivity* and *specificity*. Misclassification error represents the percentage of misclassified samples. Sensitivity is the percentage of true positives, among all positives tested. Specificity is the percentage of true negatives among all negatives tested. Real world problems deal with irregular data for which accuracy is not optimal. This motivates the rush towards the design of new classifiers. Medical data sets are practical examples of data sets hard to classify. Having a better classification accuracy on medical data can have a drastic impact both on the quality of life of a patient and on the promptness of diagnosis. Sending a patient to the most appropriate medical cures, or

identifying whether a set of patients will have a recurrence of a breast cancer, can sensibly improve patient's lifestyle and diagnosis rapidity.

The case study analyzed in this work will is the WPBC data set (Wisconsin Prognostic Breast Cancer, [4]). The version of the data set we are using contains medical data of 155 patients which underwent surgery for the removal of metastasized lymph nodes. WPBC is an example of a data set for which is difficult to improve classification accuracy, since patients belonging to different classes have many attributes with close values. It is really difficult to reach an accuracy higher than 80% without adding the prior knowledge of a field expert. On WPBC, 18.06% of the points belong to the class +1. In this case, the classifier will most probably be a trivial classifier, completely influenced by the points in class −1, which are the 81.94% of the whole data set. If training set were composed of a large number of samples, there could be an overfitting problem. Training a classifier with too many data would thus be risky, as the classifier could perfectly classifies training data, but it would not generalize.

A natural approach is to plug a priori knowledge in a classifier adding more points to the data set. This results in higher computational complexity and overfitting. On the other hand, Mangasarian [8] has shown that it is possible to analitically express knowledge as additional terms of the cost function of the optimization problem defining SVM. This solution has the advantage not to increase the dimension of the training set, thus to avoid overfitting and poor generalization of the classification model [2].

In this paper we show how nonlinear knowledge can be extended to Generalized Proximal SVM (GEPSVM) [9]. This method is based on generalized eigenvalue computation and, as it will be explained in the following, its computational properties make it a good candidate to understand how prior knowledge can be used to improve classification methods. We show that the proposed model with prior knowledge can substantially increase the original GEPSVM accuracy, providing results that well compare with those reported in [10].

Hereafter we use the following notation, respecting the following guidelines. All vectors are column vectors, if they are not transposed by a prime $'$. Inner product, or scalar product, of two vectors x and y in the n-dimensional real space $\mathbb{R}^n$ is indicated by $x'y$. Given a vector $x \in \mathbb{R}^n$, $\|x\|_1$ denotes the 1-norm: $(\sum_{i=1}^{n} |x_i|)$ while $\|x\|$ denotes the 2-norm: $\left(\sum_{i=1}^{n} (x_i)^2\right)^{\frac{1}{2}}$. The transpose of the matrix $A \in \mathbb{R}^{m \times n}$ is A^T, while A_i and $A_{\cdot j}$ denote respectively the i-th row and the j-th column of matrix A. A vector made up of 1s is indicated as e, so that for $e \in \mathbb{R}^m$ and $y \in \mathbb{R}^m$ $e'y$ represents the sum of all the components of y. A null vector will be denoted by 0.
Given two matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times k}$, a *kernel* $K(A, B)$ maps $\mathbb{R}^{m \times n} \times \mathbb{R}^{n \times k}$ into $\mathbb{R}^{m \times k}$. More precisely, if x and y are column vectors in $\mathbb{R}^n$ then $K(x', y)$ will be a real number in $\mathbb{R}$, $K(x', B)$ a row vector in $\mathbb{R}^m$ and $K(A, B)$ a matrix $m \times k$. A common kernel in nonlinear classification tasks is the gaussian kernel where the ij-th element is defined as follows: $(K(A, B))_{ij} = \epsilon^{-\mu \left\| A_i' - B_{\cdot j} \right\|^2}$, where A and B are matrices with the same number of columns, μ is a positive

constant and ϵ is the Napier's constant, the base of the natural logarithm. As we deal with classification problems, each point $x \in \mathbb{R}^n$ is assigned to a class c, with $c \in \{-1, 1\}$. So, for set Γ of m real points, which can be represented by a matrix $A \in \mathbb{R}^{m \times n}$, there is an associated a vector $c \in \{-1, 1\}^m$ of labels denoting the class of each point of the set.
The remainder of this paper is organized as follows. Section 2 describes how nonlinear knowledge is added to Smooth SVM. In Section 3, we present a new algorithm based on GEPSVM, which includes nonlinear knowledge as a set of linear inequalities. In Section 4, numerical results are reported on the WPBC case study. In Section 5, experimental results are provided for more data sets, and finally, in Section 6, conclusions are drawn and future work is proposed.

2 Related work

The state of the art in binary classification is represented by SVM (*Support Vector Machines*) [15, 3]. SVMs separate the input space in two halfspaces finding the hyperplane $x'w - \gamma = 0$ which maximizes the margin between the two classes. The margin is the distance from the hyperplane of the closest point. The hyperplane is actually found by minimizing the norm of w, with constraints to correctly classify points of both classes.
Using the kernel trick, it is possible to obtain a nonlinear separating surface that correctly classifies nonlinearly separable classes, still working on a linear program [10]. Thus the resulting hyperplane, projected in the feature space [14], has equation:

$$f(x) \equiv K(x', B^T)u - \gamma \ = 0, \tag{1}$$

where $B \in \mathbb{R}^{k \times n}$ and $K(x', B^T) : \mathbb{R}^{1 \times n} \times \mathbb{R}^{n \times k} \rightarrow \mathbb{R}^{1 \times k}$ is an arbitrary kernel function. Parameters $u \in \mathbb{R}^k$ and $\gamma \in \mathbb{R}$ are determined solving the following quadratic optimization problem [8]:

$$\begin{array}{ll} \underset{u,y}{min} & e'y + \dfrac{1}{2}u'u \\ s.t. & D(K(\Gamma, \Gamma^T)u - e\gamma) + y \geq 0, \ y \geq 0. \end{array} \tag{2}$$

where D is a diagonal matrix, with the diagonal elements equal to the labels of the corresponding element of the training set (matrix Γ). Such condition places the points belonging to the two classes $+1$ and -1, represented by the matrix Γ, on two different sides of the nonlinear separation surface (1). Problem (2) corresponds to the following linear programming problem [10]:

$$\begin{array}{lll} \underset{u,\gamma,y,s}{min} & \nu e'y + e's & \\ s.t. & D(K(\Gamma, \Gamma^T)u - \gamma e) + y & \geq e, \\ & -s \leq u & \leq s, \\ & y & \geq 0. \end{array} \tag{3}$$

The nonlinear classification model cannot describe the discriminating function in terms of inequalities involving linear relations among features. This can

be perceived as a problem in case of medical diagnosis, in which doctors prefer to find simple correlations between the results of a clinic exams and the diagnosis or prognosis of an illness. On the other hand, it is generally accepted that results achieved by nonlinear models provide higher classification accuracy. Furthermore, with the advent of high throughput medical equipments, the number of exams to consider for a diagnosis can be very high and cannot be correlated only with the experience. Finally, methods that provide explicit classification rules are not guaranteed to find a set of rules small enough to be easy readable.

In order to improve the results obtained by a classifier solely from the training set it is possible to impose the knowledge of an expert into the learning phase of the function (1). Such expertise is represented by the following implication which represents a region Δ in the input space in which points x are known to belong to class +1:

$$g(x) \leq 0 \Rightarrow K(x', B^T)u - \gamma \geq \alpha, \forall x \in \Delta. \tag{4}$$

$g(x) : \Delta \subset \mathbb{R}^n \rightarrow \mathbb{R}$ is a function defined on the subset $\Delta \subset \mathbb{R}^n$ where prior knowledge imposes to the classification model $K(x', B^T)u - \gamma$ to be greater than, or equal to, a non negative number α, to classify points $x \in \{x|g(x) \leq 0\}$ as belonging to class +1.

Given the theorem of the alternatives for a convex function, implication (4) can be expressed as a linear inequality in the parameters (u, γ) of the classification model:

Theorem 1 (Mangasarian, 2006 [10]). *Given $u \in \mathbb{R}^k$, $\gamma \in \mathbb{R}$, if there is a $v \in \mathbb{R}$, $v \geq 0$ such that:*

$$K(x', \Gamma^T)u - \gamma - \alpha + v'g(x) \geq 0, \forall x \in \Delta \tag{5}$$

then the implication (4) holds.

Finally, to add positive nonlinear knowledge to problem (3) using Theorem 1:

$$\begin{array}{ll} \underset{u,\gamma,y}{min} & \nu e'y + e's \\ s.t. & D(K(A, B^T)u - \gamma e) + y \geq e, \\ & -s \leq u \leq s,\ y \geq 0, \\ & K(x_i', B^T)u - \gamma - \alpha + v'g(x_i) + z_i \geq 0, \\ & v \geq 0,\ z_i \geq 0, \quad i = 1, \ldots, l. \end{array} \tag{6}$$

To add negative nonlinear knowledge just consider the following implication:

$$h(x) \leq 0 \Rightarrow K(x', B^T)u - \gamma \leq -\alpha, \forall x \in \Lambda \tag{7}$$

where $h(x) : \Lambda \subset \mathbb{R}^n \rightarrow \mathbb{R}$ represents the region in the input space where implication (7) forces the classification function to be less than or equal to $-\alpha$, in order to classify the points $x \in \{x|h(x) \leq 0\}$ as -1.

Now we can finally formulate the linear program (3) with nonlinear knowledge included in the cost function:

$$\begin{aligned}
\min_{u,\gamma,y,s,v,p,z_1,\ldots,z_l,q_1,\ldots,q_t} \quad & \nu e'y + e's + \sigma(\sum_{i=1}^{l} z_i + \sum_{j=1}^{t} q_j) \\
s.a. \quad & D(K(A,B^T)u - \gamma e) + y \geq e, \\
& -s \leq u \leq s,\ y \geq 0, \\
& K(x_i',B^T)u - \gamma - \alpha + v'g(x_i) + z_i \geq 0, \\
& v \geq 0,\ z_i \geq 0,\ i = 1,\ldots,l, \\
& -K(x_j',B^T)u + \gamma - \alpha + p'g(x_j) + q_j \geq 0, \\
& p \geq 0,\ q_j \geq 0,\ j = 1,\ldots,t.
\end{aligned} \tag{8}$$

The LP problem (8) minimizes the margin between the two classes constraining the classification problem to leave the two a priori knowledge sets in the corresponding halfspace.

3 Nonlinear knowledge in GEPSVM

We recall that a SVM binary classifier finds a hyperplane which divides the space into two halfspaces. Points laying in one halfspace belong to class $+1$, the others to class -1. A different approach is used in Proximal Support Vector Machines [5], where the class of a point is determined by the closeness to one of two hyperplanes.

Given matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{k\times n}$, respectively representing points of class $+1$ and -1, $\Gamma = A \cup B$, we can find the hyperplane approximating the class $+1$ solving the following minimization task:

$$\min_{u,\gamma\neq 0} \frac{\|K(A,\Gamma)u - e\gamma\|^2}{\|K(B,\Gamma)u - e\gamma\|^2}, \tag{9}$$

which finds the hyperplane minimizing the distance from the points of class $+1$ and at the same time maximizing the distance from the points of class -1. Conversely, the hyperplane for points in class -1, can be found solving the reciprocal of (9):

$$\min_{u,\gamma\neq 0} \frac{\|K(B,\Gamma)u - e\gamma\|^2}{\|K(A,\Gamma)u - e\gamma\|^2}. \tag{10}$$

Equation (10) finds the hyperplane minimizing distance from points in class -1 and maximizing distance from points in class -1. Each of these problems can be solved using regularization as proposed in [7]:

$$\min_{u,\gamma\neq 0} \frac{\|K(A,\Gamma)u - e\gamma\|^2 + \delta \left\|\widetilde{K}_B u - e\gamma\right\|^2}{\|K(B,\Gamma)u - e\gamma\|^2} \tag{11}$$

$$\min_{u,\gamma\neq 0} \frac{\|K(B,\Gamma)u - e\gamma\|^2 + \delta \left\| \widetilde{K}_A u - e\gamma \right\|^2}{\|K(A,\Gamma)u - e\gamma\|^2} \tag{12}$$

where $\widetilde{K}_A$ and $\widetilde{K}_B$ are diagonal matrices, with diagonal elements taken respectively from matrices $K(A,\Gamma)$ and $K(B,\Gamma)$.

Given

$$\begin{aligned} G &= [K(A,\Gamma) \ -e]^T [K(A,\Gamma) \ -e], \\ H &= [K(B,\Gamma) \ -e]^T [K(B,\Gamma) \ -e], \\ z &= [u' \quad \gamma]', \end{aligned} \tag{13}$$

equation (9) becomes:

$$\min_{z\in\mathbb{R}^m} \frac{z'Gz}{z'Hz}. \tag{14}$$

Similarly for B, we obtain the reciprocal problem:

$$\min_{z\in\mathbb{R}^m} \frac{z'Hz}{z'Gz}. \tag{15}$$

Equations (14) and (15) represent Rayleigh quotients of the eigenvalue problems $Gz = \lambda Hz$ and its reciprocal.
The minimum eigenvectors obtained as solution to (11,12) give the proximity planes $P_i, 1 = 1, 2$. A given point x will thus be classified according to the following formula:

$$class(x) = \underset{i=1,2}{argmin} \{dist(x, P_i)\}, \tag{16}$$

using the distance

$$dist(x, P_i) = \frac{|K(x,\Gamma)u - \gamma|}{\|u\|}. \tag{17}$$

GEPSVM algorithm has several advantages with respect to SVM. First of all, in its linear formulation, it can be used to classify problems that are not linearly separable. Furthermore, its computational complexity is dominated by the number of training samples. Finally, its implementatino is reduced to eigenpairs computation, which can be expressed in a single line code in many problem solving environments such as R and Matlab.

It is possibile to add nonlinear prior knowledge formulating the model in terms of a constrained generalized eigenvalue problems. The latter has been extensively studied and a procedure for its solution has been proposed by Golub in [6].

If G and H, as defined in (14), are symmetric matrices of order n, constraints can be expressed by the equation:

$$C^T z = 0, \tag{18}$$

where C is a $n \times p$ matrix of rank r, with $r < p < n$. The constrained eigenvalue problem for the classification surface for points in class +1 is:

$$\begin{array}{ll} \underset{z \in \mathbb{R}^m}{min} & \dfrac{z'Gz}{z'Hz} \\ s.t. & C^T z = 0. \end{array} \tag{19}$$

Let Δ be the set of class +1 points describing nonlinear knowledge, constraint matrix C must represent knowledge imposed on class +1 points, hence it will be:

$$C = \begin{bmatrix} K(\Delta, \Gamma) & -e \end{bmatrix}^T \tag{20}$$

Matrix C needs to be rank deficient in order to have non-trivial solution. The set of constraints 18 requires all points in Δ to have null distance from the hyperplane, and thus to belong to class +1.

The QR decomposition of C gives two matrices Q and R such that $C = QR$. Q is an orthonormal matrix where $Q^TQ = I$. R is an order r upper triangular matrix. Reordering the rows of C, we can write:

$$Q^T C = \begin{bmatrix} R & S \\ 0 & 0 \end{bmatrix}$$

where S is a $r \times (p - r)$ matrix. Let

$$z = Qw = Q \begin{bmatrix} y \\ v \end{bmatrix}$$

where y is a vector of the first r components of w and v of the last $(n - r)$ components of w, thus having a representation of z in the space generated by Q. We have:

$$C^T z = \begin{bmatrix} R^T & 0 \\ S^T & 0 \end{bmatrix} \begin{bmatrix} y \\ v \end{bmatrix} = 0$$

and hence $y = 0$. Defining $z = Qw$ it is possible to reformulate equation (14) as:

$$\underset{w \neq 0}{min} \quad \frac{w' \; Q^T \; G \; Q \; w}{w' \; Q^T \; H \; Q \; w}.$$

To simplify, we let $L = Q^T G Q$ and $M = Q^T H Q$, with

$$L = \begin{bmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{bmatrix}, M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$$

where L_{11}, M_{11} are $r \times r$ matrices and L_{22}, M_{22} are $(n-r) \times (n-r)$ matrices. Both L and M are symmetric matrices. Moreover, being matrix M positive definite we have

$$0 < \lambda_{min}(M) \leq \lambda_{min}(M_{22}) \leq \lambda_{max}(H_{22}) \leq \lambda_{max}(H) \tag{21}$$

where λ_{min} and λ_{max} represent minimum and maximum eigenvalues. Leading back to a minimization problem, we have to find w such that:

$$\min_{w \neq 0} \quad \frac{w'Lw}{w'Mw}. \tag{22}$$

Minimization problem (22) contains positive nonlinear knowledge represented by C. This expression is Rayleigh quotient of the generalized eigenvalue problem $Lw = \lambda Mw$. Stationary points are those and only those corresponding to the eigenvectors of (22). Moreover, being M positive definite, Rayleigh quotient is limited and varies in the interval determined by minimum and maximum eigenvalues [13]. Considering (21), we just need to search stationary values of the equation:

$$L_{22}\ v = \lambda M_{22} v. \tag{23}$$

Being L and M symmetric and M positive definite, also L_{22} and M_{22} will be symmetric, and M_{22} positive definite.

So far, having found the $n - r$ eigenvalues and eigenvectors of $L_{22}v_i = \lambda_i M_{22} v_i$, $i = 1 \ldots n - r$, we calculate the components of the vector z, original solution of the problem (19):

$$w_i = Q \begin{bmatrix} 0 \\ \cdots \\ I_{n-r} \end{bmatrix} v_i. \tag{24}$$

The constrained method just introduced has a lower complexity, compared to the original method, in the solution of the eigenvalue problem (23), which involves matrices of order $(n-r)$, although an initial QR factorization is needed for the matrix C.

4 A case study

The above method has been tested on a publicly available data set, the Wisconsin Prognostic Breast Cancer, from UCI repository [4]. Different methods are compared using misclassification accuracy, sensitivity and specificity. Results for SVM are taken from [10], while results for GEPSVM are calculated using a GNU/linux PC, kernel 2.6.9-42 with AMD Opteron 64 bits of the series 284 (2.2GHz), 4 Gigabyte RAM. The version of Matlab [11] used is 7.3.0.298($R2006b$). Accuracy, sensitivity and specificity are calculated upon Leave-one-out classification, where the parameters for each method are defined through a ten-fold cross validation grid-search over the whole data set. GEPSVM algorithm is implemented in Matlab, using *eig* function for solving the generalized eigenvalue problems, while Mangasarian's SVM results are computed using MATLAB function *linprog*.
The data set provides 30 cytological features plus tumor size and the number of metastasized lymph nodes. Moreover, for each of the 198 patient, it provides the number of months after which the patient has been diagnosed a new cancer. If

there has been no recurrence, the data set contains information on how long the patient has been under analysis.

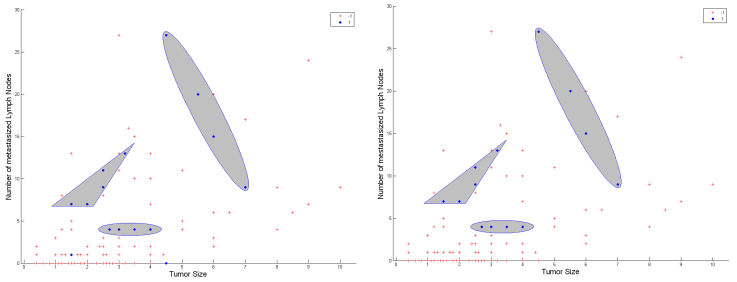
In our work, we want to identify those patients which had a recurrence in a period of 24 months, discriminating them from those which did not have any recurrence. This is a subset *Upsilon* of the data set:

$$\Upsilon = \{x \in WPBC | \text{ property p holds for x}\}$$

where the property p is defined as follows:

$$\text{p holds } \textit{iff}: \begin{cases} \text{the patient has had a recurrence in the 24 months period,} \\ \text{the patient had not recurrence.} \end{cases}$$

After this filtering, the remaining data set contains 155 patients.



(a) GEPSVM results with nonlinear knowledge areas highlighted

(b) SVM results with nonlinear knowledge areas highlighted

To simulate the expertise of a surgeon we use the same areas identified by Managasarian in [10] and described by the following formulas:

$$\begin{aligned} \left\| \begin{pmatrix} (5.5)\, x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} (5.5)\, 7 \\ 9 \end{pmatrix} \right\| + \left\| \begin{pmatrix} (5.5)\, x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} (5.5)\, 4.5 \\ 27 \end{pmatrix} \right\| - 23.0509 &\leq 0 \Rightarrow f(x) \geq 1 \\ \begin{pmatrix} -x_2 + 5.7143 x_1 - 5.75 \\ x_2 - 2.8571 x_1 - 4.25 \\ -x_2 + 6.75 \end{pmatrix} &\leq 0 \Rightarrow f(x) \geq 1 \\ \tfrac{1}{2} \left(x_1 - 3.35\right)^2 + \left(x_2 - 4\right)^2 - 1 &\leq 0 \Rightarrow f(x) \geq 1. \end{aligned} \tag{25}$$

Equations (25) describe three areas in a two dimensional representation of the data set. The x-axis is the tumor size (the next to last feature of the data set) while the y-axis is the number of metastasized lymph nodes (the last feature of the data set). In accordance with Mangasarian we decided to take those 14 points which belong to the three areas described by equations (25). We note that

those 14 points are among the support vectors that have been misclassified by SVM in leave one out (LOO) validation.

So far, the lowest accuracy error without knowledge is 13.7% [1]. Adding knowledge it is possible to decrease misclassification rate to around 9.0% (Table 1).

Classifier	Misclassification error	Sensitivity	Specificity
SVM	0.1806	0	1.000
SVM with knowledge	0.0903	0.5000	1.000
GEPSVM	0.1806	0	1.000
GEPSVM with knowledge	0.0903	0.5172	0.9930
Improvement due to knowledge	50.0%		

Table 1. Leave One Out misclassification percentage, sensitivity and specificity on WPBC data set (24 months).

In Table 1 we report missclassification error percentage, sensitivity and specificity for SVM and GEPSVM with and without knowledge. We note that without knowledge both methods produce trivial results.

When knowledge is used, both methods have the same accuracy, but GEPSVM has different values of sensitivity and specificity. These are due to the fact that it not only correctly classifies the knowledge points, but it also adds two more points to class $+1$, one of which is misclassified. It is important to point out that, with respect to SVM, there is one more patient, not belonging to those taken into account by knowledge, which has been correctly classified by the new method. This is depicted in Figures 1(a) and 1(b). We can analyze our approach in terms of sensitivity and specificity as introduced in Section 1. The accuracy of the classifier, with respect to class -1 is measured in terms of specificity. In order to avoid sending a healthy patient to surgery, we need to reach the highest specificity we can. With our approach, specificity decreases of 0.7%. This may seems inappropriate but considering sensitivity we realize that the number of unhealthy patients correctly recognized is increased by 1.72%. Thus, while we see a little decrease in specificity, it is counterbalanced by a growth in sensitivity which is more than the double in percentage. This means that, with respect to SVM, we identify a greater number of patients that may need surgery.

5 Numerical experiments

To further asses the classification accuracy of GEPSVM with prior knowledge, we have used three publicly available data sets [12, 4]: Thyroid, Heart and Banana. Thyroid contains information about 215 patients, with 5 cytological features. There are 65 patients affected by a thyroid disease. They represent 30.23% of

the data set (class +1). Remaining 150 patients belong to class -1. Heart is also a medical data set, with 270 patients and 13 features. 120 patients, 44.44% of the data set, belong to class +1, representing patients affected by heart disease. Remaining 150 patients belong to class -1. Finally, Banana is an artificial data set of 2-dimensional points which are grouped together in a shape of a banana.

We decided to choose points to add in prior knowledge in the following way. For each data set, we have executed a LOO validation and we have chosen the misclassified points that have null distance from the hyperplanes evaluated by GEPSVM on the whole data set. In Table 2 we report classification accuracy, sensitivity and specificity for GEPSVM and its formulation with prior knowledge. In Table 3 we also report the number of points, for each class, used as prior knowledge. We note that the performance of the method is substantially

Data set	Results without knowledge			Results with knowledge		
	Misclassification error	Sensitivity	Specificity	Misclassification error	Sensitivity	Specificity
Thyroid	0.0698	0.8769	0.9533	0.0093	0.9662	1.0000
Heart	0.1926	0.7833	0.8267	0.0852	0.9167	0.9133
Banana	0.1172	0.8261	0.9264	0.0170	0.9696	0.9933

Table 2. *Leave One Out* misclassification error percentage, sensitivity and specificity on different data sets.

Data set	Prior Knowledge points:	
	Class +1	Class -1
Thyroid	8	7
Heart	18	18
Banana	40	22

Table 3. Number of points used as prior knowledge for each data set.

improved in each case. It is interesting to note that the number of true positive and negative points is substantially improved.

6 Conclusions and future work

In the present work we have proposed a new method to incorporate nonlinear knowledge provided by an expert in GEPSVM, in a fashion similar to what has been done in [10]. Results show that accuracy of the new algorithm well

compares with existing ones and it improves with respect to GEPSVM without knowledge.

In future, we will test and compare the method against other standard data sets. Finally, we believe further investigation needs to be devoted to the identification of regions where knowledge is needed, in order to improve generalization of the classification model. We will investigate how the expression of a priori knowledge in terms of probability of a patient to belong to one class can affect classification models.

References

1. K. P. Bennett. Decision tree construction via linear programming. In *Procedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society Conference (Utica, Illinois)(M.Evans,ed.)*, pages 82–90, 1992.
2. C. M. Bishop. *Neural networks for pattern recognition*. Oxford Press, 1995.
3. C. Cortes and V. Vapnik. Support vector machines. *Machine Learning*, 20:273–279, 1995.
4. C.L. Blake D.J. Newman, S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998. http://mlearn.ics.uci.edu/MLRepository.html.
5. G. Fung and O. L. Mangasarian. Proximal support vector machine classifiers. In *Knowledge Discovery and Data Mining*, pages 77–86, 2001.
6. G. H. Golub and R. Underwood. Stationary values of the ratio of quadratic forms subject to linear constraints. *Zeitschrift für Angewandte Mathematik und Physik (ZAMP)*, 21(3):318–326, 1970.
7. M. R. Guarracino, C. Cifarelli, O. Seref, and P. M. Pardalos. A classification algorithm based on generalized eigenvalue problems. *Optimization Methods and Software*, 22(1):73–81, 2007.
8. Y. Lee and O. L. Mangasarian. Ssvm: A smooth support vector machine for classification, 1999.
9. O. L. Mangasarian and E. W. Wild. Multisurface proximal support vector classification via generalized eigenvalues. Technical Report 04-03, Data Mining Institute, September 2004.
10. O. L. Mangasarian and E. W. Wild. Nonlinear knowledge-based classification. Technical report, Data Mining Institute Technical Report 06-04, Computer Science Department, University of Wisconsin, Madison, Wisconsin, November 2006.
11. MATLAB. *User's guide*. The Mathworks, Inc., Natick, MA 01760, 1994–2006. http://www.mathworks.com.
12. S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. Muller. Fisher discriminant analysis with kernels, 1999.
13. B. N. Parlett. *The Symmetric Eigenvalue Problem*. SIAM, Philadelphia, PA, 1998.
14. B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.
15. V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

A study of the SEMINTEC approach to frequent pattern mining

Joanna Józefowska[1], Agnieszka Ławrynowicz[1], and Tomasz Łukaszewski[1]

Institute of Computing Science, Poznan University of Technology,
ul. Piotrowo 2, 60-965 Poznan, Poland
{jjozefowska, alawrynowicz, tlukaszewski}@cs.put.poznan.pl

Abstract. This paper contains the experimental investigation of various settings under an approach to frequent pattern mining in description logics ($\mathcal{DL}$) knowledge bases that we coined SEMINTEC. Frequent patterns in this approach are the conjunctive queries to $\mathcal{DL}$ knowledge base. First, we prove that the approach introduced in our previous publication, for the DLP fragment of $\mathcal{DL}$ family of languages, is also valid for more expressive languages. Next, we present the experimental results on knowledge bases of different sizes and complexities.

1 Introduction

Recent developments in knowledge representation raise the challenges for the research on exploiting new representations in knowledge discovery algorithms. Explicitly and formally represented *prior (background) knowledge*, describing relationships in a given domain, may improve the process and the results of knowledge discovery. Continuous progress in using prior knowledge of different forms can be witnessed within the relational data mining (RDM) research. In RDM methods inductive logic programming (ILP) techniques or just the first-order logic notation are used. RDM approaches usually use Datalog as a representation language. Recently, together with the growing adoption of the Semantic Web technologies, another logic-based formalism, description logics ($\mathcal{DL}$) [1], has gained much attention. Implicit for $\mathcal{DL}$, Open World Assumption (OWA), allowing to handle incomplete information, is especially well-suited for the dynamic and distributed nature of the Semantic Web knowledge bases.

A $\mathcal{DL}$ knowledge base typically consists of terminological (schema) part, called *TBox* and assertional (data) part, called *ABox*. Existing $\mathcal{DL}$ reasoners have been mostly developed to efficiently handle complex TBoxes, not focusing on handling large ABoxes. For the data mining task, handling efficiently large datasets is crucial. Only recently the topic of the performance of the query answering over knowledge bases with large ABoxes has gained more attention. Recently implemented reasoning engine KAON2[1] outperforms other reasoners in case of a high number of instances in the knowledge base and not very complex TBox [13]. It uses the algorithm that reduces a knowledge base $\mathcal{KB}$ in $\mathcal{DL}$ to

[1] http://kaon2.semanticweb.org

a knowledge base in Disjunctive Datalog DD($\mathcal{KB}$) [5], which enables using optimization techniques that proved to be effective in disjunctive databases, such as magic-sets or join-order optimizations. KAON2 also supports the so-called $\mathcal{DL}$*-safe rules* [12], a combination of $\mathcal{DL}$ and function-free Horn rules, where very expressive $\mathcal{DL}$ can be used, while preserving the *decidability* property.

In [6] we discussed the potential of using this combination with the query answering method implemented in KAON2 for frequent pattern discovery in knowledge bases in $\mathcal{DL}$ and containing Horn rules. In [7] we presented an approach to frequent pattern mining in knowledge bases in OWL DLP, the language restricted to the intersection of the expressivity of $\mathcal{DL}$ and Logic Programming. In this paper we prove that our method for building patterns remains correct for more expressive languages from $\mathcal{DL}$ family. We discuss different settings that we implemented under our approach. The results of the experimental investigation under these settings, for different kinds of knowledge bases, are presented.

The rest of the paper is organized as follows. In Section 2 we present the related work. In Section 3 we present the data mining setting. Section 4 contains the overview of our approach. In Section 5 the experimental results are presented and Section 6 concludes the paper.

2 Related work

The ILP methods, in particular RDM ones, are closely related to our work. WARMR [3] was the first relational data mining system proposed for frequent pattern discovery. The initial approach was further refined and optimized [2]. Another RDM system, FARMER [14, 15], instead of ILP techniques, uses only the first-order logic notation and an efficient *trie* data structure. Encouraged by the results achieved by FARMER we decided to use similar structure in our approach. The aforementioned methods use, as a generality relation, the θ-subsumption, an approximation of a logical implication. In turn, the system c-armr [16], also an ILP frequent query miner, implements the refinement operator under generalized subsumption. C-armr induces frequent s-free queries, which do not contain redundant literals, where redundancy is defined relative to a background theory containing the semantic knowledge. Our approach is also based on the generalized subsumption and exploits the background knowledge for pruning redundant patterns.

Datalog allows the interaction of variables in arbitrary ways. $\mathcal{DL}$ is able to represent rich structural knowledge. The combination of the expressive power of $\mathcal{DL}$ and Datalog would be thus desirable. To the best of our knowledge, there have been only one approach, named SPADA [8, 9], that use such an expressive representation for frequent pattern mining. SPADA uses hybrid $\mathcal{AL}$-log [4] language, which combines the $\mathcal{ALC}$ language from $\mathcal{DL}$ family with Datalog rules. In these rules, there can be $\mathcal{ALC}$ concepts as predicates in the unary atom constraints in the body, but any roles. However, in contrast to our approach, patterns in SPADA can contain n-ary Datalog predicates. The $\mathcal{DL}$-safe rules combination supports more expressive $\mathcal{DL}$, and allows using both concepts and roles in

atoms, they may be also used in rule heads. Furthermore the $\mathcal{DL}$-safe rules query answering algorithm, as an extension of deductive database techniques, runs in EXP time while the algorithm for $\mathcal{AL}$-log in single NEXP time. In SPADA the notion of description granularity is exploited, what means that the patterns contain the concepts from demanded levels of a concept taxonomy. In our approach, a pattern may contain concepts from different levels of a taxonomy.

3 Frequent patterns in $\mathcal{DL}$ knowledge bases

3.1 Pattern discovery task

The general formulation of the frequent pattern discovery problem was specified by [10]. It was further extended to deal with more expressive language in case of RDM methods in [3]. With respect to these formulations we define our task as:

Definition 1. *Given*

- *a knowledge base in $\mathcal{DL}$ $\mathcal{KB}$,*
- *a set of patterns in form of queries Q that all contain a reference concept $\hat{C}$,*
- *a minimum support threshold minsup specified by the user*

and assuming that queries with support s are frequent in $\mathcal{KB}$ given $\hat{C}$ if s≥minsup, the task of frequent pattern discovery is to find the set $\mathcal{F}$ of frequent queries.

The $\hat{C}$ parameter determines what is counted. A query atom with the $\hat{C}$ concept contains the only one distinguished variable in the query (*key* variable).

Definition 2. *A support of the query Q with respect to the knowledge base $\mathcal{KB}$ is defined as the ratio between the number of instances of the $\hat{C}$ concept that satisfy the query Q and the total number of instances of the $\hat{C}$ concept (the trivial query for the total number of the instances is denoted Q_{ref}):*

$$support(\hat{C}, Q, KB) = \frac{|answerset(\hat{C}, Q, KB)|}{|answerset(\hat{C}, Q_{ref}, KB)|} \tag{1}$$

3.2 The data mining setting

We assume pattern mining in knowledge bases $\mathcal{KB}$ represented in $\mathcal{SHIF}$ subset of $\mathcal{DL}$ family of languages, which corresponds to OWL-Lite variant of the Web Ontology Language (OWL)[2] (without datatypes). In our approach the intensional background knowledge is represented in a TBox. An ABox contains instances (extensional background knowledge). Our goal is to find frequent patterns in the form of conjunctive, $\mathcal{DL}$-safe queries over $\mathcal{KB}$. The notion of $\mathcal{DL}$-safe queries, that are a part of KAON2 reasoning engine that we use in our approach, was introduced in [12]. $\mathcal{DL}$-safe queries are the conjunctive queries without true non-distinguished variables. All variables in such a query are bound to individuals explicitly occurring in the knowledge base, even if they are not returned as

[2] http://www.w3.org/TR/owl-features/

part of the query answer. The frequent pattern $\mathcal{Q}$ that we look for has the form of the conjunctive, positive $\mathcal{DL}$-safe query over $\mathcal{KB}$, whose answer set contains individuals of the $\hat{C}$ concept and that has the linkedness property. We use the following notation for such patterns (queries):

$$q(key) : -\hat{C}(key), \alpha_1, ..., \alpha_n$$

where q(*key*) denotes that *key* is the only one distinguished variable and α_1,...,α_n represent atoms of the query (with atomic concepts and atomic roles as predicates). A trivial pattern is the query of the form: q(*key*):-$\hat{C}$(*key*). For the *Client* being the $\hat{C}$ concept the following example query can be imagined: *q(key):- Client(key), isOwnerOf(key, x), Account(x), hasLoan(x, y), Loan(y).*

The generality notion that we use in our approach is based on the *query containment*. Given two queries Q_1 and Q_2 to the knowledge base $\mathcal{KB}$ we say that Q_1 is at least as general as Q_2 under query containment, $Q_1 \succeq Q_2$, iff in every possible state of the $\mathcal{KB}$ the answer set of the Q_2 is contained in the answer set of the Q_1. According to the definition of the query support we can say that the query containment is monotonic w.r.t. support. More specific query is built from more general one by adding an atom to a query.

3.3 The correctness of the pattern refinement method for $\mathcal{SHIF}$

In KAON2, $\mathcal{DL}$ $\mathcal{KB}$ is translated into Disjunctive Datalog, DD($\mathcal{KB}$). The difference between the general first-order semantics of $\mathcal{DL}$ and minimal model semantics of Disjuctive Datalog is not relevant for answering positive queries in positive Datalog programs, because all positive ground facts entailed by $\mathcal{KB}$ are contained in each minimal model of DD($\mathcal{KB}$) and vice versa. The reduction to DD($\mathcal{KB}$) produces only positive programs [5] and this property ensures that our method for building more specific patterns, consisting of adding literals one by one, is correct.

Lemma 1. *Let Q2 be a query over KB, built from the query Q1 by adding an atom. It holds that* $Q_1 \succeq Q_2$.

Proof (Sketch). If the program in Disjunctive Datalog remains fixed, also the answer sets stay fixed. The program DD($\mathcal{KB}$) is independent of the query, as long as the query is built from positive atomic concepts or roles (see: [12]). Queries in our approach, do not contain any atoms with complex concepts or roles (only with atomic ones), thus the query does not introduce any new symbols to a TBox and the reduction is independent of the query. Hence, DD($\mathcal{KB}$) can be computed once and remains fixed while answering to any number of the queries of the described form. Thus adding positive literals to the query can at most reduce answers.

3.4 Illustrative example

As an illustrative example within this paper we consider the knowledge base describing bank services and clients.

Example 1. The TBox is presented below (*Account*, *Client*, *CreditCard* and *Loan* are disjoint with each other and *Man* and *Woman* are also disjoint):

$Man \sqsubseteq Client$
$Woman \sqsubseteq Client$
$Gold \sqsubseteq CreditCard$
$Interesting \equiv \exists\ hasLoan.Loan$
$\top \sqsubseteq \forall\ isOwnerOf.(Account \sqcup CreditCard)$
$\top \sqsubseteq \forall\ hasLoan.Loan$
$\top \sqsubseteq \forall\ hasLoan^{-}.Account$
$\top \sqsubseteq \leq\ 1\ isOwnerOf^{-}$
$\top \sqsubseteq \forall\ isOwnerOf^{-}.Client$

In the ABox we have the following assertions:

Man(Marek).
Man(Adam).
Woman(Anna).
Woman(Maria).
Account(a1).
Account(a2).
Account(a3).
CreditCard(c1).
CreditCard(c2).
isOwnerOf(Marek, a1).
isOwnerOf(Marek, c1).
isOwnerOf(Anna, a2).
isOwnerOf(Anna, c2).
isOwnerOf(Maria, a3).

Let's assume that our reference concept is *Client*. Then the query Q_{ref} has the form *q(key):-Client(key)* and has 4 items in its answerset. Let's assume further that we would like to calculate the support of the example query Q of the form *q(key):-Client(key), isOwnerOf(key, x), Account(x), isOwnerOf(key, y), CreditCard(y)*. The query Q has two items in its answerset that are the clients having at least one account and at least one credit card. The support of the query Q is then calculated as: $support(\hat{C}, Q, KB) = \frac{2}{4} = 0.5$.

4 Overview of the approach

Our algorithms are based on the property of the query support that for every pair of patterns *p1* and *p2*: $p1 \succeq p2 \Rightarrow support(p1) \geq support(p2)$. It can be thus apriori determined that more specific patterns subsumed by an infrequent pattern are also infrequent.

We use the special *trie* data structure that was successfully used in FARMER. In the trie data structure, nodes correspond to the atoms of the query. Every path from the root to a node corresponds to a query (see Figure 2). New nodes are added to the trie, only if the resulting queries are frequent. Thus only leaves that correspond to frequent queries are expanded. We distinguish two ways in which atoms can be added as leaves to the trie, as described in Definition 3[3].

Definition 3 (Refinement rules). *Atoms are added to the trie as:*

1. ***dependent atoms*** *(use at least one variable of the last atom in the query),*
2. ***right brothers*** *of a given node (these are the copies of atoms that have the same parent node that a given node and are placed on the right side of a given node).*

[3] In [7] there was also the third rule, generating copies of atoms. Here we decided to include this into the first rule, as copies of atoms are just the special case of dependent atoms.

Depenedent atoms are built from predicates from the *admissible predicates* list. This list is computed for each predicate, when it is added in some atom to the trie for the first time. It contains also the information which variables of the child node are to be shared with the parent node (dependently on shared variables combinations, a predicate may be added in several ways as admissible one, also to itself). The list is then stored in a hash structure and retrieved when the atom with the given predicate is expanded for the next time. To add a predicate to the list, the intersections of descriptions (from parent and child predicate) describing future shared variables have to be satisfiable wrt the TBox.

Right brother copying mechanism takes care that all possible subsets, but only one permutation, out of a set of dependent atoms is considered. Variables in binary predicates are renamed, where needed, while being copied.

4.1 The implemented settings

We implemented two versions of our approach: generating all *semantically closed patterns* (that is the patterns to which it is not possible to add any literal without effecting the semantics. These are the largest patterns in a class of semantically equivalent patterns) and, generating all *semantic equivalence classes of patterns* (that is at least one representative of each class of semantically equivalent patterns, but here we are interested in generating possibly the shortest patterns). An example closed pattern in Figure 2 is *q(key):-Client(key), isOwnerOf(key,x0), CreditCard(x0).* An example representative of an equivalence class is *q(key):-Client(key), isOwnerOf(key,x0), Gold(x0)* (to make it closed the literal *CreditCard(x0)* should be added). For each one of these two versions, we apply a number of different syntactic and semantic rules to generate all patterns and eliminate redundant patterns of each kind.

By the procedure *checkSyntactically* in Algorithm 1 we denoted the rules that can be performed by looking only at the syntactic form of a pattern, without using the semantic information from a TBox. The rules rely on introducing new variable names systematically, level by level, what helps to maintain the hash lists of previously added dependent atom forms. As the atoms of the form already generated will be copied as right brothers, they should not be generated again in the same form (different only due to the variable renaming) as dependent atoms in the next level.

On the semantic part (*checkSemantically* in Algorithm 1), the method starts from classifying the concept taxonomy, which, together with properties hierarchy, serves for regular construction of a trie. In case of closed patterns, to admissible predicates list the concepts and properties from the top level of hierarchies are added, and their direct subconcepts/subproperties are added to them on the next level (see: Figure 2a, where only the top concepts *CreditCard* and *Account* were considered as admissible predicates of *isOwnerOf*). In case of equivalence classes, the whole branches from the top to the bottom of a hierarchy are added at the same level (see: Figure 2b). The subsumption hierarchy is also used in several rules, for example, when the domain/range of a given property is equivalent to some class that we are going to add as an admissible predicate to this

property, in case of equivalence classes, we do not add this class as it does not bring any new semantic information (for example: *Loan(x1)* would not be added to *hasLoan(x0,x1)*). The attributes of properties (symmetric, inverse) are treated differently in the two settings, e.g. for closed patterns we generate both, property and it's inverse, while for equivalence classes, the information that one property is an inverse of another one helps to prune semantically redundant literals where possible. For transitive properties, the transitive closure is generated in closed patterns case. In the current approach we do not check whether a new pattern is not semantically equivalent to one of the previously generated ones (which would be expensive), what results in generating redundant patterns.

Before submitting the query to calculate it's support, we test if the constructed query is satisfiable wrt a TBox T, to eliminate unnecessary computation wrt the data base. We decided to test two methods: *complete test of query satisfiability* and it's *approximation*. First consists of checking whether $T \cup \exists x, y : Q$ is satisfiable, that is, whether there is a model of T in which there is some valuation for the distinguished variables x and nondistinguished variables y. The variables are skolemized, and, assuming that $Q(a, b)$ is a new ABox, it is checked whether T is satisfiable in conjunction with that ABox. In second, for each variable in a query, a description is built as an intersection of all descriptions from concepts, domains and ranges of properties describing this variable in a query. The descriptions are kept on the hash list associated with every node and updated for new atom being added to the query (see: Figure 2a). It is checked whether the intersection of the descriptions of the shared variables from a new atom and from a given query, to which we are going to add this atom, are satisfiable.

Below we present the general node expansion algorithm[4]. $P(\boldsymbol{x}, \boldsymbol{y})$ denotes an atom where P is a predicate name and $\boldsymbol{x}$ and $\boldsymbol{y}$ distinguished and undistinguished variables. The trie is generated up to the user-specified MAXDEPTH level.

Algorithm 1 ***expandNode(*** $A(\boldsymbol{x}, \boldsymbol{y_a})$***, nodeLevel)***

1. ***if*** *nodeLevel* $<$ *MAXDEPTH* ***then***
2. ***if*** *admissible predicates of A not computed* ***then***
3. *computeAdmissiblePredicates(A);*
4. ***for all*** $D \in$ *admissible predicates of A* ***do***
5. *build dependent atom* $D(\boldsymbol{x}, \boldsymbol{y_d})$ *of* $A(\boldsymbol{x}, \boldsymbol{y_a})$
6. ***if*** *checkSyntactically()* ***and*** *checkSemantically()* ***then***
7. ***if*** $D(\boldsymbol{x}, \boldsymbol{y_d})$ *is frequent* ***then***
8. *addChild(*$A(\boldsymbol{x}, \boldsymbol{y_a})$*,* $D(\boldsymbol{x}, \boldsymbol{y_d})$*);*
9. ***endfor***
10. ***for all*** $B(\boldsymbol{x}, \boldsymbol{y_b}) \in$ *right brothers of* $A(\boldsymbol{x}, \boldsymbol{y_a})$ ***do***
11. *create* $B'(\boldsymbol{x}, \boldsymbol{y_{b'}})$ *which is a copy of node* $B(\boldsymbol{x}, \boldsymbol{y_b})$*;*
12. ***if*** *checkSyntactically()* ***and*** *checkSemantically()* ***then***
13. ***if*** *a copy* $B'(\boldsymbol{x}, \boldsymbol{y_{b'}})$ *of* $B(\boldsymbol{x}, \boldsymbol{y_b})$ *is frequent* ***then***
14. *addChild(*$A(\boldsymbol{x}, \boldsymbol{y_a})$*,* $B'(\boldsymbol{x}, \boldsymbol{y_{b'}})$*);*
15. ***endfor***
16. ***for all*** $C(\boldsymbol{x}, \boldsymbol{y_c}) \in$ *children of* $A(\boldsymbol{x}, \boldsymbol{y_a})$ ***do***

[4] In case of closed patterns there is an additional step of generating transitive closure for transitive properties, which is not present in this algorithm.

17. *expandNode(C($\boldsymbol{x}, \boldsymbol{y_c}$), nodeLevel+1)*
18. ***endfor***

Example 2. (Following Example 1). Let's assume having the TBox from Example 1 and the ABox bigger than in previous example (for the sake of clarity we will not discuss it within this example). Then our method works as follows. First we classify a taxonomy and as an effect we obtain the classification presented in Figure 1. The top-level concepts in the example are: *CreditCard*, *Client*, *Account*, *Loan*. For the predicate *Client* admissible predicates are: $isOwnerOf$ (only first variable can be the shared one), Man and $Woman$. An example trie is presented in Figure 2. The numbers on edges refer to two ways in which the atoms can be added to the trie. Some of the hash lists of variable descriptions associated with nodes are shown in Figure 2a. The trie is built up to the level

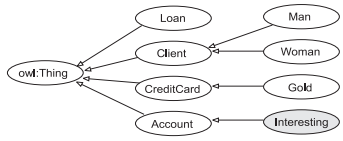


Fig. 1. Classified taxonomy.

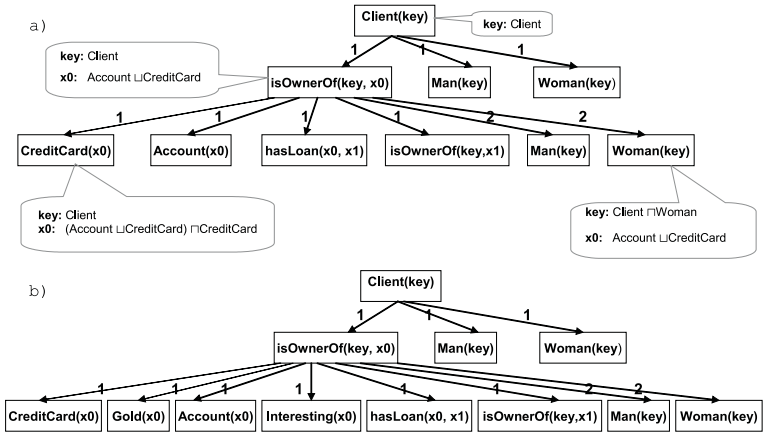


Fig. 2. A part of a trie generated for *Client* as a reference concept. (a)sematically closed patterns. (b) semantic equivalence classes of patterns

3 (to the patterns having the length of 3 atoms). The example pattern at this level is: $q(key) : -Client(key), isOwnerOf(key, x0), CreditCard(x0)$.

5 Experimental results

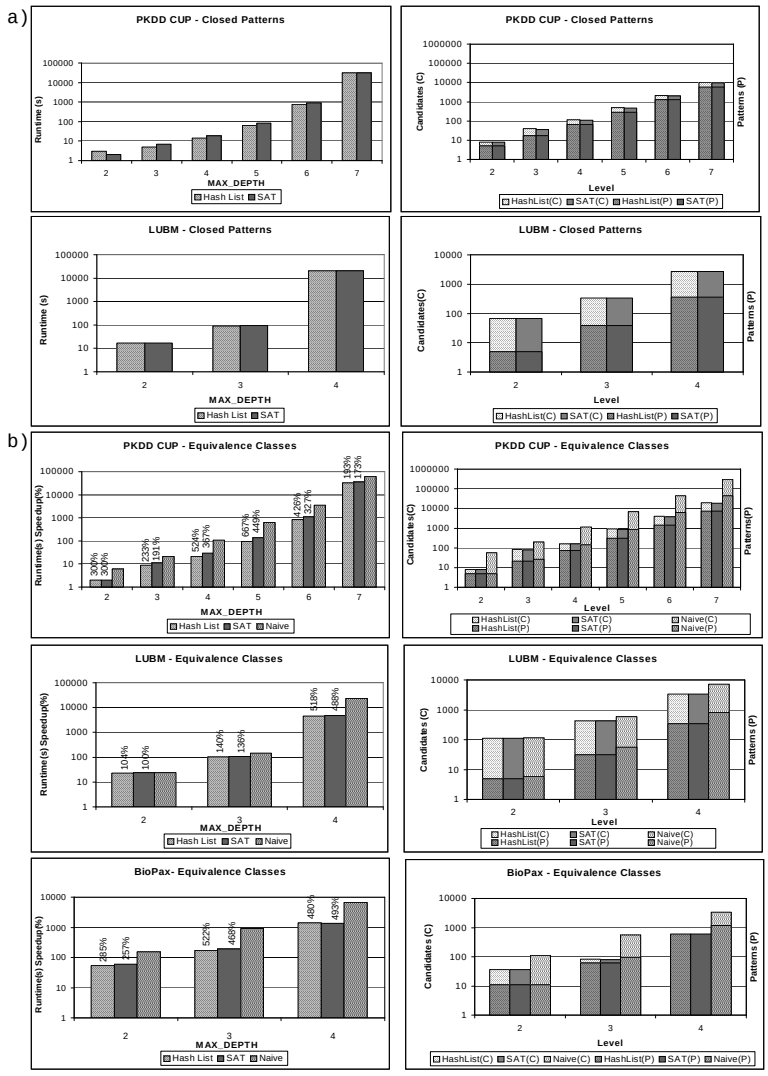
We implemented and tested all of different settings of our approach, discussed in the previous section. The primary goal of the experiments was to estimate the increase of the data mining efficiency by using background knowledge from a TBox in different settings. We wanted also to test how our method performs on datasets of different sizes and complexities (within the OWL Lite fragment), to obtain an idea what kinds of ontologies can be efficiently handled.

To test our SEMINTEC approach, we created the ontology based on the financial dataset from the PKDD'99 Discovery Challenge (*PKDD CUP*)[5]. It is relatively simple, as it does not use existential quantifiers or disjunctions. It requires, however, hard for deductive databases, equality reasoning, as it contains functionality assertions and disjointness constraints. *LUBM* is a benchmark from the Lehigh University[6], consisting of a university domain ontology and a generator of a synthetic data (we set the number of universities to 1). There are existential quantifiers used, but no disjunctions or number restrictions, hence the reduction algorithm produces an equality-free Horn program, on which query answering can be performed deterministically. The *Biopax* ontology contains biological pathway data [7]. For tests we used AgroCyc dataset. It uses existential quantifiers, disjunctions, functionality assertions and disjointness constraints. Since it contains also nominals, which KAON2 cannot handle, we adapted it for our tests. Namely, each enumerated concept $i_1, ..., i_n$ was replaced with a new concept O and it's subconcepts $O_1, ..., O_n$ and assertions $O_k(i_k)$ added. The datatype properties P_d, used in axioms, were replaced by object properties P_o and the statements of the form $\exists P_d.\{i_k\}$ with the form $\exists P_o.O_k$.

In Figure 3 some of our experimental results are presented. The tests were performed on a computer with Intel Core2 Duo 2.4GHz processor, 2GB of RAM, running Microsoft Windows Serwer 2003 Standard Edition SP1. Our implementation and KAON2 (release 2006-12-04) are in Java (version 1.5.0). The JVM heap size was limited to 1.5GB, the maximum time for each test to 25 hours. A trie was generated up to specified MAXDEPTH values, from 1 to the maximum depth where the time of 25 hours was not exceeded, and using the recursive strategy presented in the Algorithm 1. The runtimes are the times of a whole trie generation for each MAXDEPTH (maximum length of patterns). However, the numbers of candidates and frequent patterns found are shown separately for each level (each length of patterns). Besides the runtimes, there is also the speedup presented of the better methods compared to the naive approach. The bars representing the numbers of patterns are superimposed on top of the ones representing the numbers of candidates. For PKDD CUP we set the *minsup* to

[5] PKDD CUP, http://www.cs.put.poznan.pl/alawrynowicz/goldDLP2.owl
[6] LUBM, http://swat.cse.lehigh.edu/projects/lubm/
[7] BioCyc Database Collection, http://biocyc.org

Fig. 3. Experimental results (logarithmic scale). (a) semantically closed patterns. (b) semantic equivalence classes. **HashList**: an approximation of query satisfiability test. **SAT**: complete test. **Naive**: naive approach.



0.2 and the $\hat{C}$ concept to *CreditCard*. For LUBM and BioPax we set the *minsup* to *0.3* and the $\hat{C}$ concepts adequatly to *Person* and to *entity*.

An example pattern discovered, from the PKDD CUP dataset, is: *q(key):- Client(key), isOwnerOf(key,x1), NoProblemAccount(x1), hasStatementIssuance-Frequency(x1,x2), Monthly(x2), hasAgeValue(key,x3), From35To50(x3)* (support: 0,23). It describes the clients of the age between 35 and 50 years, having at least one account with the statements issued monthly and no problem status of paying off the loans (i.e. any loan granted or only no problem loans). The latter information can be read from the PKDD CUP ontology, where *NoProblemAccount* is defined as *Account* having only no problem loans ($NoProblemAccount \sqsubseteq Account \sqcap \forall hasLoan.OKLoan$). Using background knowledge from the TBox, in the equivalence classes case, saved us from generating the patterns where both, *AgeValue* and *From35To50* or *hasAgeValue* and *AgeValue* predicates, are present (the following axioms were used: $From35To50 \sqsubseteq AgeValue$ and $\top \sqsubseteq \forall hasAgeValue.AgeValue$).

In the naive approach, where no semantics is exploited, we applied all possible syntactic rules for redundant patterns elimination (see: *checkSyntactically* from Section 4.1). Nevertheless, the naive approach still generated many more patterns, to be tested, at each level. *HashList* approximation for testing query satisfiability was in most cases faster than the complete test (SAT), even though it had to evaluate more queries. Testing on LUBM showed an important feature of our approach. There are no disjointness constraints in this ontology, and hence we cannot eliminate too many from the admissible predicates lists, what causes a lot of atoms to be tested each time as dependent ones. However, when using semantic information (such as the subsumption hierarchy of concepts), there are speedups even for this ontology. For the other two ontologies, the presence of disjointness constraints helped to eliminate many literals to be tested as query refinements. LUBM has also the transitive roles, what is the cause of the big shift in the runtime in the closed patterns case.

6 Conclusion and future work

In this paper we present an evaluation of the approach to frequent pattern mining in $\mathcal{DL}$ knowledge bases. After introducing the general algorithm, based on the efficient trie data structure, we discuss different settings under this algorithm. We present the experimental results of testing them on knowledge bases of different sizes and complexities. To the best of our knowledge, there have been only one other approach for frequent pattern discovery, system SPADA, that uses $\mathcal{DL}$ for representing background knowledge. We work on how to compare our approach to this one as well as c-armr, although it may be diffucult to perform direct and fair comparison due to the different languages, different forms of patterns and different generality relations. We are going also to make our system publicly available.

In further research we plan to focus on optimization techniques and heuristic algorithms to speed up the pattern discovery process. The investigation of mea-

sures of interestingness and developing the methods for pruning huge pattern space is also worth considering. We plan to investigate using the newly discovered knowledge for the association rule discovery and conceptual clustering tasks and possibly for ontology evolution.

Acknowledgments. Work partially supported by Polish Ministry of Science and Higher Education (under grant number KBN 3T11F 025 28)

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P. (Eds.) (2003) The description logic handbook: Theory, implementation and applications, Cambridge University Press
2. Blockeel H., Dehaspe L., Demoen B., Janssens G., Ramon J., and H. Vandecasteele (2002) Improving the efficiency of Inductive Logic Programming through the use of query packs, Journal of Artificial Intelligence Research 16, 135-166
3. Dehaspe, L., Toivonen, H. (1999) Discovery of frequent Datalog patterns. Data Mining and Knowledge Discovery, 3(1): 7-36
4. Donini, F., Lenzerini, M., Nardi, D., Schaerf, A. (1998) AL-log: Integrating datalog and description logics, Journal of Intelligent Information Systems, 10:3, 227-252
5. Hustadt U., Motik B., Sattler U. (2004) Reducing $\mathcal{SHIQ}$ Description Logic to Disjunctive Datalog Programs. In Proc. of KR 2004, AAAI Press, 152-162
6. Józefowska J., Ławrynowicz A., Łukaszewski T. (2005) Towards discovery of frequent patterns in description logics with rules, In Proc. of RuleML-2005, LNCS 3791, Springer, 84-97
7. Józefowska J., Ławrynowicz A., Łukaszewski T. (2006) Frequent pattern discovery in OWL DLP knowledge bases, In Proc. of EKAW 2006, LNAI 4248, Springer, 287-302
8. Lisi F.A., Malerba D. (2004) Inducing Multi-Level Association Rules from Multiple Relation, Machine Learning Journal, 55, 175-210
9. Lisi F.A., Esposito F. (2004) An ILP Approach to Semantic Web Mining, In Proc. of Knowledge Discovery and Ontologies (KDO-2004), Workshop at ECML/PKDD 2004
10. Mannila, H., Toivonen, H. (1997) Levelwise search and borders of theories in knowledge discovery. Data Mining and Knowledge Discovery 1(3): 241-258
11. Motik B., Rosati R. (2007) A Faithful Integration of Description Logics with Logic Programming. In Proc. of IJCAI 2007, Morgan Kaufmann, 477-482
12. Motik B., Sattler U., Studer R. (2004) Query Answering for OWL-DL with Rules. In Proc. of ISWC 2004, LNCS 3298, Springer, 549-563
13. Motik B., Sattler U. (2006) A Comparison of Reasoning Techniques for Querying Large Description Logic ABoxes. In Proc. of LPAR 2006, LNCS 4246, Springer, 227-241
14. Nijssen, S., Kok, J.N. (2001) Faster Association Rules for Multiple Relations. In Proc. of IJCAI 2001, Morgan Kaufmann, 891-897
15. Nijssen, S., Kok, J.N. (2003) Efficient frequent query discovery in FARMER. In Proc. of PKDD 2003, LNAI 2431, Springer, 350-362
16. De Raedt L., Ramon J. (2004) Condensed representations for Inductive Logic Programming, Principles of Knowledge Representation and Reasoning: Proceedings of the KR2004, 438-446

Quantitative association rule mining in genomics using apriori knowledge

Filip Karel, Jiří Kléma

Department of cybernetics, Czech Technical University in Prague,
Technická 2, Praha 6, 166 27
karelf1@fel.cvut.cz, klema@labe.felk.cvut.cz

Abstract Regarding association rules, transcriptomic data represent a difficult mining context. First, the data are high-dimensional which asks for an algorithm scalable in the number of variables. Second, expression values are typically quantitative variables. This variable type further increases computational demands and may result in the output with a prohibitive number of redundant rules. Third, the data are often noisy which may also cause a large number of rules of little significance. In this paper we tackle the above-mentioned bottlenecks with an alternative approach to the quantitative association rule mining. The approach is based on simple arithmetic operations with variables and it outputs rules that do not syntactically differentiate from classical association rules. We also demonstrate the way in which apriori genomic knowledge can be used to prune the search space and reduce the amount of derived rules.

Keywords: association rules, quantitative attributes, apriori knowledge, SAGE

1 Introduction

At present, large quantities of gene expression data are generated. Data mining and automated knowledge extraction in this data belong to the major contemporary scientific challenges. For this task clustering is one of the most often used method [2] – the most similar genes are found so that the similarity among genes in one group (cluster) is maximized and similarity among particular groups (clusters) is minimized. Although very good results are gained by this method there are three main drawbacks [3]:

1. One gene has to be clustered in one and only one group, although it functions in numerous physiological pathways.
2. No relationship can be inferred between the different members of a group. That is, a gene and its target genes will be co-clustered, but the type of relationship cannot be rendered explicit by the algorithm.
3. Most clustering algorithms will make comparisons between the gene expression patterns in all the conditions examined. They will therefore miss a gene grouping that only arises in a subset of cells or conditions.

Association rule (AR) mining [1] can overcome these drawbacks. However, when dealing with datasets containing quantitative attributes it is often advisable to adapt the original AR mining algorithm. Mining of quantitative association rules (QARs) is considered as an interesting and important research problem. It was described in several papers such as [5], [6], [18], [19] which proposed various algorithmic solutions. Nevertheless, the proposed algorithms often do not take time consumption into the account.

QAR mining techniques aimed at gene-expression data were proposed for example in [4] or [15]. Half-spaces are used to generate QAR in [4], rules of the form 'if the weighted sum of some variables is greater than a threshold, then, with a high probability, a different weighted sum of variables is greater than second threshold'. An example of such rule can be '0.99 $gene_1$ - 0.11 $gene_2$ > 0.062 → 1.00 $gene_3$ > -0.032'. This approach naturally overcomes the discretization problem, on the other hand it is quite hard to understand the meaning of the rule.

In [15], the authors bring external biological knowledge to the AR mining. They mine rules which directly involve biological knowledge into the antecedent side of the rule. The given method can be applied to mine annotated gene expression datasets in order to extract associations like '$cell\_cycle \rightarrow [+]condition_1$, $[+]condition_2$, $[+]condition_3$, $[-]condition_6$', which means that, in the dataset, a significant number of the genes annotated as 'cell cycle' are over-expressed in condition 1, 2 and 3 and under-expressed in condition 6. This approach works with binary values of gene-expression only.

In this paper, QAR mining algorithm [12] is used and further developed. Despite it is very different from the classical AR algorithms, it outputs association rules in the classical form '$gene_i = <l\_value_{gi}..h\_value_{gi}> \wedge gene_j = <l\_value_{gj}..h\_value_{gj}> \wedge ... \rightarrow cancer = 0/1$. We can read this rule as 'when the value of $gene_i$ is between l_value_{gi} and h_value_{gi} and the value of $gene_j$ is between l_value_{gj} and h_value_{gj} and ... then with a high probability the cancer will (not) occur'. The task can be rephrased as search for the genes and their values that coincide with the appearance of cancer.

The algorithm is by no means limited to the particular right hand side (RHS) of rules. The target variable *cancer* is used here as it represents the most interesting outcome. The invariable RHS also simplifies the evaluation in Section 4. As follows from the structure of the rules, the presented algorithm deals with discretized quantitative attributes. A priori discretization influences resulting rules. One of the main interests of this paper is to compare the discretization into more bins (which prevents information loss) with binarization.

Background knowledge (BK) – the external apriori biological information – can be extracted using various publicly accessible web databases and tools [7], [8], [10]. Possibility of using this source of information to improve the generation of ARs is another aim of this paper. We show that appropriate implementation of BK can improve the quality of generated rules. The simplest utilization of BK is to give the rules their biological sense by straightforward annotation of the set of rules without their pruning. BK also helps to focus on specific rule subsets by early utilization of regular expressions. The most interesting use of BK is to

get the most plausible rules by application of gene similarity. Moreover, BK can significantly reduce the search space.

The paper is organized as follows: Section 2 presents the SAGE data, studies possible ways of its preprocessing and introduces apriori knowledge relevant to the given dataset. Section 3 gives an outline of QAR algorithm and discusses the ways it can employ apriori knowledge. Section 4 summarizes the reached results with the main stress on the effects of discretization and utilization of apriori knowledge. Finally we conclude in Section 5.

2 Character of SAGE data and preprocessing of raw data

The SAGE (Serial Analysis of Gene Expression) technique aims to measure the expression levels of genes in a cell population [20]. In this paper, the raw data matrix described in [11] was used. The expression dataset consists of 11082 tags (i.e., genes or attributes) whose expression was measured in 207 SAGE libraries (i.e. 207 biological situations or experiments). The tags represent the subset of human genome which is currently unambiguously identifiable by Identitag [3], the biological situations embody various tissues (brain, prostate, breast, kidney or heart) stricken by various possible diseases (mainly cancer, but also HIV and healthy tissues).

	$gene_1$	$gene_2$	...	$gene_n$	*cancer*
$situation_1$	0	15	...	0	0
$situation_2$	8	4	...	0	1
⋮	⋮	⋮	⋮	⋮	⋮
$situation_m$	3	0	...	39	1

Table 1. The structure of the raw SAGE data (n=11082, m=207), the gene values correspond to the expression of the particular gene in the particular biological situation, *cancer* stands for a binary class.

The structure of the raw SAGE expression dataset is in Table 1. As the main observed disorder is carcinoma, a target binary attribute *cancer* was introduced by the domain expert. The class value is 0 for all the healthy tissues and also the tissues suffering by other diseases than cancer (77 situations, 37.2%). It is equal to 1 for all the cancerous tissues (130 situations, 62.8%).

SAGE datasets are sparse – a great portion of gene-expression values equal to zero. The distribution of zeroes among genes is very uneven. Housekeeping genes are expressed (nearly) in all the tissues, however there is a reasonable amount of genes having zero values in almost all situations. Such genes are not suitable for further rule mining. Table 2 shows the numbers of frequently expressed genes. We can see that out of the total number of 11082 genes, only 97 have at least 95% non-zero values.

X	number of genes
5%	97
20%	305
50%	1038
80%	2703

Table 2. The number of genes having at the most X% of zero values

2.1 Discretization of expression values

In order to minimize the role of noise in SAGE data, the data are usually discretized first. As the discretization also brings the information loss, it is always disputable which type of discretization to apply. For a thorough discussion upon the impact of discretization see [16].

Binarization is now the most widely used method of discretization of gene expression data, where 0 means that the gene is under expressed and 1 means that the gene is over expressed. There are two disadvantages of data binarization: (1) it results in the biggest information loss, (2) it significantly influences (or rather forms) the output rules.

Table 3 describes the distinction among different types of binarization. 'Max -Y%' binarization means that the Y% of the highest value is the 0/1 threshold (provided the highest value of $gene_i$ is 100 and Y=90%, the threshold is 10, all the values above are encoded as 1). In 'median' binarization the border is the value of median. Logically, the most uniform distribution is obtained through the 'median' binarization. The most similar to 'median' is 'Max -80%' binarization using the gene sets with lower numbers of zeros values and 'Max -90%' using the gene sets with higher numbers of zero values.

	Max -90%	Max -80%	Max -70%	Median
X gene-set	0/1 ratio	0/1 ratio	0/1 ratio	0/1 ratio
5%	0.28 / 0.72	0.56 / 0.44	0.74 / 0.26	0.49 / 0.51
20%	0.32 / 0.68	0.59 / 0.41	0.77 / 0.23	0.49 / 0.51
50%	0.45 / 0.55	0.66 / 0.34	0.81 / 0.19	0.49 / 0.51
80%	0.60 / 0.40	0.74 / 0.26	0.84 / 0.16	0.61 / 0.39

Table 3. The results of binarization in terms of the 0/1 ratio. X defines the gene sets shown in Table 2.

Discretization into more bins enables more accurate rules. However, the classical equi-width and equi-depth approaches fail in this case. The former introduces intervals that are nearly empty, the latter keeps the same frequency across the intervals with unnatural bounds. The discretization based on 1-D clustering has to be employed. In short, the discretization steps repeated for each attribute are:

1. Initialize equi-distantly the *centers* of bins.
2. Assign every record value to the nearest center.
3. Recalculate every center position (average value of all records assigned to the center).
4. If the position of all centers did not move then end, else go to 2/.

The results of discretization into four and six bins are in Table 4. 4-bin discretization has approximately the same number of values assigned to the lowest bin as 'Max -80%'. Better resolution is obtained in higher values only. Using 6-bin discretization the resolution is better even in low values. But still low numbers of values are assigned to the higher bins. This is caused by the original distributions of gene expression values, where the majority of values is very close to zero.

	4-bin discretization	**6-bin discretization**
X gene-set	**1/2/3/4 ratio**	**1/2/3/4/5/6 ratio**
5%	0.63 / 0.24 / 0.08 / 0.05	0.45 / 0.27 / 0.13 / 0.06 / 0.06 / 0.03
20%	0.65 / 0.25 / 0.07 / 0.03	0.48 / 0.29 / 0.12 / 0.05 / 0.04 / 0.02
50%	0.69 / 0.23 / 0.06 / 0.02	0.52 / 0.27 / 0.10 / 0.04 / 0.05 / 0.01
80%	0.74 / 0.19 / 0.05 / 0.02	0.59 / 0.20 / 0.08 / 0.04 / 0.08 / 0.01

Table 4. The ratio of the number of values using the clustering discretization.

2.2 Background knowledge

Genomic websites such as NCBI [10] or EBI [9] offer a great amount of heterogeneous background knowledge available for various biological entities. In this paper we focused on Gene Ontology (GO) terms. To access the gene annotation data for every tag considered, RefSeq identifiers were translated into EntrezGene identifiers [8], the mapping approached 1 to 1 relationship. Knowing the gene identifiers, the annotations were automatically accessed through hypertext queries to the EntrezGene database [10] and sequentially parsed by Python scripts.

GO terms A list of related GO terms can be found for each gene (however for a certain portion of genes there are no GO terms available and the list is empty). This list characterizes the given gene and can be used to assume on its molecular function (MF) or the biological processes and the cellular components it participates in. The lists can be searched by regular expressions in order to focus on specific subsets of genes.

Similarity matrices GO terms can straightforwardly be used to compute similarity among genes. The rationale sustaining this method is that the more GO terms the genes share, and the more specific the terms are, the more likely the genes are to be functionally related. Two matrices – for BPs and MFs – created by authors in [11] are used. The structure of the gene similarity matrices is in

Table 5. The similarity values lie in the interval $< 0; 1 >$, where 1 stands for the genes with the identical description for the given category of terms. There are around 85% of missing similarity values (denoted n/a) for the genes with empty lists of related GO terms.

	$gene_1$	$gene_2$	$gene_3$	$gene_4$	...	$gene_n$
$gene_1$		0.15	0.75	n/a	...	n/a
$gene_2$			n/a	0.12	...	0.93
$gene_3$				0.64	...	n/a
$gene_4$					...	n/a
⋮						⋮
$gene_n$						

Table 5. The structure of the gene similarity matrix.

In order to simplify the notion of similarity, both the above-described matrices are combined into one matrix as follows:

$$sim_{ij} = sim(BP)^2_{ij} + sim(MF)^2_{ij}$$

where $sim(BP)_{ij}$ is the similarity value for the genes i and j with respect to their biological process GO terms, $sim(MF)_{ij}$ is the similarity value for the same genes with respect to their molecular function GO terms.

3 QAR algorithm

An innovative QAR algorithm [12] is used for AR generation in this paper. The detailed algorithm description is out of the scope of this paper. The essential principles of the algorithm can be summarized as follows:

1. The input of the algorithm is a set of *atomic attributes*: $a_1, a_2, ...a_n$.
2. All the atomic attributes are discretized into D discretization bins and mapped to the consecutive row of integers beginning with one and ending with D (one represents the lowest value and D the highest value of an atomic attribute).
3. These preprocessed atomic attributes $pa_1, pa_2, ...pa_n$ are used to construct compound attributes – $x_i(pa_1, pa_2, ...pa_n) : N^n \rightarrow N$. *Compound attribute* is $x_i(pa_1, pa_2, ...pa_n) = \sum_{k=1}^{n} c_k a_k$, where $c_k = \{$-1, 0, 1$\}$, where i is number of compound attribute.
4. Each atomic (compound) attribute has a discrete distribution $P_i(t)$, two atomic (compound) attributes have a joint distribution $P_{ij}(t, s)$.
5. O is a set of all compact square or rectangle areas $o \subset <-\infty, \infty>$ x $<-\infty, \infty>$. For each pair $(x_i, x_j) \in P$ the algorithm searches for the best *areas of interest* o, where for each $(\alpha, \beta) \in o$

$$P_i(\alpha)P_j(\beta) - P_{ij}(\alpha, \beta) \geq \epsilon$$

6. From the areas of interest the best rules are extracted.

This algorithm takes an inspiration from earlier proposed algorithms [6], [14] or [19], but it comes with lower time consumption and pruning of redundant rules. On the other hand, the algorithm does not exhaustively enumerate all the relevant rules as it is not based on complete search through the state space. The algorithm works for binary attributes as well, although it loses its main advantages.

3.1 Injection of background knowledge into QAR algorithm

In order to increase noise robustness, focus and speed up the search, it is vital to have a mechanism to exploit background knowledge during AR generation. In the presented algorithm, BK can be taken into the account during the phase that combines atomic attributes into compound attributes.

The first option takes advantage of the lists of terms that describe the individual atomic attributes (genes in the SAGE data). The terms enable to focus on the rules that contain genes with specific characteristics. Provided x denotes a compound attribute, the variable $regexp$(x,'$*ribosom*$') delivers the number of genes that belong to x and whose at least one term matches the regular expression '$*ribosom*$'. The variable can be employed to get a limited set of rules that concern mainly (or only) ribosomal genes.

The second option exploits the gene similarity matrices [11]. This option focuses on plausible ARs, i.e., the rules that contain at least a certain portion of genes having common properties. The properties themselves do not have to be given by the user. An association rule can originate solely from the compound attributes with the value of gene similarity higher than a user defined threshold. Provided x denotes a compound attribute, the variable $svsim(x)$ gives the number of gene pairs belonging to x whose mutual similarity is known (distinct from n/a) and $mvsim(x)$ stands for its counterpart. $Sumsim(x)$ denotes the similarity sum over the set of genes belonging to x, $insim(x, min, max)$ stands for the number of gene pairs whose similarity lies between min and max.

Consequently, the variable $\frac{sumsim(x)}{svsim(x)}$ makes the average similarity of the compound attribute x, while the variable $\frac{insim(x,thres,1)}{svsim(x)}$ gives a proportion of the strong interactions (similarity higher than the threshold) within the compound attribute. The variable $\frac{svsim(x)}{svsim(x)+mvsim(x)}$ can avoid the compound attributes with prevailing genes of an unknown function. Relational and logical operators enable to create the final constraint, e.g., $V_1 \geq thres1$ and $V_2 \neq thres2$ where V_i stands for an arbitrary variable characterizing the compound attribute. Although we consider GO terms only, the framework is obviously general and the constraints can also be simultaneously derived from different external datasets.

The described technique obviously causes early pruning of the search space. Some of the compound attributes are rejected and the algorithm does not further search for the rules which do not satisfy the condition given by BK.

4 Experiments and results

This section presents the achieved experimental results. The influence of selected discretization methods is discussed. ARs in the classical form are generated. Conditions on the gene expression values are conjuncted on their LHS, the number of conditions is limited to three. The rules always have the attribute 'cancer' on their RHS. Confidence, support [1] and lift [17] measures are used to evaluate the quality of rules.

The file with maximum of 5% zero values was used. The input table for AR mining consists of 98 genes (attributes) and 207 situations (transactions). The number of attributes is low as the general scalability of the presented algorithm is not concerned here. It has already been proven in earlier works [12,13], along with its ability to reduce redundancy of the resulting set of rules. The main concern is to demonstrate applicability of BK to further improve understandability and scalability of QAR mining.

4.1 Rules without background knowledge

Table 7 shows the influence of discretization methods on the number of generated rules. This number is several times higher using a multi-bin discretization compared with binarization. There are also distinctions among particular binarization types, although not so significant. More rules are generated using binarizations with a more uniform distribution of zero and one values.

Similarity of rules generated by different discretization techniques was also examined, although it is hard to exactly compare different sets of rules. We considered two rules equal when all the antecedent genes, which occurred in the first rule also occurred in the second rule. For example, if genes with ID numbers 9, 13 and 82 occurr in the $rule_1$ and the same genes also occurr in the $rule_2$, then $rule_1 = rule_2$, no matter what values the genes take in the rules. The results are captured in Table 6, where the value on i-th column and j-th row is gained as

$$r_{ij} = \frac{number\_of\_rules_{i,j}}{number\_of\_rules_j},$$

where $number\_of\_rules_{i,j}$ is the number of rules generated both by the i-th type of discretization and by the j-th type of discretization and $number\_of\_rules_j$ is the total number of rules generated by the j-th type of discretization.

We can see that the ratios are quite low. It means that one can achieve a certain percentage of rules that agree in both types of discretization but quite a high number of rules is different. For example, when using 'Max -70%' and 'Max -80%' we gain approximately the same absolute number of rules from which only one fifth is equal. Also, '6-bin' discretization identifies only from 60% to 70% of rules identified using other types of discretization.

Experimentally it was found that these numbers depend on min_supp threshold. Lowering min_supp the ratios of 'identical' rules increase and higher numbers of similar rules are generated.

	Max -90%	Max -80%	Max -70%	Median	4-bin	6-bin
Max -90%	1	0.37	0.07	0.57	0.30	0.56
Max -80%	0.25	1	0.21	0.41	0.58	0.51
Max -70%	0.05	0.18	1	0.39	0.45	0.74
Median	0.26	0.29	0.23	1	0.48	0.61
4-bin	0.12	0.37	0.25	0.44	1	0.58
6-bin	0.15	0.20	0.25	0.35	0.36	1

Table 6. The number of the equal rules having 3 antecedent attributes generated by different discretization methods.

4.2 Using background knowledge (BK) for rules generation

Syntactically the same rules were generated with using BK, but a pruning condition was added. Using notation from Section 3.1, the applied conditions can be written as: 'generate rules with a compound attribute x only if $insim(x, 0.65, 2) \geq 1$'. It means that x is acceptable only if there is a pair of genes of x whose similarity is higher than the $min\_sim = 0.65$ threshold (at the same time it positively holds $svsim(x) \geq 2$). This condition early prunes the space of compound attributes and it is not only a rule filtering condition as for example min_conf condition.

	Max -90%	**Max -80%**	**Max -70%**	**Median**	**4-bin**	**6-bin**
3-ant (min_conf=0.9)	1 102	1 672	1 453	2 392	2 617	4 210
3-ant (min_conf=1.0)	88	33	15	90	126	65
3-ant (min_conf=0.8)	1 681	3 227	1 977	5 453	4 432	6 966
3-ant (min_conf=0.9)	150	152	117	317	247	360

Table 7. The number of rules created by different types of discretization without using background knowledge (top) and with background knowedge (bottom). Min_supp = 0.1, min_lift = 1.3, min_similarity = 0.65

	Binarization	**4-bin**	**6-bin**
without background knowledge	1.5×10^6	6.5×10^6	1.2×10^7
with background knowledge	1.7×10^5	7.1×10^5	1.3×10^6

Table 8. Number of verifications.

The number of rules (bottom part of table 7) is approximately 10 times lower than without using BK, the same holds for the number of verifications that the algorithm carries out. For $min\_conf = 0.8$ we obtain approximately the same number of rules as for $min\_conf = 0.9$ without BK. Time consumption remains

about ten times lower as the time-consumption of used algorithm does not depend on min_conf.

Further, the similarity of rules generated with and without BK is explored. In Table 9 we can observe the top 5 genes (top) and the top 5 pairs of genes (bottom) according to the number of their occurrences in rules.

without BK				with BK			
Max -80%	Median	4-bin	6-bin	Max -80%	Median	4-bin	6-bin
4	9	2	13	41	58	13	13
75	6	13	97	18	36	97	41
70	58	6	2	43	9	41	97
43	97	3	6	16	43	16	9
72	52	97	3	52	13	58	16
4-44	21-58	25-78	13-97	3-88	16-58	13-75	6-17
4-75	9-55	2-18	2-97	53-75	13-58	13-55	11-97
55-72	9-42	89-97	2-90	42-43	22-51	6-17	11-13
4-71	9-36	2-97	13-46	41-76	43-75	13-40	13-75
4-70	9-52	3-75	13-86	41-63	43-52	11-13	13-95

Table 9. Top 5 genes (top) and top 5 pairs (bottom) according to the number of occurrences in rules.

For '4-bin' and '6-bin' discretizations the top 5 gene lists are almost the same. Without BK, all of the 4-bin discretization top genes are also the top genes for 6-bin discretization. With BK this holds for 4 out of 5 genes. By contrast, for binarizations (both with and without BK) there is no overlap in the top gene lists. If we compare the gene lists of the identical discretizations with and without using BK, we observe that the multi-bin discretization and the 'median' binarization get the identical gene sets with and without BK.

For the top 5 pairs we have very similar observations as for the lists of top 5 genes. Generally, in the categories with and without BK the 4-bin and 6-bin discretizations are giving very similar results. 'Max -80%' and 'median' binarizations differentiate quite a lot. Between the two categories the most similar results are gained for 4-bin and 6-bin discretizations.

A more detailed comparison of particular gene occurrences in generated rules with and without BK is in Figure 1. Some of the genes have almost the same number of occurrences ($gene_{13}$), whereas other genes which have a very high number of occurrences using BK do not appear frequently in runs without application of BK ($gene_{41}$).

In general, the genes with prevalence of 'n/a' values in the similarity matrices are discriminated from the rules when using BK. However, a gene without annotation can still appear in a neighborhood of 'a strong functional cluster' of other genes. This occurrence then signifies its possible functional relationship with the given group of genes and it can initiate its early annotation. On the other hand,

the genes with extensive relationships to the other genes may increase their occurrence in the rules inferred with BK.

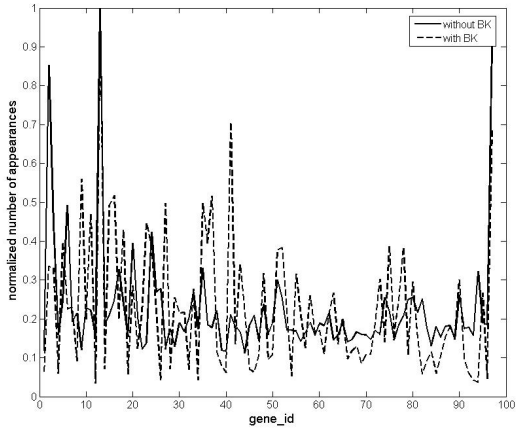


Figure 1. The frequency of particular genes in the generated rules with and without background knowledge for '6-bin' discretization.

5 Conclusions

In this paper, an alternative approach to QAR mining was verified on gene expression data. The paper discussed the influence of discretization methods on the generated rules. It was shown that the output set of rules is significantly influenced by the used discretization both wrt the number of generated rules and their composition. The presented QAR algorithm allowed us to use advantages of discretization into more bins and at the same time to generate rules without combinatoric explosion and without generation of redundant rules. In the light of our findings we think that more attention should be paid to the automatic discretization of gene expression values.

The paper also described and implemented the general framework for exploitation of BK during AR mining. It mainly helps to automatically focus on the most plausible candidate rules. At the same time, pruning conditions based on BK reduce time consumption significantly, while the number of plausible rules remains approximately the same. The conditions used in presented experiments were quite simple. Exploration of other possibilities of this framework and using more complex BK conditions is one of our major future challenges.

Acknowledgement. Filip Karel has been supported by the Ministry of Education, Youth and Sports of the Czech Republic as a part of the specific research

at the CTU in Prague - project nr. CTU0712613. Jiri Klema has been supported by the grant 1ET101210513 "Relational Machine Learning for Analysis of Biomedical Data" funded by the Czech Academy of Sciences.

References

1. R. Agrawal, T. Imelinsky, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Conference on Management of Data*, pages 207–216, Washington, D.C., 1993.
2. Eisen M. B., Spellman P. T., Brown P. O., and Botstein D. Cluster analysis and display of genome-wide expression patterns. *Proceedings of National Academy of Science of the USA 95*, pages 14863–14868,, 1998.
3. Becquet C., Blachon S., Jeudy B., Boulicaut J-F, and Gandril O. Strong-association-rule mining for large-scale gene-expression data analysis: a case study on human SAGE data. *Genome Biology*, 3:531–537, 2002.
4. Georgii E., Richter L., Ruckert U., and Kramer S. Analyzing Microarray Data Using Quantitative Association Rules. *Bioinformatics*, pages ii123–ii129, 2005.
5. T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Mining optimized association rules for numeric attributes. In *In Proc. of ACM SIGMOD Conference on Management of Data*, Montreal, Canada, 1996.
6. S. Guillaume. Discovery of ordinal association rules. In *In Proceedings of the Sixth Pacific-Asia Conference PAKDD'02*, Taiwan, 2002.
7. http://crfb.univ mrs.fr/gotoolbox/. GOToolBox website.
8. http://discover.nci.nih.gov/matchminer/. Matchminer website.
9. http://www.ebi.ac.uk/. EBI website.
10. http://www.ncbi.nlm.nih.gov/. NCBI website.
11. Kléma J., Soulet A., Crémilleux B., Blachon S., and Gandrillon O. Mining plausible patterns from genomic data. In *19th IEEE Symposium on Computer-Based Medical Systems (CBMS'06)*, pages 183–190, 2006.
12. F. Karel. Quantitative and ordinal association rules mining (qar mining). In *Knowledge-Based Intelligent Information and Engineering Systems*, volume 4251 of *LNAI*, pages 195–202. Springer, 2006.
13. F. Karel and J. Kléma. Ordinální asociační pravidla. In *Konference Znalosti 2005*, pages 226–233. VŠB-TUO, 2005.
14. R.J. Miller and Y. Yang. Association rules over interval data. In *In Proc. of ACM SIGMOD Conference on Management of Data*, Tuscon, AZ, 1997.
15. Carmona-Saez P., Chagoyen M., Rodriguez A., Trelles O., Carazo J.M., and Pascual-Montano A. Integrated analysis of gene expression by association rules discovery). *BMC Bioinformatics*, page 7:54, 2006.
16. Ruggero G. Pensa, Claire Leschi, Jérémy Besson, and Jean-François Boulicaut. Assessment of discretization techniques for relevant pattern discovery from gene expression data. In *BIOKDD*, pages 24–30, 2004.
17. G. Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In *in Knowledge Discovery in Databases*, Cambridge, 1991.
18. R. Rastogi and K. Shim. Mining optimized association rules with categorical and numeric attributes. *IEEE Trans. on KD Engineering*, 14(1), 2002.
19. R. Srikant and R. Agrawal. Mining quantitative association rules in large relational databases". In *In Proc. of ACM SIGMOD Montreal*, 1996.
20. Velculescu V., Zhang L., Vogelstein B., and Kinzler K. SAGE (Serial Analysis of Gene Expression). *Science*, page 270:484.7, 1995.

The *Ex* Project: Web Information Extraction using Extraction Ontologies

Martin Labský, Vojtěch Svátek, Marek Nekvasil and Dušan Rak

Department of Information and Knowledge Engineering,
University of Economics, Prague, W. Churchill Sq. 4, 130 67 Praha 3, Czech Republic
e-mail: {labsky,svatek,nekvasim,rakdusan}@vse.cz

Abstract. Extraction ontologies represent a novel paradigm in web information extraction (as one of 'deductive' species of web mining) allowing to swiftly proceed from initial domain modelling to running a functional prototype, without the necessity of collecting and labelling large amounts of training examples. Bottlenecks in this approach are however the tedium of developing an extraction ontology adequately covering the semantic scope of web data to be processed and the difficulty of combining the ontology-based approach with inductive or wrapper-based approaches. We report on an ongoing project aiming at developing a web information extraction tool based on richly-structured extraction ontologies and with additional possibility of (1) semi-automatically constructing these from third-party domain ontologies, (2) absorbing the results of inductive learning for subtasks where pre-labelled data abound, and (3) actively exploiting formatting regularities in the wrapper style.

1 Introduction

Web information extraction (WIE) represents a specific category of web mining. It consists in the identification of typically small pieces of relevant text within web pages and their aggregation into larger structures such as data records or instances of ontology classes. As its core task is application of pre-existent patterns or models (in contrast to inductively discovering new patterns), it falls under the notion of 'deductive' web mining [10], similarly as e.g. web document classification. As such, some kind of prior knowledge is indispensable in WIE. However, the 'deductive' aspects of WIE are often complemented with inductive ones, especially in terms of learning the patterns/models (at least partly) from training data.

In the last decade, WIE was actually dominated by two paradigms. One—*wrapper*-based—consists in systematically exploiting the surface structure of HTML code, assuming the presence of regular structures that can be used as anchors for the extraction. This approach is now widely adopted in industry, however, its dependence on formatting regularity limits its use for diverse categories of web pages. The other—*inductive*—paradigm assumes the presence of training data: either web pages containing pre-annotated tokens or stand-alone examples of data instances. It is linked to exploration of various computational learning paradigms, e.g. Hidden-Markov Models, Maximum Entropy Models, Conditional Random Fields [7] or symbolic approaches

such as rule learning [1]. Again, however, the presence of (sufficient amounts of) annotated training data is a pre-condition that is rarely fulfilled in real-world settings, and manual labelling of training data is often unfeasible; statistical bootstrapping alleviates this problem to some degree but at the same time burdens the whole process with 'heavy computational machinery', whose requirements and side-effects are not transparent to a casual user of a WIE tool. In addition, both approaches usually deliver extracted information as rather weakly semantically structured; if WIE is to be used to fuel semantic web repositories, secondary mapping to ontologies is typically needed, which makes the process complicated and possibly error-prone.

There were recently proposals for pushing ontologies towards the actual extraction process as immediate prior knowledge. *Extraction ontologies* [3] define the concepts, the instances of which are to be extracted, in the sense of various attributes, their allowed values as well as higher level (e.g. cardinality or mutual dependency) constraints. Extraction ontologies are assumed to be hand-crafted based on observation of a sample of resources; however, due to their clean and rich conceptual structure (allowing partial intra-domain reuse and providing immediate semantics to extracted data), they are superior to ad-hoc hand-crafted patterns used in early times of WIE. At the same time, they allow for rapid start of the actual extraction process, as even a very simple extraction ontology (designed by a competent person) is likely to cover a sensible part of target data and generate meaningful feedback for its own redesign; several iterations are of course needed to obtain results in sufficient quality. It seems that for web domains that consist of a high number of relatively tiny and evolving resources (such as web product catalogs), information extraction ontologies are the first choice. However, to make maximal use of available data and knowledge and avoid overfitting to a few data resources examined by the designer, the whole process must not neglect available labelled data, formatting regularities and even pre-existing domain ontologies.

In this paper we report on an ongoing effort in building a WIE tool named *Ex*, which would synergistically exploit all the mentioned resources, with central role of extraction ontologies. Section 2 explains the structure of extraction ontologies used in *Ex*. Section 3 describes the steps of the information extraction process. Section 4 briefly reports on experiments in two different domains. Finally, section 5 surveys related research, and section 6 outlines future work.

2 Ex(traction) ontology content

Extraction ontologies in *Ex* are designed so as to extract occurrences of *attributes* (such as 'age' or 'surname'), i.e. standalone named entities or values, and occurrences of whole *instances* of *classes* (such as 'person'), as groups of attributes that 'belong together', from HTML pages (or texts in general) in a domain of interest.

2.1 Attribute-related information

Mandatory information to be specified for each attribute is: name, data type (string, long text, integer, float) and dimensionality (e.g. 2 for screen resolution like 800x600). In order to automatically extract an attribute, additional knowledge is typically needed.

Extraction knowledge about the attribute *content* includes (1) textual value patterns; (2) for integer and float types: min/max values, a numeric value distribution and possibly units of measure; (3) value length in tokens: min/max length constraints or a length distribution; (4) axioms expressing more complex constraints on the value and (5) coreference resolution knowledge. Attribute *context* knowledge includes (1) textual context patterns and (2) formatting constraints.

Textual patterns in *Ex* (for both the value and the context of an attribute) are regular patterns primarily defined at the level of words (tokens). They may be inlined in the extraction ontology or as (possibly large) external files, and may include the following:

- specific tokens, e.g. 'employed by'
- token *wildcards*, which require one or more token properties to have certain values (e.g. any capital or uppercase token, any token whose lemma is 'employ')
- *character-level* regular expressions for individual tokens
- *references* to other matched *attribute candidates*: a *value* pattern containing a reference to another attribute means that it can be nested inside this attribute's value; for *context* patterns, attribute references help encode how attributes follow each other
- *references* to other *matched patterns*; this allows for construction of complex grammars where rules can be structured and reused
- *references* to *named entities* provided by other systems: these could include part-of-speech tags, parsed chunks or output from other IE/NER systems[1]

For *numeric* types, default value patterns for integer/float numbers are provided. Tabular, uniform, normal and mixture distributions are available to model attribute values. Linking a numeric attribute to unit definitions (e.g. to various currency units) will automatically create value patterns containing the numeric value surrounded by the units. In case there are multiple convertible units the extraction knowledge is reused.

For both attribute and class definitions, *axioms* can be specified that impose constraints on attribute value(s). For a single attribute, the axiom checks the to-be-extracted value and is either satisfied or not (which may boost or suppress the attribute candidate's score). For a class, each axiom may refer to all attribute values present in the partially or fully parsed instance. For example, a price with tax must be greater than the price without tax. Axioms can be authored using the JavaScript[2] scripting language. We chose JavaScript since it allows arbitrarily complex axioms to be constructed and also because the web community is used to it.

In addition, *formatting constraints* may be provided for each attribute. Currently, four types of formatting constraints are supported: (1) the whole attribute value is contained in a single parent, i.e. it does not include other tags or their boundaries; (2) the value fits into the parent; (3) the value does not cross any inline formatting elements; (4) it does not cross any block elements. We investigate how custom constraints could easily be added by users. By default, all four constraints are in effect and influence the likelihood of attribute candidates being extracted.

[1] So far we experimented with lemmatizers and POS taggers.
[2] http://www.mozilla.org/rhino

2.2 Class-related information

Each *class definition* enumerates the attributes which may belong to it, and for each attribute it defines a *cardinality* range. Extraction knowledge may address both content and context of the class. *Class content patterns* are analogous to the attribute value patterns, however, they may match *parts* of an instance and must contain at least one *reference* to a member attribute. Class content patterns may be used e.g. to describe common wordings used between attributes or just to specify attribute ordering. *Axioms* are used to constrain or boost instances based on whether their attributes satisfy the axiom. For each attribute, an *engagedness* parameter may be specified to estimate the apriori probability of the attribute joining a class instance (as opposed to standalone occurrence). Regarding class context, analogous *class context patterns* and similar *formatting constraints* as for attributes are in effect also for classes. An excerpt from an extraction ontology about computer monitor descriptions is shown in Fig. 1.

2.3 Extraction evidence parameters

All types of extraction knowledge mentioned above, i.e. value and context patterns, axioms, formatting constraints and ranges or distributions for numeric attribute values and for attribute content lengths, are essentially pieces of evidence indicating the presence (or absence) of a certain attribute or class instance. In *Ex*, every piece of evidence may be equipped with two probability estimates: precision and recall. The *precision* of evidence states how probable it is for the predicted attribute or class instance to occur given the evidence holds, disregarding the truth values of other evidence. For example, the precision of a left context pattern "person name: $" (where $ denotes the predicted attribute value) may be estimated as 0.8; i.e. in 80% of cases we expect a person name to follow in text after a match of the "person name:" string. The *recall* of evidence states how abundant the evidence is among the predicted objects, disregarding whether other evidence holds. For example, the "person name: $" pattern could have a low recall since there are many other contexts in which a person name could occur.

Pattern precision and recall can be estimated in two ways. First, annotated documents can be used to estimate both parameters using simple ratios of counts observed in text. In this case, it is necessary to smooth the parameters using an appropriate method. For a number of domains it is possible to find existing annotated data, e.g. web portals often make available online catalogs of manually populated product descriptions linking to the original sellers' web pages. When no training data is available or if the evidence seems easy to estimate, the user can specify both parameters manually. For the experimental results reported below we estimated parameters manually.

3 The extraction process

The inputs to the extraction process are the extraction ontology and a set of documents. Extraction consists of six stages depicted in Fig. 2.

```
<class id="Monitor">
 <pattern id="name_price_order" cover="0.8">
   $name (<tok/>{0,20} $price){1,4}
 </pattern>
 <axiom cover="1"> $price_with_tax &gt; $price_wo_tax </axiom>
 <attribute id="name" type="name" card="1" eng="0.70">
  <value>
   <pattern cover="0.5" p="0.8" ignore="case">
    (LCD (monitor|panel)?)? <pattern src="manuf.txt" ign="case"/>
    (<tok type="ALPHANUM|ALPHA|INT"/>|<tok case="UC"/>){1,2}
   </pattern>
   <length> <distribution min="1" max="7"/> </length>
   <pattern cover="0.5" type="fmt"> fits_in_parent </pattern>
   <pattern cover="1.0" type="fmt"> no_cross_blocks </pattern>
  </value>
 </attribute>
```

Fig. 1. Sample extraction ontology for computer monitors

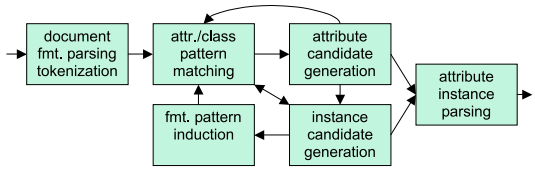


Fig. 2. Extraction process schema

3.1 Document preprocessing

First, the analysed document is loaded and its formatting structure is read into a simplified DOM tree of formatting objects. To robustly read web pages containing invalid HTML we employ the CyberNeko HTML parser[3]. Any text found in the formatting elements and their attributes is tokenized using a configurable tokenizer. A flat array of tokens is created for the document with each token linking to its parent formatting element. As part of tokenization, new words are registered in a common vocabulary, lemmatized and linked to their lemmas (if available), and classified by token type (e.g. alphanumeric) and case (e.g. capital).

3.2 Attribute candidate generation

After loading the document, all attribute value and attribute context patterns of the ontology are matched against the document's tokens. Where a value pattern matches, the

[3] `http://people.apache.org/~andyc/neko/doc/html/`

system attempts to create a new candidate for the associated attribute (attribute candidate – AC). If more value patterns match at the same place, or if there are context pattern matches for this attribute in neighbouring areas, then the corresponding evidence is turned on as well for the AC. Evidence corresponding to all other non-matched patterns is kept off for the AC. Also, during the creation of the AC, all other evidence types (axioms, formatting constraints, content length and numeric value ranges) are evaluated and set. The set of all evidence Φ_A known for the attribute A is used to compute a *conditional probability estimate* P_{AC} of how likely the AC is given all the observed evidence values:

$$P_{AC} = P(A|E \in \Phi_A) \tag{1}$$

The full formula is described and derived in [6]. We assume conditional independence of evidence given that the attribute holds or not. The AC is created only if P_{AC} exceeds a pruning threshold defined by the extraction ontology.

In places where a context pattern matches and there are no value pattern matches in neighbourhood, the system tries to create ACs of various length (in tokens) in the area pointed to by the context pattern. For patterns which include other attributes, we run the above process until no new ACs are generated.

The set of (possibly overlapping) ACs created during this phase is represented as an *AC lattice* going through the document, where each AC is scored by $score(AC) = log(P_{AC})$. Apart from the ACs which may span multiple tokens, the lattice also includes one 'background' state for each token that takes part in some AC. A background state BG_w for token w is scored as follows:

$$score(BG_w) = \min_{AC, w \in AC} log(\frac{1 - P(AC)}{|AC|}) \tag{2}$$

where $|AC|$ is the length of the AC in tokens. The extraction process can terminate here if no instance generation or formatting pattern induction is done, in which case all ACs on the best path through the lattice are extracted.

3.3 Instance candidate generation

At the beginning of the instance candidate (IC) generation phase, each AC is used to create a simple IC consisting just of that single AC. Then, a bottom-up IC generation algorithm is employed to generate increasingly complex ICs from the working set of ICs. At each step, the highest scoring (seed) IC is chosen and its neighbourhood is searched for ACs that could be added to it without breaking ontological constraints for the IC class. Only a *subset* of the constraints is taken into account at this time as e.g. some minimum cardinality constraints or axioms could never get satisfied initially. Each added AC is also examined to see whether it may corefer with some AC that is already present in the IC; if yes, it is only added as a reference and it does not affect the resulting IC score. To detect coreferences, the extraction ontology author may specify for each attribute a binary comparison function that compares two attribute values to determine whether they corefer (by default the values must equal to corefer).

After adding ACs to the chosen seed IC, that IC is removed from the working set and the newly created larger ICs are added to it. The seed IC is added to a *valid IC set* if

it satisfies *all* ontological constraints. As more complex ICs are created by combining simpler ICs with surrounding ACs, a limited number of ACs is allowed to be skipped (AC_{skip}) between the combined components, leading to a penalization of the created IC. The IC scores are computed based on their AC content and based on the observed values of evidence E known for the IC class C:

$$sc_1(IC) = exp(\frac{\sum_{AC \in IC} log(P_{AC}) + \sum_{AC_{skip} \in IC}(1 - log(P_{AC_{skip}}))}{|IC|}) \quad (3)$$

$$sc_2(IC) = P(C|E \in \Omega_C) \quad (4)$$

where $|IC|$ is the number of member ACs and Ω_C is the set of evidence known for class C; the conditional probability is estimated as in Eq. 1. By experiment we chose the Prospector [2] pseudo-bayesian method to combine the above into the final IC score:

$$score(IC) = \frac{sc_1(IC)sc_2(IC)}{sc_1(IC)sc_2(IC) + (1 - sc_1(IC))(1 - sc_2(IC))} \quad (5)$$

The IC generation algorithm picks the best IC to expand using the highest $score(IC)$. The generation phase ends when the working set of ICs becomes empty or on some terminating condition such as after a certain number of iterations or after a time limit has elapsed. The output of this phase is the set of valid ICs.

3.4 Formatting pattern induction

During the IC generation process, it may happen that a significant part of the created valid ICs satisfies some (apriori unknown) *formatting pattern*. For example, a contact page may consist of 6 paragraphs where each paragraph starts with a bold person name together with scientific degrees. A more obvious example would be a table with the first two columns listing staff first names and surnames. Then, if e.g. 90 person names are identified in such table columns and the table has 100 rows, the induced patterns make the remaining 10 entries more likely to get extracted as well.

Based on the lattice of valid ICs, the following *pattern induction* procedure is performed. First, the best scoring sequence of non-overlapping ICs is found through the lattice. Only the ICs on the best path take part in pattern induction. For each IC, we find its nearest containing formatting block element. We then create a subtree of formatting (incl. inline) elements between the containing block element (inclusive) and the attributes comprising the IC. This subtree contains the names of the formatting elements (e.g. paragraph or bold text) and their order within parent (e.g. the first or second cell in table row). Relative frequencies of these subtrees are calculated over the examined IC set (separately for each class if there are more). If the relative and absolute frequencies of a certain subtree exceed respective configurable thresholds, a new formatting pattern is induced and the subtree is transformed into a new *context pattern* indicating the presence of the corresponding class. This induced formatting context pattern is an example of 'local' evidence only useful within the currently analysed document (or a set of documents coming from the same source). The precision and recall of the induced context patterns are based on the relative frequencies with which the patterns hold in the document (or document set) with respect to the observed ICs.

The newly created context patterns are then fed back to the pattern matching phase, where they are matched and applied. This extra iteration rescores existing ACs and ICs and may as well yield new ACs and ICs which would not have been created otherwise. With our current implementation we have so far only experimented with pattern induction for ICs composed of a single attribute. Using this feature typically increases recall but may have adverse impact on precision. One approach to avoid degradation of precision is to provide attribute evidence which will prevent unwanted attributes from being extracted.

3.5 Attribute and instance parsing

The purpose of this final phase is to output the most probable sequence of instances and standalone attributes through the analysed document. The valid ICs are merged into the AC lattice so that each IC can be avoided by taking a path through standalone ACs or through background states. In the lattice, each IC is scored as $score(IC)|IC|$. This lattice is searched for *n* best sequences of non-overlapping extractable objects and these sequences are finally output. Consequently, the best path through the document may contain both instances and standalone attributes.

3.6 Incorporating third party tools

In practical WIE tasks it often happens that some of the attributes of interest are relatively easy to extract using manually specified evidence, some require *machine learning* algorithms such as CRFs [7] in order to achieve good extraction results, and some may benefit from a combination of both. To support all three cases, *Ex* allows named entity candidates identified by other engines to be included in all types of textual patterns described above. For example, suppose our task is to extract instances of a Person class composed of a person name and a scientific degree. Let's also suppose we have training data for person names but no data for degrees. A viable approach would then be to train e.g. a CRF classifier to identify person names in text and to specify evidence for degrees manually. To incorporate the CRF classifier's suggestions into the extraction ontology, a simple *attribute value pattern* like "${crf:personname}" can be added to the person name attribute. Here, ${} denotes a reference to an external named entity, *crf* is the source component name and *personname* is the identifier output by the CRF classifier. The precision for this value pattern can either be derived from the CRF classifier confidence score, or we can use the expected precision of the classifier for this attribute. To estimate the recall of the pattern, we can use the expected recall achieved by the classifier. Additionally to this pattern, the user may specify more patterns to correct (limit or extend) the classifier's suggestions.

4 Experimental Results

4.1 Contact Information on Medical Pages

In the EU (DG SANCO) MedIEQ project[4] we experiment with several dozens of medical website quality criteria, most of which are to be evaluated with the assistance of IE

[4] `http://www.medieq.org`

tools. One of them is the presence and richness of contact information. Table 1 shows early results for contact IE. The data set consists of 109 HTML documents, which were all manually classified as contact pages (each coming from a different website); in total there are 146 HTML files as some documents include frames or iframes. The documents contain 6930 annotated named entities of 10 types. The contact extraction ontology was written based on seeing the first 30 documents of the total data; it also refers to gazetteers such as lists of city names, common first names and surnames. The ontology contains about 100 textual patterns for the context and content of attributes and of the single extracted 'contact' class, attribute length distributions and several axioms. The effort spent on developing and tuning the ontology was about 2-3 person-weeks. In the strict mode of evaluation, only exact matches are considered to be successfully extracted. In the loose mode, partial credit is given to incomplete or overflown matches; e.g. extracting 'John Newman' where 'John Newman Jr.' was supposed to be extracted will count as a 66% match (based on overlapping word counts). The performance is probably underestimated since the reliability of manual annotation was very low: the inter-annotator agreement between the 3 human annotators was only 73.2% on average, and e.g. for person names it only reached 68.7%. We are working to fix these inconsistencies. Fig. 3 shows sample automatically annotated data.

Table 1. Contact IE results

	strict mode			loose mode		
attribute	prec	recall	F	prec	recall	F
title	0.71	0.82	0.76	0.78	0.86	0.82
name	0.66	0.51	0.58	0.74	0.56	0.64
street	0.62	0.52	0.56	0.85	0.67	0.75
city	0.47	0.73	0.57	0.48	0.76	0.59
zip	0.59	0.78	0.67	0.67	0.85	0.75
country	0.58	0.89	0.70	0.59	0.89	0.71
phone	0.97	0.84	0.90	0.99	0.87	0.93
email	1.00	0.99	1.00	1.00	0.99	1.00
company	0.57	0.37	0.44	0.81	0.51	0.63
dept.	0.51	0.31	0.38	0.85	0.45	0.59
overall	0.70	0.62	0.66	0.78	0.68	0.72

4.2 Weather Forecasts

Finally, we experimented with the domain of weather forecasts. Here our goal was to investigate the possibility to assist the ontology engineer in reusing existing *domain ontologies* in order to develop the extraction one/s. An advantage of this domain was the fact that several OWL ontologies were available for it. We analysed three of them by means of applying generic rules of two kinds:

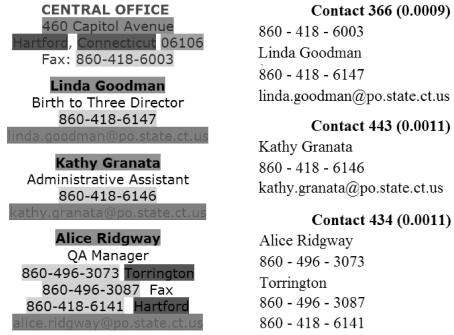


Fig. 3. Sample automatically annotated data; extracted instances on the right.

1. Rules suggesting the *core class/es* for the extraction ontology. As the extraction ontology for extraction from HTML-formatted text[5] is typically more class-centric and hierarchical than a properly-designed domain ontology, only few classes from the domain ontology are likely to become classes in the extraction ontology, while others become attributes that are dependent on the core class/es. For example, 'Day' is typically an attribute of a 'Forecast' class in an extraction ontology, while in the domain ontology they could easily be two classes connected by a relationship. One of such core class selection rules is, in verbal form, e.g. "Classes that appear more often in the domain than in the range of object properties are candidates for core class/es.".
2. Rules performing the actual *transformation*. Examples of such rules are e.g. "A data type property D of class C may directly yield an attribute of C." or "A set of mutually disjoint subclasses of class C may yield an attribute, whose values are these subclasses."

Most such independently formulated selection and transformation rules appeared as performing well in the initial experiment in the weather domain; details are in [5]. Transformation rules seemed, by first judgement, to suggest a sensible and inspiring, though by far not complete, skeleton of an extraction ontology. Testing this ontology on real weather forecast records is however needed for proper assessment.

In general, although the first experiments look promising, extensive usage of domain ontologies as starting point for extraction ontologies seems to be hindered by unavailability of high-quality domain ontologies for most domains, e.g. in relation to different categories of products or services, judging by the results of Swoogle-based[6] retrieval. This obstacle is likely to disappear in the not-so-distant future, as the semantic web technology becomes more widespread.

[5] This is not the case for extraction from free text, which is more relation-centric.
[6] `http://swoogle.umbc.edu`

5 Related Work

Most state-of-the-art WIE approaches focus on identifying structured collections of items (records), typically using inductively learnt models. Ontologies are often considered but rather as additional structures to which the extracted data are to be adapted after they have been acquired from the source documents, for the sake of a follow-up application [4]. There is no provision for directly using the rich structure of a domain-specific ontology in order to guide the extraction process. The approach to WIE that is inherently similar to ours (and from which we actually got inspiration in the early phase of our research) is that developed by Embley and colleagues at BYU [3]. The main distinctive features of our approach are: (1) the possibility to provide the extraction patterns with probability estimates (plus other quantitative info such as value distributions), allowing to calculate the weight for every attribute candidate as well as instance candidate; (2) the effort to combine hand-crafted extraction ontologies with other sources of information—HTML formatting and/or known data instances (3) the pragmatic distinction between extraction ontologies and domain ontologies proper: extraction ontologies can be arbitrarily adapted to the way domain data are typically *presented* on the web while domain ontologies address the domain as it is (but can be used as starting point for designing extraction ontologies). For similarly pragmatic reasons (easy authoring), we also used a proprietary XML syntax for extraction ontologies. An objective comparison between both approaches would require detailed experiments on a shared reference collection.

An approach to automatically discover new extractable attributes from large amounts of documents using statistical and NLP methods is described in [8]. On the other hand, formatting information is heavily exploited for IE from tables in [11]. Our system has a slightly different target; it should allow for fast IE prototyping even in domains where there are few documents available and the content is semi-structured. While our system relies on the author to supply coreference resolution knowledge for attribute values, advanced automatic methods are described e.g. in [13]. A system described in [12] uses statistical methods to estimate the mutual affinity of attribute values.

Our ideas and experiments on domain ontology selection and transformation to extraction ontology are related to the generic research in ontology selection [9] and content evaluation[7], especially with respect to the notion of intra-ontology concept centrality; this relationship deserves further study.

6 Conclusions

The *Ex* system attempts to unify the often separate phases of WIE and ontology population. Multiple sources of extraction knowledge can be combined: manually encoded knowledge, knowledge acquired from annotated data, and knowledge induced from common formatting patterns by the means of wrapper induction. An alpha version of *Ex* (incl. extraction ontology samples) is publicly available[8].

[7] `http://km.aifb.uni-karlsruhe.de/ws/eon2006/`
[8] `http://eso.vse.cz/~labsky/ex`

Future work will concentrate on the integration with trainable machine learning algorithms, on improving the extraction results and on covering more domains. We currently experiment with IE from online product descriptions, where we develop an extraction ontology for each type of examined product. Typically extracted attributes include product name, price, picture and multiple product-specific attributes. In order to obtain annotated data, we cooperate with one of the largest Czech web portals. Both instance parsing and formatting pattern induction algorithms need improvement in accuracy and speed. We also plan to investigate how text mining over the extraction results could help us identify 'gaps' in the ontology, e.g. non-labelled tokens frequently appearing inside a 'cloud' of annotations are likely to be unrecognised important values. Finally, we intend to provide support for semi-automated transformation of domain ontologies to extraction ones.

The research was partially supported by the EC under contract FP6-027026, Knowledge Space of Semantic Inference for Automatic Annotation and Retrieval of Multimedia Content - K-Space. The medical website application is carried out in the context of the EC-funded (DG-SANCO) project MedIEQ.

References

1. Ciravegna, F.: LP2 – an adaptive algorithm for information extraction from web-related texts. In: Proc IJCAI-2001.
2. Duda, R.O., Gasching, J., and Hart, P.E. Model design in the Prospector consultant system for mineral exploration. In: Readings in Artificial Intelligence, pp. 334–348, 1981.
3. Embley, D. W., Tao, C., Liddle, D. W.: Automatically extracting ontologically specified data from HTML tables of unknown structure. In *Proc. ER '02*, pp. 322–337, London, UK, 2002.
4. Kiryakov, A., Popov, B., Terziev, I., Manov, D., Ognyanoff, D.: Semantic annotation, indexing, and retrieval. In: *J. Web Sem.*, volume 2, pp. 49–79, 2004.
5. Labsky, M., Nekvasil, M., Svatek, V.: Towards Web Information Extraction using Extraction Ontologies, Domain Ontologies, and Inductive Learning. Accepted as poster paper for Proc. of K-CAP 2007, Whistler, Canada, ACM 2007.
6. Labsky, M., Svatek, V: Information extraction with presentation ontologies. Technical report, KEG UEP, `http://eso.vse.cz/~labsky/ex/ex.pdf`.
7. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proc. 18th International Conf. on Machine Learning, pp. 282–289. Morgan Kaufmann, San Francisco, CA, 2001.
8. Popescu, A., Etzioni, O.: Extracting Product Features and Opinions from Reviews. In: Proc. EMNLP 2005.
9. Sabou, M., Lopez, V., Motta, E.: Ontology selection for the real semantic web: How to cover the queen's birthday dinner? In: Proc. EKAW 2006. Springer LNCS, 2006.
10. Svatek, V., Labsky, M., Vacura, M.: Knowledge Modelling for Deductive Web Mining. In: Proc. EKAW 2004, Springer Verlag, LNCS, 2004.
11. Wei, X., Croft, B., McCallum, A.: Table Extraction for Answer Retrieval. In: Information Retrieval Journal, vol. 9, issue 5, pp. 589-611, 2006.
12. Wick, M., Culotta, A., McCallum, A.: Learning Field Compatibilities to Extract Database Records from Unstructured Text. In: Proc. EMNLP, 2006.
13. Yates, A., Etzioni, O.: Unsupervised Resolution of Objects and Relations on the Web. In: Proc. HLT 2007.

On Ontologies as Prior Conceptual Knowledge in Inductive Logic Programming

Francesca A. Lisi and Floriana Esposito

Dipartimento di Informatica, Università degli Studi di Bari
Via E. Orabona 4, 70125 Bari, Italy
{lisi, esposito}@di.uniba.it

Abstract. In this paper we provide a survey of Inductive Logic Programming (ILP) attempts at using Ontologies as prior conceptual knowledge. In particular, we take a critical look at two ILP proposals based on knowledge representation frameworks that integrate Description Logics and Horn Clausal Logic and draw from them general conclusions that can be considered as guidelines for an upcoming Onto-Relational Learning aimed at extending Relational Learning to account for Ontologies.

1 Introduction

During the last decade increasing attention has been paid on *ontologies* and their role in Intelligent Systems as a means for conveying conceptual knowledge [6]. An ontology is a formal explicit specification of a shared conceptualization for a domain of interest [10]. Among the other things, this definition emphasizes the fact that an ontology has to be specified in a language that comes with a formal semantics. Only by using such a formal approach ontologies provide the machine interpretable meaning of concepts and relations that is expected when using an ontology-based approach. In Artificial Intelligence, an ontology refers to an engineering artifact (more precisely, produced according to the principles of *Ontological Engineering* [9]), constituted by a specific vocabulary used to describe a certain reality, plus a set of explicit assumptions regarding the intended meaning of the vocabulary words. This set of assumptions has usually the form of a First Order Logic (FOL) theory, where vocabulary words appear as unary or binary predicate names, respectively called concepts and relations. In the simplest case, an ontology describes a hierarchy of concepts related by subsumption relationships; in more sophisticated cases, suitable axioms are added in order to express other relationships between concepts and to constrain their intended interpretation. Among the formalisms proposed by Ontological Engineering, the most currently used are *Description Logics* (DLs) [1].

Prior conceptual knowledge is also a core ingredient in Inductive Logic Programming (ILP) [21]. ILP was born at the intersection of Concept Learning [20] and Logic Programming [19]. Thus it has been historically concerned with rule induction from examples within the representation framework of Horn Clausal Logic (HCL) and with the aim of prediction. Currently ILP covers a broader area

Table 1. Syntax and semantics of basic DLs.

bottom (resp. top) concept	$\bot$ (resp. $\top$)	$\emptyset$ (resp. $\Delta^{\mathcal{I}}$)
atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
(abstract) role	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
(abstract) individual	a	$a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
concept negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
concept intersection	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
concept union	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
value restriction	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y\ (x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
existential restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y\ (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
at least number restriction	$\geq nR$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert\{y \lvert (x,y) \in R^{\mathcal{I}}\}\rvert \geq n\}$
at most number restriction	$\leq nR$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert\{y \lvert (x,y) \in R^{\mathcal{I}}\}\rvert \leq n\}$
concept equivalence axiom	$C_1 \equiv C_2$	$C_1^{\mathcal{I}} = C_2^{\mathcal{I}}$
concept subsumption axiom	$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$
concept assertion	$a : C$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role assertion	$\langle a, b\rangle : R$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

of research investigating - among other things - novel tasks like the descriptive ones that are more peculiar to Data Mining and novel approaches like the ones collectively known as Statistical Relational Learning. Though the use of background knowledge has been widely recognized as one of the strongest points of ILP when compared to other forms of Concept Learning, the background knowledge in ILP is often not organized around a well-formed conceptual model. This practice seems to ignore the growing demand for an ontological foundation of knowledge in intelligent systems. Rather, it highlights some difficulties in accommodating ontologies in ILP. Indeed the underlying Knowledge Representation (KR) frameworks (DLs and HCL, respectively) are deeply different in several respects but can be combined according to some limited forms of hybridization [23]. In this paper we take a critical look at ILP attempts at using ontologies as prior conceptual knowledge by adopting these hybrid KR frameworks.

The paper is organized as follows. Section 2 provides the basic notions of DLs. Section 3 briefly describes different forms of integration of DLs and HCL. Section 4 compares two ILP frameworks for hybrid DL-HCL formalisms. Section 5 concludes the paper with final remarks.

2 Ontologies and Description Logics

When a DL-based ontology language is adopted, an ontology is nothing else than a DL knowledge base (KB). DLs are a family of decidable FOL fragments that allow for the specification of knowledge in terms of classes (*concepts*), binary relations between classes (*roles*), and instances (*individuals*) [3]. Complex con-

cepts can be defined from atomic concepts and roles by means of constructors (see Table 1). E.g., concept descriptions in the basic DL $\mathcal{AL}$ are formed according to only the constructors of atomic negation, concept conjunction, value restriction, and limited existential restriction. The DLs $\mathcal{ALC}$ and $\mathcal{ALN}$ are members of the $\mathcal{AL}$ family. The former extends $\mathcal{AL}$ with (arbitrary) concept negation (also called complement and equivalent to having both concept union and full existential restriction), whereas the latter with number restriction. A DL KB can state both is-a relations between concepts (*axioms*) and instance-of relations between individuals (resp. couples of individuals) and concepts (resp. roles) (*assertions*). The semantics of DLs is defined through a mapping to FOL. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ for a DL KB consists of a domain $\Delta^{\mathcal{I}}$ and a mapping function $\cdot^{\mathcal{I}}$. In particular, individuals are mapped to elements of $\Delta^{\mathcal{I}}$ such that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b$ (*Unique Names Assumption* (UNA) [22]). Also the KB represents many different interpretations, i.e. all its models. This is coherent with the *Open World Assumption* (OWA) that holds in FOL semantics. The main reasoning task for a DL KB is the *consistency check* that is performed by applying decision procedures based on tableau calculus.

3 Ontologies and Logic Programming

The integration of Ontologies and Logic Programming follows the tradition of KR research on *hybrid systems*, i.e. those systems which are constituted by two or more subsystems dealing with distinct portions of a single KB by performing specific reasoning procedures [8]. The motivation for investigating and developing such systems is to improve on two basic features of KR formalisms, namely *representational adequacy* and *deductive power*, by preserving the other crucial feature, i.e. *decidability*. In particular, combining DLs with HCL can easily yield to undecidability if the interface between them is not reduced.

$\mathcal{AL}$-log [7] is a hybrid KR system that integrates $\mathcal{ALC}$ and DATALOG [5]. In particular, variables occurring in the body of rules may be constrained with $\mathcal{ALC}$ concept assertions to be used as 'typing constraints'. This makes rules applicable only to explicitly named objects. Reasoning for $\mathcal{AL}$-log knowledge bases is based on *constrained SLD-resolution*, i.e. an extension of SLD-resolution with a tableau calculus for $\mathcal{ALC}$ to deal with constraints. Constrained SLD-resolution is *decidable* and runs in single non-deterministic exponential time. Constrained SLD-refutation is a complete and sound method for answering *ground* queries.

A comprehensive study of the effects of combining DLs and HCL can be found in [13]. Here the family **Carin** of hybrid languages is presented. Special attention is devoted to the DL $\mathcal{ALCNR}$. The results of the study can be summarized as follows: (i) answering conjunctive queries over $\mathcal{ALCNR}$ TBoxes is decidable, (ii) query answering in a logic obtained by extending $\mathcal{ALCNR}$ with non-recursive DATALOG rules, where both concepts and roles can occur in rule bodies, is also decidable, as it can be reduced to computing a union of conjunctive query answers, (iii) if rules are recursive, query answering becomes undecidable, (iv) decidability can be regained by disallowing certain combinations of con-

structors in the logic, and (v) decidability can be regained by requiring rules to be *role-safe*, where at least one variable from each role literal must occur in some non-DL-atom. As in $\mathcal{AL}$-log, query answering is decided using constrained resolution and a modified version of tableau calculus.

Besides decidability, another relevant issue is *DL-safeness* [23]. A safe interaction between the DL and the HCL part of an hybrid KB allows to solve the semantic mismatch between DLs and HCL, namely the OWA for DLs and the CWA for HCL[1]. In this respect, $\mathcal{AL}$-log is DL-safe whereas CARIN is not.

4 Ontologies in Inductive Logic Programming

Learning in DL-HCL hybrid languages has very recently attracted some attention in the ILP community. Two ILP frameworks have been proposed that adopt a hybrid representation for both hypotheses and background knowledge.

In [24], the chosen language is CARIN-$\mathcal{ALN}$. The framework focuses on discriminant induction and adopts the ILP setting of learning from interpretations. The target concept is a unary DATALOG predicate, therefore hypotheses are represented as CARIN-$\mathcal{ALN}$ rules with a DATALOG literal in the head. The coverage relation of hypotheses against examples and the generality relation between two hypotheses are based on the existential entailment algorithm of CARIN. In particular, the generality relation is defined as an extension of Buntine's generalized subsumption [4]. Following [24], Kietz studies the learnability of CARIN-$\mathcal{ALN}$, thus providing a pre-processing method which enables ILP systems to learn CARIN-$\mathcal{ALN}$ rules [12].

In [14], the representation and reasoning means come from $\mathcal{AL}$-log. Hypotheses are represented as constrained DATALOG clauses that are linked, connected (or range-restricted), and compliant with the bias of Object Identity (OI)[2]. Note that this framework is general, meaning that it is valid whatever the scope of induction (description/prediction) is. Therefore the literal in the head of hypotheses represents a concept to be either discriminated from others (*discriminant induction*) or characterized (*characteristic induction*). The generality relation for one such hypothesis language is an adaptation of generalized subsumption [4], named $\mathcal{B}$-subsumption, to the $\mathcal{AL}$-log KR framework. It gives raise to a quasi-order and can be checked with a decidable procedure based on constrained SLD-resolution [16]. Coverage relations for both ILP settings of learning from interpretations and learning from entailment have been defined on the basis of query answering in $\mathcal{AL}$-log [15]. As opposite to [24], the framework has been implemented in an ILP system [18]. More precisely, an instantiation of it for the case of *characteristic induction from interpretations* has been considered. Indeed, the system supports a variant of a very popular data mining task - frequent pattern discovery - where rich prior conceptual knowledge is taken into

[1] Note that the OWA and CWA have a strong influence on the results of reasoning.

[2] The OI bias can be considered as an extension of the UNA from the semantic level to the syntactic one of $\mathcal{AL}$-log. It can be the starting point for the definition of either an equational theory or a quasi-order for constrained DATALOG clauses.

account during the discovery process in order to find patterns at multiple levels of description granularity. The search through the space of patterns represented as unary conjunctive queries in $\mathcal{AL}$-log and organized according to $\mathcal{B}$-subsumption is performed by applying an ideal downward refinement operator [17].

5 Conclusions and Future Directions

In this position paper, we have provided a brief survey of ILP literature addressing the issues raised by having ontologies as prior knowledge. From the comparative analysis of [24] and [14], a common feature emerges. Both proposals resort to Buntine's generalized subsumption, being it the one even in ILP that can actually exploit prior knowledge. We would like to emphasize that in both the extension of [4] to hybrid DL-HCL formalisms is not trivial.

Ontologies and DLs are playing a relevant role in the definition of the *Semantic Web* [2]. Indeed, the design of the ontology language for the Semantic Web, OWL[3], has been based on the very expressive DL $\mathcal{SHOIN}(\mathbf{D})$ [11]. Whereas OWL is already undergoing the standardization process at W3C, the debate around a unified language for *rules* is still ongoing. There are proposals trying to extend OWL with constructs inspired to HCL in order to 'to build rules on top of ontologies'. Acquiring these rules is a task that can be automated by applying Machine Learning/Data Mining algorithms conceived for hybrid DL-HCL formalisms. We would like to emphasize that the *DL-safeness* and the *decidability* of these formalisms are two desirable properties which are particularly appreciated both in ILP and in the Semantic Web application area. Yet, following the guidelines of [24] and [14], new ILP frameworks can be designed to deal with more expressive hybrid DL-HCL languages. The augmented expressive power may be due to a more expressive DL (than $\mathcal{ALC}$ and $\mathcal{ALN}$), or a more expressive HCL fragment (than DATALOG), or a looser integration between the DL and the HCL parts. An important requirement will be the definition of a *semantic* generality relation for hypotheses as highlighted by the comparative analysis of [24] and [14]. Each of these frameworks will contribute to the upcoming Onto-Relational Learning aimed at extending Relational Learning to account for ontologies in a clear, well-founded and systematic way analogously to what has been done in Statistical Relational Learning.

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
2. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May, 2001.
3. A. Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1–2):353–367, 1996.

[3] `http://www.w3.org/2004/OWL/`

4. W. Buntine. Generalized subsumption and its application to induction and redundancy. *Artificial Intelligence*, 36(2):149–176, 1988.
5. S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases*. Springer, 1990.
6. B. Chandrasekaran, J.R. Josephson, and V.R. Benjamins. What are ontologies, and why do we need them? *IEEE Intelligent Systems*, 14(1):20–26, 1999.
7. F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. $\mathcal{AL}$-log: Integrating Datalog and Description Logics. *J. of Intelligent Information Systems*, 10(3):227–252, 1998.
8. A.M. Frisch and A.G. Cohn. Thoughts and afterthoughts on the 1988 workshop on principles of hybrid reasoning. *AI Magazine*, 11(5):84–87, 1991.
9. A. Gómez-Pérez, M. Fernández-López, and O. Corcho. *Ontological Engineering*. Springer, 2004.
10. T. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220, 1993.
11. I. Horrocks, P.F. Patel-Schneider, and F. van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.
12. J.-U. Kietz. Learnability of description logic programs. In S. Matwin and C. Sammut, editors, *Inductive Logic Programming*, volume 2583 of *Lecture Notes in Artificial Intelligence*, pages 117–132. Springer, 2003.
13. A.Y. Levy and M.-C. Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104:165–209, 1998.
14. F.A. Lisi. Principles of Inductive Reasoning on the Semantic Web: A Framework for Learning in $\mathcal{AL}$-log. In F. Fages and S. Soliman, editors, *Principles and Practice of Semantic Web Reasoning*, volume 3703 of *Lecture Notes in Computer Science*, pages 118–132. Springer, 2005.
15. F.A. Lisi and F. Esposito. Efficient Evaluation of Candidate Hypotheses in $\mathcal{AL}$-log. In R. Camacho, R. King, and A. Srinivasan, editors, *Inductive Logic Programming*, volume 3194 of *LNAI*, pages 216–233. Springer, 2004.
16. F.A. Lisi and D. Malerba. Bridging the Gap between Horn Clausal Logic and Description Logics in Inductive Learning. In A. Cappelli and F. Turini, editors, *AI*IA 2003: Advances in Artificial Intelligence*, volume 2829 of *Lecture Notes in Artificial Intelligence*, pages 49–60. Springer, 2003.
17. F.A. Lisi and D. Malerba. Ideal Refinement of Descriptions in $\mathcal{AL}$-log. In T. Horvath and A. Yamamoto, editors, *Inductive Logic Programming*, volume 2835 of *Lecture Notes in Artificial Intelligence*, pages 215–232. Springer, 2003.
18. F.A. Lisi and D. Malerba. Inducing Multi-Level Association Rules from Multiple Relations. *Machine Learning*, 55:175–210, 2004.
19. J.W. Lloyd. *Foundations of Logic Programming*. Springer, 2nd edition, 1987.
20. T.M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.
21. S.-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Artificial Intelligence*. Springer, 1997.
22. R. Reiter. Equality and domain closure in first order databases. *Journal of ACM*, 27:235–249, 1980.
23. R. Rosati. Semantic and computational advantages of the safe integration of ontologies and rules. In F. Fages and S. Soliman, editors, *Principles and Practice of Semantic Web Reasoning*, volume 3703 of *Lecture Notes in Computer Science*, pages 50–64. Springer, 2005.
24. C. Rouveirol and V. Ventos. Towards Learning in CARIN-$\mathcal{ALN}$. In J. Cussens and A. Frisch, editors, *Inductive Logic Programming*, volume 1866 of *Lecture Notes in Artificial Intelligence*, pages 191–208. Springer, 2000.

Using prior knowledge in biological pathway discovery
(Invited Talk)

Stephen Muggleton

Imperial College London, London, UK
shm@doc.ic.ac.uk

Abstract. Within the new area of Systems Biologists there is widespread use of graph-based descriptions of bio-molecular interactions which describe cellular activities such as gene regulation, metabolism and transcription. Biologists build and maintain these network models based on the results of experiments in wild-life and mutated organisms. This presentation will provide an overview of recent ILP research in Systems Bioloogy, concentrating on the use of encoded prior knowledge. Indeed one of the key advantages of ILP in this area is the availability of background knowledge on existing known biochemical networks from publicly available resources such as KEGG (used in data sets such as those in the Nature paper by Bryant, King, Muggleton, etc). The topic has an inherent importance owing to its application in biology and medicine. Moreover, descriptions have an inherent relational structure in the form of spatial and temporal interactions of the molecules involved. We will argue the requirements for rich probabilistic logic-based representations which can be machine learned. From a logical perspective the objects include genes, proteins, metabolites, inhibitors and cofactors. The relationships include biochemical reactions in which one set of metabolites is transformed to another mediated by the involvement of an enzyme. One of the representational challenges is that within various databases the same object can be referred to in several ways, which brings in the problem of identity uncertainty. The available genomic information is also very incomplete concerning the functions and even the existence of genes and metabolites, leading to the necessity of techniques such as logical abduction to introduce novel functions and even invention of new objects.

Evaluation of GUHA Mining with Background Knowledge

Martin Ralbovský

Department of Information and Knowledge Engineering,
University of Economics, Prague, W. Churchill Sq. 4, 130 67 Praha 3, Czech Republic
martin.ralbovsky@gmail.com

Abstract. Background knowledge is used for evaluation of specific KDD technique – GUHA method. This is done by verification of verbal background knowledge rules on a medical STULONG dataset. Formalization for the verbal rules was developed and tools for verification of the rules against output of GUHA procedures implemented. We conducted experiments that and drew conclusions about the mostly used settings of GUHA procedures.

Keywords: Background knowledge, GUHA Method, STULONG dataset

1 Introduction

Process of knowledge discovery in databases (KDD) can be affected by using domain knowledge. In [16] authors identify four KDD stages where proper domain knowledge (ontologies) can be helpful: data understanding, task design, result interpretation and result dissemination over the semantic web. In this work, we are interested in evaluation of KDD techniques with respect to the used domain knowledge and examined data. The evaluation should help to improve the task design and result interpretation KDD stages.

We are using the STULONG[1] database as the examined data. The STULONG database is an extensive epidemiological study of atherosclerosis primary prevention and was examined also in [16]. Besides the data, STULONG contains some domain knowledge examples created by medical experts. The knowledge (here named *background knowledge*) consists of verbal rules expressing relationships between two entities in the domain.

Because of the fact, that most of the data mining analysis with STULONG were done with tools implementing GUHA method, we chose this method to be evaluated by the background knowledge. By evaluation we mean constructing various data mining tasks that should approve or disapprove the background knowledge in the STULONG data and drawing conclusions from the results of the tasks. We invented a formalization of verbal background knowledge rules and implemented automatic tools to verify them against the outputs of GUHA

[1] http://euromise.vse.cz/stulong

mining tasks. To our best knowledge, this work is the first work to evaluate GUHA mining on bases of comprehensive background knowledge verification.

The work is structured as follows: section 2 describes the GUHA method, GUHA procedures used in this work and also recent tools implementing the method. Section 3 explains background knowledge used, new formalization of the background knowledge and example of the formalization. Section 4 shows conducted experiments and evaluates the GUHA method on basis of the experiments. Section 5 puts the work into context of other works dealing with background knowledge and section 6 concludes the work and gives ideas about future research.

2 The GUHA Method

GUHA method is one of the first methods of exploratory data analysis, developed in the mid-sixties in Prague. It is a general mainframe for retrieving interesting knowledge from data. The method has firm theoretical foundations based on observational calculi and statistics [5]. For purpose of this work let us explain only the basic principles of the method, as shown in Figure 1.

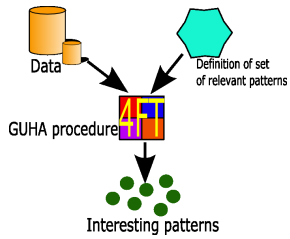


Fig. 1. The GUHA method

GUHA method is realized by GUHA procedures such as 4FT procedure to be described, located in the middle of the figure. Inputs of the procedure are data and a simple definition of a possibly large set of relevant patterns. The procedure automatically generates all the relevant patterns and verifies them against the provided data. Patterns that are true are output of the procedure. In this work, we use procedure 4FT (described in section 2.1) and procedure KL (described in section 2.2).

2.1 Procedure 4FT

Classical *apriori* [1] association mining searches rules in form $X \longrightarrow Y$, where X and Y are sets of items. Procedure 4FT searches (in the simplified form) for generalized association rules in form $\varphi \approx \psi$, where φ and ψ are *Boolean attributes*

and $\approx$ is a *4ft-quantifier*. Relation $\varphi \approx \psi$ is evaluated on the basis of *4ft table*, as shown in Table 1.

The term *Boolean attribute* uses attributes. We use the term attribute in the sense of *categorial attribute*, i.e. attribute with finite number of values. Let A be an attribute, $\mathrm{A} = \{a_1, a_2 ... a_n\}$ and $\alpha \subset A$, $\alpha \neq \emptyset$. Then $A(\alpha)$ is a *basic Boolean attribute*.

Each *basic Boolean attribute* is a *Boolean attribute*. If α and β are *Boolean attributes*, $\alpha \wedge \beta$, $\alpha \vee \beta$ and $\neg\alpha$ are *Boolean attributes*.

M	ψ	$\neg\psi$
φ	a	b
$\neg\varphi$	c	d

Table 1: 4ft table

Table 1. 4FT contingency table

A *4ft table* is a quadruple of natural numbers $\langle a,\ b,\ c,\ d \rangle$ so that:

- a: number of objects (rows of M) satisfying φ and ψ
- b: number of objects (rows of M) satisfying φ and not satisfying ψ
- c: number of objects (rows of M) not satisfying φ but satisfying ψ
- d: number of objects (rows of M) satisfying neither φ nor ψ

4ft-quantifier expresses kind of dependency between φ and ψ. The quantifier is defined as a condition over the 4ft table. In this work, we use the two most common 4ft-quantifiers: *founded implication* and *above average dependence*.

The *founded implication* is the basic quantifier for the 4FT procedure. It is defined by the following condition:

$$a \geq Base \wedge \frac{a}{a+b} \geq p$$

where *Base* and p are threshold parameters of the procedure. The *Base* parameter represents absolute number of objects that satisfies φ. In our work we will use relative *Base* representation, $\frac{a}{a+b+c+d}$. The *Base* parameter corresponds to the *support* and p to the *confidence* parameters of classical association mining.

The *above average dependence* is defined by the following condition:

$$\frac{a}{a+b} \geq (p)\frac{a+c}{a+b+c+d} \wedge a \geq Base$$

where p and *Base* are user-defined parameters[2]. Again, we will use the relative *Base* representation $\frac{a}{a+b+c+d}$. So, the quantifier can be verbally interpreted as

[2] The p parameter is originally defined in [12] as $\frac{a}{a+b} \geq (1+p)\frac{a+c}{a+b+c+d}$. We alter this definition in order to avoid negative p results in the experiments.

"among object satisfying φ, there are at least p per cent more objects satisfying ψ then among all observed objects and there are at least Base per cent of observed objects satisfying φ and ψ".

2.2 Procedure KL

Procedure KL [13] searches (in the simplified form) for rules in form $R \sim C$, where R and C are categorial attributes. The symbol $\sim$ is called *KL-quantifier*. The rule $R \sim C$ means, that categorial attributes R and C are in relation described by $\sim$. In this work, we are using the *Kendall's quantifier*.

Kendall's quantifier is based on *Kendall's coefficient* τ_b[15]. It is defined as

$$\tau_b = \frac{2(P-Q)}{\sqrt{(n^2 - \sum_k n_{k,*}^2)(n^2 - \sum_l n_{*,l}^2)}}$$

where

$$P = \sum_k \sum_l n_{k,l} \sum_{i>k} \sum_{j>l} n_{i,j}, Q = \sum_k \sum_l n_{k,l} \sum_{i>k} \sum_{j<l} n_{i,j}$$

τ_b ranges from $\langle -1, 1 \rangle$, where values $\tau_b > 0$ indicate positive ordinal dependence[3], values $\tau_b < 0$ negative ordinal dependence, $\tau_b = 0$ ordinal independence and $|\tau_b| = 1$ functional dependence of C on R. In this work, we are using the Kendall's quantifier to construct *abstract quantifiers* discussed in section 3.1.

2.3 GUHA Tools

Apart from the tools presented in this section, several systems implementing GUHA procedures were developed in the past. In recent years, the *LISp-Miner* system has been the most significant GUHA tool. This system has been under development since 1996 at the University of Economics, Prague. It includes six GUHA procedures including procedure KL and lighter version of 4FT procedure [11] in addition to other data preparation and result interpretation modules.

In 2004, the Ferda project started as an initiative to build a new visual data mining successor of the *LISp-Miner* system. Creators (at the Faculty of Mathematics and Physics, Charles University, Prague) succeeded in developing an user friendly visual system with advanced features such as high level modularity, support for distributed computing or reusability of the task setting [6]. At present there are several research activities taking advantage of the system. Figure 2 shows the Ferda working environment. For purposes of this work, there were modules implemented in the Ferda system as well.

[3] High values of C often coincide with high values of R, low values of C often coincide with low values of R.

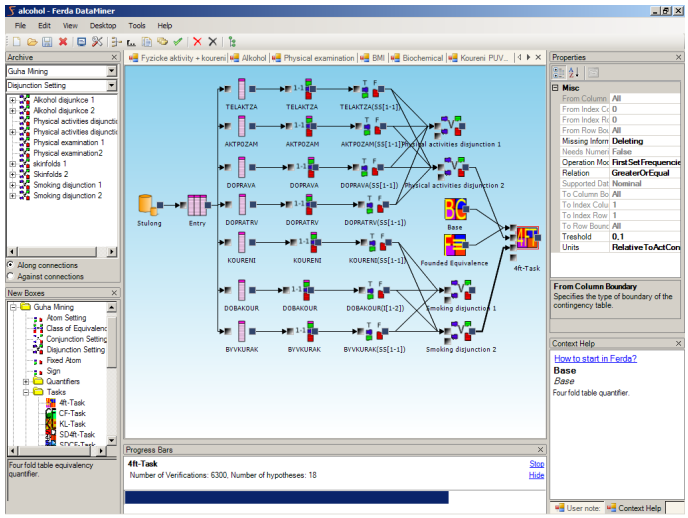


Fig. 2. Ferda environment

3 Background Knowledge

3.1 Considered Background Knowledge Types

Background knowledge (also *field knowledge* or *prior knowledge*) is knowledge that comes from the user or a community of users and integrates knowledge, which the community can agree upon and consider it common. Various fields of KDD define background knowledge differently, there is no central theory for the term. In the context of GUHA mining, we think of background knowledge as a part of domain knowledge, knowledge that is specific to particular domains (medicine, chemistry, etc.). We define background knowledge as a set of various verbal rules that are accepted in a specific domain as a common knowledge[4]. The rule can describe functional dependence between quantities, relationship between entities or say something about their behavior. Below are presented example rules taken from STULONG[5]:

– If education increases, wine consumption increases as well.

[4] Note the difference between our vague definition and precise definitions e.g. in ILP

[5] The study (STULONG) was realized at the 2nd Department of Medicine, 1st Faculty of Medicine of Charles University and Charles University Hospital, U nemocnice 2, Prague 2 (head. Prof. M. Aschermann, MD, SDr, FESC), under the supervision of Prof. F. Boudík, MD, ScD, with collaboration of M. Tomečková, MD, PhD and Ass. Prof. J. Bultas, MD, PhD. The data were transferred to the electronic form by the European Centre of Medical Informatics, Statistics and Epidemiology of Charles

– Patients with greater responsibility in work tend to drive to work by car.

3.2 Background Knowledge Formalization

In order to automatically verify background knowledge against the data, a new formalization needed to be thought out. Background knowledge contains heterogeneous verbal formulations of dependences and relationships in the domain. The relevance and validity of the formulations varies: the relationships in physics are formed exactly by mathematical equations, but for example in sociology they mean only expected behavior or opinion of a group of people. Our aim is to find formalization usable for both domains.

We present a new *Formalization with attributes, validation literals and abstract quantifiers* first used in [10]. The main idea behind the formalization is to make it as close to GUHA terms as possible while still enabling large expressive possibilities of the verbal rule. Because of shorter format of the article, we present only an overview and an example of the new formalization with a little reasoning. The topic is fully covered in [10], section 3.2.2.

Attribute is the basic term for the new formalization. *Attribute* is defined as a result of domain categorization and is used to create *categorial attributes*, inputs of the *KL* procedure.

Validation literal is a special type of *literal* used to express background knowledge. *Literal* is a basic Boolean attribute or its negation. We define the *literal length* as the size of the categories' subset. *Validation literal* is a literal, which has *literal length* equal to 1.

Abstract quantifier is a generalization of a quantifier or quantifiers of a procedure (4FT or KL). The idea behind abstract quantifiers is to create a "black-box" quantifier: user does not need to fill any numeral parameters of the quantifier. The quantifier is then more suitable for transferring verbal background knowledge rules into formalized form.

3.3 Formalization Example

With all the terms explained, let us see how the formalization is applied to a specific verbal rule **If education increases, wine consumption increases as well.** as presented in Section 3.1. The rule defines relationship between two measurable quantities of a patient. These quantities are stored in the database in the form of columns of a table, so attributes can be created. We name the attributes for **education** and **wine consumption** *education* and *wine* respectively.

For this paragraph we will consider only the KL procedure. The procedure searches (in the simplified form) for rules in form $R \sim C$, where R and C

University and Academy of Sciences (head. Prof. RNDr. J. Zvárová, DrSc). The data resource is on the web pages `http://euromise.vse.cz/challenge2004`. At present time the data analysis is supported by the grant of the Ministry of Education CR Nr LN 00B 107.

are categorial attributes, which derive from attributes. When **education** and **wine consumption** out of the rule are to be formalized with R and C of the hypothesis, then the part **If ... increases, ... increases as well** could be formalized with a proper abstract quantifier. We call this quantifier *increasing dependence* and is implemented as a special setting of the Kendall quantifier (to be described later). With all the knowledge stated above, the rule **If education increases, wine consumption increases as well** can be formally written as *education* $\uparrow$ *wine*, where $\uparrow$ states for increasing dependence abstract quantifier.

We can also define the formalization for the 4FT procedure. The hypotheses of this procedure consist of Boolean attributes, therefore it is better to use validation literals. If we presume correct categorization, out of *attributes education* and *wine* the *validation literals education(HIGH)* and *wine(HIGH)* can be created. Similarly to KL formalization we can use abstract quantifier to note the dependence. Then the rule **If education increases, wine consumption increases as well** can be formalized as *education(HIGH)* $\Rightarrow$ *wine(HIGH)* with a proper abstract quantifier $\Rightarrow$

Formalization with the 4FT procedure cannot consider the whole *attributes* but only some of its categories, thus it is weaker. However there may be situations when it is feasible to use the 4FT procedure. If the examined attribute is not ordinal, the KL procedure cannot be used. Also there may be ordinal attributes with such a small number of categories, that is preferred to use 4FT procedure (which was often the case in our experiments).

In the beginning of section 3.2, a requirement was given on the formalization to be able to represent various kinds of relationships between the entities of the domain. The formalization with attributes, validation literals and abstract quantifiers fulfills this requirement, because the formalization does not pose any restrictions on the relationships - the relationship is expressed by the abstract quantifier.

4 Experiments

The main reason for constructing a formalization was to experimentally find out, if background knowledge gained from domain experts is apparent in the data by GUHA means. This part of the paper gives information about experiments: section 4.1 describes experiments' setup, sections 4.2 and 4.3 show two conducted experiments and section 4.4 evaluates the results of the experiments.

4.1 Setup

Modules of the Ferda system were created for purposes of this work and of work [10]. The modules enable the formalization with attributes, validation literals and abstract quantifiers setting. They also automatically find rules from the output of 4FT and KL procedures that match the formalized background knowledge. The details of the implementation, with proper explanation of the modules and description of algorithms can be found in [10].

Special attention was paid to selection of abstract quantifiers. For the KL procedure, we chose variations of Kendall quantifier named *increasing* and *decreasing dependence* for observing positive and negative ordinal dependence. Out of many 4ft-quantifiers, we chose the two most used quantifiers introduced in section 2.1, the founded implication and above average dependence. We presumed that if they are most used, they should be somehow "good".

We chose 8 sample rules constructed by medical experts concerning education and responsibility in work. These rules were selected as a sample of the rules that can be mined upon (without changing the database schema). Rules are listed in Table 2. We used the same common categorization of STULONG attributes both for the task settings and for the formalization settings.

Number	**Rule - left side**	**right side**
1	If education increases	physical activity after work increases as well
2	If education increases	responsibility in work increases as well
3	If education increases	wine consumption increases as well
4	If education increases	smoking decreases
5	If education increases	physical activity in work decreases
6	If education increases	beer consumption decreases
7	Patients with greater responsibility in work	tend to drive to work by car
8	Patients with smaller responsibility in work	tend to use public transport to get to work

Table 2. Verified rules

4.2 Default Quantifiers' Settings

There are threshold values of parameters defined for each quantifier, which tell us when quantifier's output is significant. We call them *default quantifiers' settings*. These values were set up by an agreement among data mining experts. The aim of the first conducted experiment was to verify, if there are in there are any rules verified with aid of formalization and abstract quantifiers defined in previous section backing the background knowledge with default settings.

We chose 0.7 and -0.7 value of the Kendall's coefficient for the increasing and decreasing dependency abstract quantifiers respectively. For the founded implication quantifier, the default values are 0.95 for the p parameter and 0.05 for the (relative) $Base$ parameter. For the above average dependence quantifier, the default values are 1.2 for th p parameter and again 0.05 for the $Base$ parameter.

Table 3 shows the results of the first experiment. The **ID**, **DD**, **FI** and **AA** stands for increasing dependence, decreasing dependence, founded implication and above average dependence quantifiers. **YES** means that the rule was found

Rule number	ID	DD	FI	AA
1	YES	x	NO	NO
2	YES	x	NO	NO
3	NO	x	YES	NO
4	x	NO	NO	NO
5	x	NO	NO	YES
6	x	NO	NO	NO
7	x	x	NO	NO
8	x	x	NO	NO

Table 3. Verification of quantifier's settings

with the given quantifier, **NO** means that the rule was not found and **x** means that the rule was not meaningful for the given quantifier.

Before we draw any conclusions from the experiment, let us first state some presumptions about the data source. The data table *Entry*, which was mined upon, contains records about the entry examination of 1417 patients. Because of this number, we consider the data to be statistically significant. We also presume no errors in the data and proper categorization (described in [10]). Finally, if we want to question settings of individual quantifiers, we presume that the background knowledge rules are "somehow stored" in the data. For example that the number of patients approving the background knowledge rule is greater then the number of patients disapproving the rule.

The most interesting result of the experiment the disapproval of all the rules except one with the founded implication and also the above average quantifier. The fact leads to a conclusion that the p parameters of 4FT quantifiers are too restrictive, e.g. there should be 95% confidence of the rule when using founded implication quantifier.

4.3 Suitable Quantifiers' Settings

As the previous section showed, the default settings of a quantifier can be misleading. The next conducted experiment tries to find suitable quantifiers' settings, based on the background knowledge rule validation. We gradually decreased the p settings of the founded implication and above average dependence quantifiers. We did not experiment with the KL quantifiers, because of the complexity of the problem[6]. With this technique, we could examine more background knowledge rules, determine the value of the parameter for each rule and compute the average of the values for each examined dataset. New mining with the quantifier can be done with this average value and new relevant relationships in the data could be discovered.

As we can see in Table 4, the results of the experiment are rather disappointing for the founded implication quantifier. Majority of rules had the P value

[6] The results need not to improve merely by changing a parameter of a quantifier. We also need to take the shape of the KL contingency table into consideration.

Rule number	FI	AA
1	0.83	1,03
2	0.72	0.43
3	1	0.68
4	0.32	1.17
5	0.28	1.34
6	0.38	1.17
7	0.16	1.15
8	0.64	1.07

Table 4. Exact quantifiers values

below 0.5. We got better results for the above average dependence quantifier where the p parameter was only twice below 1. However, only once the value exceeded the desired 1.2 value.

4.4 Evaluation

Considering the KL procedure, we obtained reasonable results for the increasing dependence and bad results for the decreasing dependence abstract quantifiers. This may be caused by the fact, that the categorial attributes R and C of the task setting contained few categories and thus irregularities of the KL contingency table (see [13] for details) could easily affect the quantifier.

Considering the 4FT procedure, there results for above average dependence were reasonable. On the other hand, the most used quantifier founded implication did not prove to be useful at all. This may be caused by the fact, that for rules no. 4, 5 and 6 (φ increasing, ψ decreasing) founded implication is not a suitable quantifier.

Although the formal theory of the quantifiers (KL and 4FT) is well developed [12], our experiments showed that semantically sound interpretation is yet to be researched. [7] is the first attempt of summarized semantic explanation of significant quantifiers.

5 Related Work

In [2], authors use background knowledge for subgroup discovery. Important part of the work tries to divide background knowledge into classes and deals separately with each class. Unfortunately the rules and formalization defined in this work does not belong to any of the classes defined.

In [4], authors developed ideologically similar approach: they used classical association mining in cooperation with a Bayesian network to store the knowledge from domain experts (here called *a priori expert knowledge*) and improved both the association rules mining and the Bayesian network in iterations. This approach is stronger from the methodological point of view (complex methodology is defined) and also enables revision of the domain expert knowledge.

However, our background knowledge formalization is less restrictive than the Bayesian network and the GUHA procedures offer greater possibilities than the classical association mining.

[9] show another formalization of background knowledge. It is based on qualitative models and used for induction learning. This model is not suitable for GUHA mining, mainly because the strict mathematical requirements of the model.

The data from STULONG itself have been matter of long run research [3, 8]. [14] deals with background knowledge rules annotation into a attribute matrix. This annotation is a simplification of our formalization without proper explanation of suggested abstract quantifiers.

6 Conclusion and Future Work

We focused on evaluation of specific KDD technique – GUHA mining by verifying background knowledge rules on a specific STULONG dataset. Background knowledge consisted of verbal rules created by domain experts. These rules express relationship between two entities of the domain.

In order to automatically verify background knowledge rules, a formalization of them needed to be developed. Our formalization with attributes, validation literals and abstract quantifiers is tailored for the GUHA method, more specifically their 4FT and KL procedures. We also developed automatic tools for the verification of formalized background knowledge rules against the output of the two GUHA procedures.

Experiments were conducted testing background knowledge rules from the STULONG dataset against the data. In the first experiment, we tried to verify background knowledge against default settings of mostly used quantifiers. We found default settings of the quantifiers too strict. Then we continued to find out which values of quantifiers' parameters can verify the background knowledge rules. Results of this experiment were reasonable for the above average dependence quantifier but disappointing for the founded implication quantifier.

The overall output of the experiments is, that more attention should be paid to semantic interpretation of quantifiers and their default settings. This should be the main direction for the future work in the field. New abstract quantifiers need to be defined and conditions for their usage investigated. This requires cooperation between the domain experts and data miners. One of the possible improvements helping to better reflect real life situations is enabling fuzzy quantifiers and attributes.

Acknowledgment

This work was supported by the project MSM6138439910 of the Ministry of Education of the Czech Republic, project IG407056 of University of Economics, Prague and by the project 201/05/0325 of the Czech Science Foundation.

We would like to thanks our research colleagues Jan Rauch and Vojtěch Svátek for help, valuable comments and reviews.

References

1. Agrawal R., Imielinski T., Swami A.: *Mining association rules between sets of items in large databases.* Proc. of the ACM SIGMOD Conference on Management of Data, p. 207 – 216
2. Atzmueller M., Puppe F.: *A Methodological View on Knowledge-Intensive Subgroup Discovery.* In: S. Staab and V. Svátek (Eds.): EKAW 2006, LNAI 4248, Springer-Verlag 2006, p. 318 – 325
3. Berka P., Tomečková M., Rauch J.: *The ECML/PKDD Discovery Challenge on the Atherosclerosis Risk Factors Data.* In: Popelínský L., Krátký M. (ed.): Znalosti 2005. Ostrava, VB TU Ostrava 2005, pp. 18-28. ISBN 80-248-0755-6.
4. Fauré C., Delpart S., Boulicaut J., Mille A.: *Iterative Bayesian Network Implementation by Using Annotated Association Rules.* In: S. Staab and V. Svátek (Eds.): EKAW 2006, LNAI 4248, Springer-Verlag 2006, p. 326 – 333
5. Hájek P., Havránek, T.: *Mechanising Hypothesis Formation - Mathematical Foundations for a General Theory.* Springer-Verlag: Berlin - Heidelberg - New York, 1978.
6. Kováč M., Kuchař T., Kuzmin A., Ralbovský M.: *Ferda, New Visual Environment for Data Mining.* Znalosti 2006, Conference on Data Mining, Hradec Králové 2006,p. 118 – 129 (in Czech)
7. Kupka D.: *User support 4ft-Miner procedure for Data Mining.* Master Thesis, Faculty of Mathematics and Physics, Charles University, Prague 2006 (in Czech)
8. Lucas N., Azé J.. Sebag M.: *Atherosclerosis Risk Identification and Visual Analysis.* In Berka, P. (ed.): Discovery Challenge Workshop Notes. ECML/PKDD-2002. Helsinki 2002.
9. Matwin S., Rouge T.: *Explainable Induction with an Imperfect Qualitative Model.* `http://citeseer.ist.psu.edu/matwin95explainable.html`
10. Ralbovský M.: *Usage of Domain Knowledge for Applications of GUHA Procedures*, Master Thesis, Faculty of Mathematics and Physics, Charles University, Prague 2006 (in Czech)
11. Ralbovský M., Kuchař T.: *Using Disjunctions in Association Mining.* In: Perner P.: Advances in Data Mining, Springer-Verlag 2007, to appear
12. Rauch J.: *Logic of Association Rules.* In: Applied Inteligence, Vol. 22, Issue 1, p. 9 – 28
13. Rauch J., Šimůnek M., Lín V.: *Mining for Patterns Based on Contingency Tables by KL-Miner First Experience.* In: Lin T.Y. (ed.): Foundations and Novel Approaches in Data Mining. Springer-Verlag, pp. 155–167
14. Rauch J., Tomečková M.: *System of Analytical Questions and Reports on Mining in Health Data – a Case Study.* Submitted to IADIS 2007
15. Řehák J., Řeháková B.: *Analysis of Categorized Data in Sociology* (in Czech). Academia: Prague, Czechoslovakia, 1986
16. Svátek V., Rauch J., Ralbovský M.: *Ontology-Enhanced Association Mining.* In: Ackermann, Berendt (eds.). Semantics, Web and Mining, Springer-Verlag, 2006

Dealing with Background Knowledge in the SEWEBAR Project

Jan Rauch, Milan Šimůnek

Faculty of Informatics and Statistics, University of Economics, Prague *
nám W. Churchilla 4, 130 67 Prague, Czech Republic,
rauch@vse.cz, simunek@vse.cz

Abstract. SEWEBAR is a research project the goal of which is to study possibilities of dissemination of analytical reports through Semantic Web. We are interested in analytical reports presenting results of data mining. Each analytical report gives answer to one analytical question. Lot of interesting analytical questions can be answered by GUHA procedures implemented in the LISp-Miner system. The project SEWEBAR deals with these analytical questions. However the process of formulating and answering such analytical questions requires various background knowledge. The paper presents first steps in storing and application of several forms of background knowledge in the SEWEBAR project.

1 Introduction

SEWEBAR (SEmantic WEB and Analytical Reports) [13] is an academic research project developed at Faculty of Informatics and Statistics of University of Economics, Prague. The project is inspired by *10 challenging problems in data mining research* see http://www.cs.uvm.edu/~icdm/. One of them is characterized as *mining complex knowledge from complex data*. Necessity to relate data mining results to the real world decisions they affect is emphasized in relation to this problem. One way how to meet this requirement is to arrange results of data mining into an analytical report structured both according to an analyzed problem and to user's needs. The background knowledge must be taken into consideration when preparing the report. An attempt to produce such analytical report automatically is described in [8], these possibilities are also discussed in [9]. Such analytical reports are natural candidates for Semantic Web [6]. A possible result of development in this direction is outlined in Fig. 1 [13].

There are hospitals storing data concerning patients in their databases. Automatically (or semi-automatically) produced *local analytical reports* give answers to *local analytical questions* concerning patients, their illnesses and treatments in particular hospitals. There are also (semi)automatically produced *global analytical reports*. Each global analytical report summarizes two or more local analytical reports. It is supposed that content of analytical reports is indexed by logical

* The work described here has been supported by the grant 201/05/0325 of the Czech Science Foundation

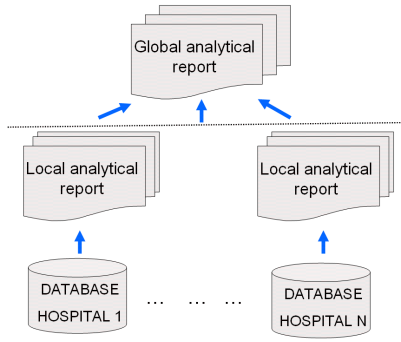


Fig. 1. Sharing analytical reports on Semantic Web

formulas corresponding to patterns resulting from data mining instead of usual key words. The possibility of indexing content of analytical reports by logical formulas is outlined in [9], see also [5, 7].

Various aspects of the SEWEBAR project are discussed in [13]. It is shown that background knowledge related to analyzed data plays important role in the whole project. It can be used to formulate reasonable analytical question, to arrange analytical reports and to apply analytical procedures. It is also shown that the GUHA procedures implemented in the academic software system LISP-Miner [10–12] are suitable to solve lot of reasonable analytical questions. However to make the process of application of background knowledge effective, it is necessary to integrate the operations of storing, maintaining and of application of background knowledge into the architecture of the LISp-Miner system [16]. A new part of the LISp-Miner called *LMKB* (*LISp-Miner Knowledge Base*) was implemented to ensure new requirements related to the SEWEBAR project. This paper presents main features of the *LMKB* and outlines its possibilities.

We use two medical data sets, mentioned in section 2. The LM Knowledge Base *LMKB* is introduced in section 3. Formulation of local analytical questions using knowledge stored in the *LMKB* is sketched in section 4. Application of the GUHA procedure SD4ft-Miner to solve a local analytical question formulated using *LMKB* is described in section 5. Possibilities of application of knowledge stored in the *LMKB* in the SD4ft-Miner procedure are discussed in section 6.

2 Data Sets STULONG and ADAMEK

We deal with two data sets concerning cardiology – STULONG and ADAMEK. We use them as examples of databases of hospitals mentioned in Fig. 1. Data set STULONG concerns *Longitudinal Study of Atherosclerosis Risk Factors*

(`http://euromise.vse.cz/challenge2004/`) [1]. Data set consists of four data matrices, we deal with data matrix *Entry* only. It concerns 1 417 patients – men that have been examined at the beginning of the study. Each row of data matrix describes one patient. Data matrix has 64 columns corresponding to particular attributes – characteristics of patients.

The second data set called ADAMEK concerns preventive cardiology. There is a definition of a set of items to be used to describe cardiologic patients. This set is called *Minimum data model of cardiology patient* [17]. Corresponding methodology is applied in two out-patients departments (Prague and Čáslav) since 2 000, see also `http://www.euromise.org/health/preventive_cardio.html`. The application results in a data matrix concerning 1122 patients (9/2006). Data matrix has 180 columns corresponding to particular attributes.

3 LISp-Miner Knowledge Base

There are various types of background knowledge related to datasets STULONG and ADAMEK and maintained in LISp-Miner Knowledge Base. Important portion of knowledge is given by a structure of a set of attributes. In our case, the attributes can be divided into 26 groups (e.g *Social characteristics*), see Tab. 1. Each pair of these groups has no common attributes and union of all groups is the set of all attributes. We call these groups *basic groups of attributes*. They are defined in *Minimum data model of cardiology patient* [17].

Some of attributes in Tab. 1 are actually meta-attributes. It means that they differ in particular datasets what concerns their possible values. An example is meta-attribute *Education* that has two realizations - attribute *Education* in STULONG with categories (i.e. possible values) *basic*, *apprentice*, *secondary university* and attribute *Education* in ADAMEK with categories *basic*, *secondary*, *higher*. To be consistent we can formally deal also with meta-attribute *Sex* even if its both realizations have the same categories etc. Important knowledge concerning mutual influence of attributes is actually related to meta-attributes, see below.

Three examples of combination of various realizations of particular meta-attributes in both data sets follow, all examples are taken from Tab. 1.

- The meta-attribute is nominal or ordinal, it is realized in both data sets and it has the same categories (i.e. possible values) in both data sets, e.g. *Sex*.
- The meta-attribute is nominal or ordinal, it is realized in both data sets, but the sets of its categories in STULONG and ADAMEK are different, e.g. *Education*.

[1] The study (STULONG) was realized at the 2nd Department of Medicine, 1st Faculty of Medicine of Charles University and Charles University Hospital, U nemocnice 2, Prague 2 (head. Prof. M. Aschermann, MD, SDr, FESC), under the supervision of Prof. F. Boudík, MD, ScD, with collaboration of M. Tomečková, MD, PhD and Ass. Prof. J. Bultas, MD, PhD. The data were transferred to the electronic form by the European Centre of Medical Informatics, Statistics and Epidemiology of Charles University and Academy of Sciences (head. Prof. RNDr. J. Zvárová, DrSc).

attribute	STULONG	ADAMEK
basic group of attributes *Personal information* - 3 attributes total		
Age	natural number	
Sex	*M / F*	
Department	*Prague*	*Prague / Čáslav*
basic group of attributes *Social characteristics* - 4 attributes total		
Marital status	*single / married / widowed / divorced*	
Education	*basic / apprentice / secondary / university*	*basic / secondary / higher*
Lives alone	not used	*yes / no*
Responsibility in a job	*manager / partly independent / pensioner*	not used
group *Physical examination* - 7 attributes total		
Weight (kg)	rational numbers range $\langle 41.6; 187)$	rational numbers range $\langle 52; 133)$
Systolic blood pressure (mm Hg)	natural numbers from the range $\langle 70; 230)$ categories defined as intervals $\langle 70; 80), \ldots, \langle 220; 230)$	
...	...	...

Table 1. Basic groups of attributes for STULONG and ADAMEK data sets

– The meta-attribute is realized in only one of data sets STULONG and ADAMEK, e.g. *Lives alone* and *Responsibility in a job*.

Definition of meta-attributes and attributes together with possible categories is stored and maintained in LISp-Miner Knowledge Base. Information on basic groups of (meta-)attributes as defined in Tab. 1 is very important, it reflects the *Minimum data model of cardiology patient* [17]. The basic groups of (meta-)attributes are perceived by physicians as reasonable sets of attributes. Thus the definition of basic groups of attributes is also stored in LISp-Miner Knowledge Base.

However there are also additional important groups of (meta-)attributes. An example is the group *Cardiovascular risk factors* that contains e.g. attributes *Hypertension*, *Mother hypertension*, *Obesity*, *Smoking* etc. coming from several basic groups. Thus there are tools also for maintaining definitions of additional groups of (meta-)attributes in LISp-Miner Knowledge Base. The information on groups of attributes is used e.g. in formulation of local analytical questions see section 4.

The above mentioned information has close relation to ontologies. There is some experience concerning ontologies in data mining with LISp-Miner [14]. However integration of ontology - like knowledge into LISp-Miner architecture will be very probably more effective than using general ontologies. We can moreover maintain also additional types of knowledge in LISp-Miner Knowledge Base. An example is information on mutual influence of meta-attributes. This knowledge is managed in the way outlined in Fig. 2.

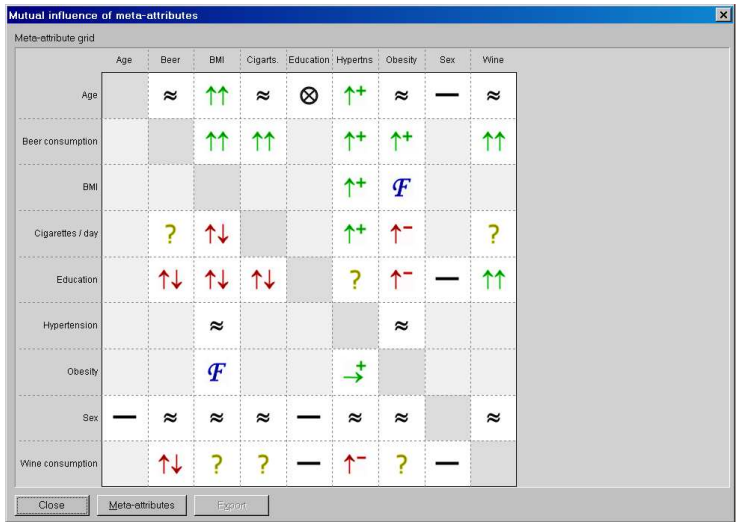


Fig. 2. Mutual influence of meta-attributes

There are four types of attributes: Boolean (e.g. Hypertension), nominal (e.g. *Sex*, *Department*), ordinal (e.g. Education), and cardinal (e.g. BMI i.e. Body Mass Index). There are several types of influences among attributes, most of them are relevant to specified types of attributes. The following types are used in Fig. 2:

- "–" – there is no influence
- $\otimes$ – there is some influence but we are not interested in
- $\approx$ – there is some influence
- $\uparrow\uparrow$ – if the row attribute increases then the column attribute increases too, both attributes are cardinal or ordinal
- $\uparrow\downarrow$ – if the row attribute increases then the column attribute decreases, both attributes are cardinal or ordinal
- $\uparrow^{+}$ – if the row attribute increases then the relative frequency of patients satisfying column attribute increases, row attribute is cardinal or ordinal and column attribute is Boolean
- $\uparrow^{-}$ – if the row attribute increases then the relative frequency of patients satisfying column attribute decreases, row attribute is cardinal or ordinal and column attribute is Boolean
- ? – there could be an influence, no detail is known
- $\mathcal{F}$ – means that there is a strong dependency like function, e.g. *obese* is equivalent to BMI ≥ 32

– $\rightarrow^{+}$ – truthfulness of the row attribute increases then relative frequency of true values of column attribute, both attributes are Boolean.

4 Formulating Local Analytical Questions

There are various local analytical questions that can be generated on the basis of background knowledge [13]. The principle consists in application of a "local question pattern" to items of background knowledge. Examples of *items of background knowledge* are basic or additional groups of attributes or particular cells of Tab. 2 that are not marked by "–". Thus *Education* $\uparrow\downarrow$ *BMI* is an example of item of background knowledge. The signs like $\uparrow\uparrow$ or $\uparrow\downarrow$ are called *type of attributes relation*.

An example of a "local question pattern" (written in `typewriter`) is:
`Which items of background knowledge of the type` *type of attributes relation* `concerning` *attribute* `are observed in` *data set*`?`
An example of a local analytical question generated by this "local question pattern" is the question *Which items of background knowledge of the type* $\uparrow\downarrow$ *concerning attribute* Education *are observed in ADAMEK data?*

An additional "local question pattern" is the pattern :
`What are the differences between particular values of attribute` *attribute* `as for relation of Boolean characteristics of groups of attributes` *group 1 of attributes* `and` *group 2 of attributes* `in` *data set* `?`
It can be written in a symbolical way as

$$\textit{data set}: \quad \bowtie^{?} (\textit{attribute}) : \mathcal{B}[\textit{group 1 of attributes}] \approx \mathcal{B}[\textit{group 2 of attributes}]$$

An example of such a local analytical question is the question

$$\text{ADAMEK:} \quad \bowtie^{?} (\textit{department}) : \mathcal{B}[\textit{Social characteristics}] \approx \mathcal{B}[\textit{Alcohol}]$$

that means *What are the differences between particular departments what concerns relation of Boolean characteristics of social characteristics and of alcohol drinking?* The attribute *department* has only two values - *Prague* and *Čáslav*, see section 2. Thus we can write this analytical question in the form

$$\text{ADAMEK:} \quad \textit{Prague} \bowtie^{?} \textit{Čáslav} : \mathcal{B}[\textit{Social characteristics}] \approx \mathcal{B}[\textit{Alcohol}] \ .$$

This analytical question is solved in the next section.

There are lot of various "local question patterns" that can be applied to particular items of background knowledge stored in the *LISp-Miner Knowledge Base*. It is crucial that the local analytical questions generated this way can be solved by particular analytical procedures implemented in the LISp-Miner system. There are six analytical GUHA procedures in the LISp-Miner system that mines for various patterns verified on the basis of one or two contingency tables, [11, 13]. One of these procedures is the procedure SD4ft-Miner shortly described in the next section.

However the systematic description of all local question patterns is still a research task.

5 Applying SD4ft-Miner

We show how the local analytical question

$$\text{ADAMEK:} \quad \check{C}\acute{a}slav \bowtie^{?} Prague: \ \mathcal{B}[Social\ characteristics] \approx \mathcal{B}[Alcohol]$$

specified in the previous section can be solved by the SD4ft-Miner procedure. We use attributes - social characteristics *Marital status*, *Education* and *Lives alone*, (*Responsibility in a job* is not used in ADAMEK data) see Tab. 1. We also use three attributes describing alcohol consumption: *Beer*, *Wine*, and *Distillates*. The original values were transformed such that these attributes have the following possible values (i.e. categories):
Beer : 0, 1, ..., 7, 8–10, 11–15, > 15 [unit: 0.5 liter/week]
Wine : 0, 1, ..., 10, > 10 [unit: 0.2 liter/week]
Distillates : 0, 0.5, 1, 2, > 2 [unit: 0.05 liter/week].

The procedure *SD4ft-Miner* mines for SD4ft-patterns of the form

$$\alpha \bowtie \beta: \ \varphi \approx \psi \ / \ \gamma \ .$$

Here α, β, γ, φ, and ψ are Boolean attributes derived from the columns of analyzed data matrix $\mathcal{M}$ ($\mathcal{M}$ is ADAMEK in our case). The attributes α and β define two subsets of rows (i.e. subsets of patients in our case). The attribute γ defines a condition, it can be omitted in our case. The attributes φ and ψ are called *antecedent* and *succedent* respectively.

The SD4ft-pattern $\alpha \bowtie \beta : \varphi \approx \psi/\gamma$ means that the subsets of patients given by Boolean attributes α and β differ as for validity of association rule $\varphi \approx \psi$ when the condition given by Boolean attribute γ is satisfied. A measure of difference is defined by the symbol $\approx$ that is called *SD4ft-quantifier*. An example of an SD4ft-pattern is the pattern

$$\check{C}\acute{a}slav \bowtie Prague: \ married \Rightarrow^{D}_{0.228} \ Distillate(0) \wedge Wine(0) \ / \ male \ .$$

It means that the patients from Čáslav differ from the patients from Prague what concerns relation of Boolean attributes *married* and $Distillate(0) \wedge Wine(0)$ (i.e. does not drink neither distillates nor wine) when we consider only male patients.

The difference is given by the *SD4ft-quantifier* $\Rightarrow^{D}_{p}$ with parameter p, in our example it is $p = 0.228$. We introduce it using general notation α, β, γ, φ, and ψ for Boolean attributes derived from the columns of general analyzed data matrix $\mathcal{M}$. The *SD4ft-quantifier* concerns two four-fold contingency tables (i.e. 4ft-tables) $4ft(\varphi, \psi, \mathcal{M}/(\alpha \wedge \gamma))$ and $4ft(\varphi, \psi, \mathcal{M}/(\beta \wedge \gamma))$, see Fig. 3.

The 4ft-table $4ft(\varphi, \psi, \mathcal{M}/(\alpha \wedge \gamma))$ of φ and ψ on $\mathcal{M}/(\alpha \wedge \gamma)$ is the contingency table of φ and ψ on $\mathcal{M}/(\alpha \wedge \gamma)$. The data matrix $\mathcal{M}/(\alpha \wedge \gamma)$ is a data sub-matrix of $\mathcal{M}$ that consists of exactly all rows of $\mathcal{M}$ satisfying $\alpha \wedge \gamma$. It means that $\mathcal{M}/(\alpha \wedge \gamma)$ corresponds to all objects (i.e. rows) from the set defined by α that satisfy the condition γ.

$\mathcal{M}/(\alpha \wedge \gamma)$	ψ	$\neg\psi$
φ	$a_{\alpha\wedge\gamma}$	$b_{\alpha\wedge\gamma}$
$\neg\varphi$	$c_{\alpha\wedge\gamma}$	$d_{\alpha\wedge\gamma}$

$4ft(\varphi, \psi, \mathcal{M}/(\alpha \wedge \gamma))$

$\mathcal{M}/(\beta \wedge \gamma)$	ψ	$\neg\psi$
φ	$a_{\beta\wedge\gamma}$	$b_{\beta\wedge\gamma}$
$\neg\varphi$	$c_{\beta\wedge\gamma}$	$d_{\beta\wedge\gamma}$

$4ft(\varphi, \psi, \mathcal{M}/(\beta \wedge \gamma))$

Fig. 3. 4ft-tables $4ft(\varphi, \psi, \mathcal{M}/(\alpha \wedge \gamma))$ and $4ft(\varphi, \psi, \mathcal{M}/(\beta \wedge \gamma))$

It is $4ft(\varphi, \psi, \mathcal{M}/(\alpha \wedge \gamma)) = \langle a_{\alpha\wedge\gamma}, b_{\alpha\wedge\gamma}, c_{\alpha\wedge\gamma}, d_{\alpha\wedge\gamma}\rangle$ where $a_{\alpha\wedge\gamma}$ is the number of rows of data matrix $\mathcal{M}/(\alpha \wedge \gamma)$ satisfying both φ and ψ, $b_{\alpha\wedge\gamma}$ is the number of rows of $\mathcal{M}/(\alpha \wedge \gamma)$ satisfying φ and not satisfying ψ, etc. The 4ft-table $4ft(\varphi, \psi, \mathcal{M}/(\beta \wedge \gamma))$ of φ and ψ on $\mathcal{M}/(\beta \wedge \gamma)$ is defined analogously.

The *SD4ft-quantifier* $\Rightarrow_p^D$ is related to the condition

$$\left| \frac{a_{\alpha\wedge\gamma}}{a_{\alpha\wedge\gamma} + b_{\alpha\wedge\gamma}} - \frac{a_{\beta\wedge\gamma}}{a_{\beta\wedge\gamma} + b_{\beta\wedge\gamma}} \right| \geq p \ .$$

This condition means that the absolute value of difference between the confidence of the association rule $\varphi \approx \psi$ on data matrix $\mathcal{M}/(\alpha \wedge \gamma))$ and the confidence of this association rule on data matrix $\mathcal{M}/(\beta \wedge \gamma))$ is at least p. The SD4ft-pattern $\alpha \bowtie \beta : \ \varphi \Rightarrow_p^D \psi \ / \ \gamma$ is *true on data matrix* $\mathcal{M}$ if the condition related to $\Rightarrow_p^D$ is satisfied on data matrix $\mathcal{M}$.

The SD-4ft pattern

$$\check{C}\acute{a}slav \bowtie Prague : \ married \Rightarrow_{0.228}^{D} \ Distillate(0) \wedge Wine(0) \ / \ male$$

is verified using the 4ft-tables $\mathcal{T}_{\check{C}\acute{a}slav}$ and $\mathcal{T}_{\text{Prague}}$ see Fig. 4. It is easy to verify that this pattern is true in ADAMEK data. Let us note that the sum

ADAMEK / ($\check{C}\acute{a}slav \wedge$ male)	$Distillate(0) \wedge Wine(0)$	$\neg$ ($Distillate(0) \wedge Wine(0)$)
married	115	115
$\neg$ *married*	23	30

$\mathcal{T}_{\check{C}\acute{a}slav}$ = 4ft(*married*, $Distillate(0) \wedge Wine(0)$, ADAMEK /($\check{C}\acute{a}slav \wedge$ male))

ADAMEK / ($Prague \wedge$ male)	$Distillate(0) \wedge Wine(0)$	$\neg$ ($Distillate(0) \wedge Wine(0)$)
married	34	91
$\neg$ *married*	16	47

$\mathcal{T}_{Prague}$ = 4ft(*married*, $Distillate(0) \wedge Wine(0)$, ADAMEK /($Prague \wedge$ male))

Fig. 4. 4ft-tables $\mathcal{T}_{\check{C}\acute{a}slav}$ and $\mathcal{T}_{\text{Prague}}$

of all frequencies from 4ft-tables $\mathcal{T}_B$ and $\mathcal{T}_C$ is smaller than 1122 because of omitting missing values.

The input of the procedure SD4ft-Miner consists of the SD4ft-quantifier and of several parameters that define the sets of relevant Boolean attributes α, β, γ, φ, and ψ. Their detailed description is out of the scope of this paper, see e.g. [10, 11].

We show only possibilities suitable for our simple example. We do not use the condition γ, thus it is $a_{\alpha\wedge\gamma} = a_\alpha, b_{\alpha\wedge\gamma} = b_\alpha$, etc. There are tens of SD4ft-quantifiers, we use the above defined SD4ft-quantifier $\Rightarrow^D_{0.2}$ (i.e. $\Rightarrow^D_p$ for $p = 0.2$) together with two additional conditions:

$$| \frac{a_\alpha}{a_\alpha + b_\alpha} - \frac{a_\beta}{a_\beta + b_\beta} | \geq \ 0.2 \ \wedge a_\alpha \geq 30 \ \wedge a_\beta \geq 30 \ .$$

The set of relevant Boolean attributes α consists of one Boolean attribute *Department*(*Prague*), similarly for β and *Department*(Čáslav). The expressions *Department*(*Prague*) and *Department*(Čáslav) are examples of *basic Boolean attributes*. The basic Boolean attribute is an expression $A(\omega)$ where $\omega \subset \{a_1, \ldots a_k\}$ and $\{a_1, \ldots a_k\}$ is the set of all possible values of the attribute A.

The basic Boolean attribute $A(\omega)$ is true in row o of data matrix in question if it is $a \in \omega$ where a is the value of A in row o. The subset ω is called a *coefficient* of basic Boolean attribute $A(\alpha)$. The basic Boolean attribute *Department*(*Prague*) means that the patient was examined in Prague.

Let us remember that we are solving the local analytical question ADAMEK: *Čáslav* $\bowtie^?$ *Prague* : $\mathcal{B}$[*Social characteristics*] $\approx \mathcal{B}$[*Alcohol*]. The set $\mathcal{B}$[*Alcohol*] of relevant succedents – Boolean attributes describing alcohol consumption is defined in Fig. 5. In Fig. 5 there is the expression

`Min. length: 1 Max. length: 3 Literals boolean operation type: Conjunction`

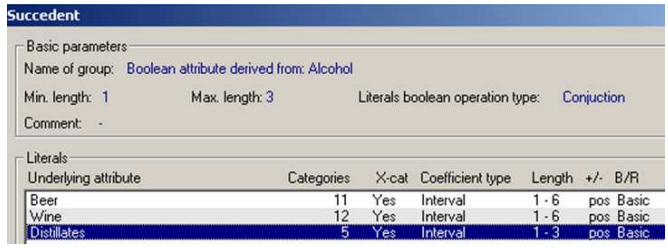


Fig. 5. Definition of the set $\mathcal{B}$[*Alcohol*]

that means that each succedent is a conjunction of 1 – 3 basic Boolean attributes chosen from the sets $\mathcal{B}$[*Beer*], $\mathcal{B}$[*Wine*], and $\mathcal{B}$[*Distillates*], (maximally one attribute is used from each of these sets). In the row of attribute *Beer* in Fig. 5 there is coefficient type defined as `Interval` with `Length 1-6`. It means that coefficients ω of basic Boolean attributes *Beer*(ω) are intervals consisting of 1–6

categories of all 11 categories { 0, 1, . . . , 7, 8–10, 11–15, > 15 } of the attribute *Beer*, see above.

Interval of length 3 consists of 3 *consecutive* categories. *Beer*(0,1,2) and *Beer*(1,2,3) are examples of basic Boolean attributes with coefficients – intervals of length 3. Similarly for additional lengths of intervals. The coefficient of the basic Boolean attribute *Beer*(1,4,9) is not the interval. We write *Beer*(0–2) instead of *Beer*(0,1,2), *Beer*(6–15) instead of *Beer*(6,7,8–10,11–15) etc. It is easy to verify that there are 51 basic Boolean attributes in the set $\mathcal{B}$[*Beer*]. Each of these basic Boolean attributes is of the form *Beer*(ω) where ω is an interval of the length 1–6. Similarly there are 57 basic Boolean attributes in the set $\mathcal{B}$[*Wine*] (12 categories, coefficients specified as `Interval, Length 1-6` and 12 basic Boolean attributes in the set $\mathcal{B}$[*Distillates*] (5 categories, coefficients specified as `Interval, Length 1-3`). It means that there are more than 38 000 basic Boolean attributes in the set $\mathcal{B}$[*Alcohol*].

We define the set $\mathcal{B}$[*Social characteristics*] in a similar way as conjunction of 1–3 basic Boolean attributes. However basic Boolean attributes with coefficients with one category are used together with two additional basic Boolean attributes *Education*(basic,secondary) and *Education*(secondary, higher). It means that there are more than $3 * 10^6$ relevant patterns $\check{C}\acute{a}slav \bowtie Prague : \varphi \Rightarrow^{D}_{0.2} \psi$ such that $\varphi \in \mathcal{B}$[*Social characteristics*] and $\psi \in \mathcal{B}$[*Alcohol*]. The procedure SD4ft-Miner generates and verifies them in 53 sec at PC with 1.6 GHz and 2GB RAM. There are 66 patterns satisfying given conditions, the strongest (i.e. with highest difference of confidences) is the pattern

$$\check{C}\acute{a}slav \bowtie Prague : \ Marital\ status(married) \wedge Education(higher) \Rightarrow^{D}_{0.24} Beer(0) \ .$$

It says that the rule $Marital\ status(married) \wedge Education(higher) \approx Beer(0)$ has confidence 0.24 higher for patients from *Čáslav* than for patients from *Prague*. The corresponding 4ft-tables (concise form) are in Fig. 6.

Čáslav	*Beer*(0)	¬ *Beer*(0)
married ∧ *higher*	39	28
¬(*married* ∧ *higher*)	272	315

Prague	*Beer*(0)	¬ *Beer*(0)
married ∧ *higher*	37	73
¬(*married* ∧ *higher*)	151	208

Fig. 6. 4ft-tables $4ft(married \wedge higher, Beer(0), \check{C}\acute{a}slav\ )$ and $4ft(\varphi, \psi, \mathcal{M}/(\beta \wedge \gamma))$

Let us remark that the procedure SD4ft-Miner does not use the apriori algorithm. Its implementation is based on bit string representation of analyzed data. The same approach is used for additional 5 GUHA procedures implemented in the LISp-Miner system [10, 11].

6 Using Background Knowledge in SD4ft-Miner

The procedure SD4ft-Miner has very fine tools to tune the sets of Boolean attributes to be automatically generated [10, 11], see e.g. definition of the set

$\mathcal{B}[Alcohol]$ in section 5. To deal with these tools effectively, it is necessary to use detailed information on attributes and its values.

Possibilities of definition of set of coefficients (see e.g `Interval Length 1-6` for coefficients of basic Boolean attributes $Beer(\omega)$) must me combined with possibilities of definition of the set of particular categories (there are e.g. 11 categories $\{ 0, 1, \ldots, 7, 8\text{–}10, 11\text{–}15, > 15 \}$ of attribute *Beer*). The application of additional five similar analytical procedures of the LISp-Miner system [11] brings similar problems.

Our goal is to produce the analytical reports automatically. It requires both automatic setting up of input parameters and automatic modification of input parameters of analytical procedures depending on results of mining, see also *EverMiner* project [11]. There are tens of particular relatively small tasks that can be solved automatically, however lot of relevant background knowledge is needed. Important portion of this knowledge is now stored in LISp-Miner Knowledge Base as details concerning particular meta-attributes. In some cases it is related to particular attributes. The detailed description of this knowledge and of its application is out of the scope of this paper, we give only two examples.

The first example concerns recommended categories of particular meta-attributes with rational values. For attributes - instances of meta-attribute *Age* it is suitable to use intervals $\langle 0; 10), \langle 10; 20), \ldots$ or $\langle 0; 5), \langle 5; 10), \ldots$. For attributes - instances of meta-attribute *Systolic blood pressure* it is suitable to use intervals $\langle 70; 80), \langle 80; 90), \ldots$ or $\langle 70; 75), \langle 75; 80), \ldots$. The second possibility is used when the first one gives no interesting results.

The second example concerns definition of set of coefficients that will be automatically generated. One of possible rules is: *If attribute is ordinal then use* `Interval Length 1-F` *where* `F` $= \frac{\text{number of categories}}{3}$. An additional rule is: *If there is no result for* `F` $= \frac{\text{number of categories}}{3}$ *then use* F $= \frac{\text{number of categories}}{2}$.

7 Conclusions

We outlined the project SEWEBAR and we also shown that various forms of background knowledge stored in LISp-Miner Knowledge Base can be used in formulation of local analytical questions. We also demonstrated that one of local analytical questions formulated this way can be solved using GUHA procedure that is a part of LISp-Miner system. We outlined that background knowledge stored in LISp-Miner Knowledge Base can be used to solve these question automatically.

It follows also from additional experience [11–14] that the background knowledge stored in LISp-Miner Knowledge Base can be used both to formulate and solve additional local analytical questions. A way to produce resulting analytical report is outlined in [13], see also [15].

Thus the we will concentrate in our further work to develop software supporting formulation and answering local analytical questions as well as producing of corresponding analytical report. The experience from this process will be used in formulation and answering global analytical reports.

References

1. Mareš R et al (2002) User interfaces of the medical systems - the demonstration of the application for the data collection within the frame of the minimal data model of the cardiological patient. Cor et Vasa, Vol. 44, No. 4 Suppl., pp. 76 (in Czech)
2. Aggraval R et al (1996) Fast Discovery of Association Rules. In Fayyad, U. M. et al.: Advances in Knowledge Discovery and Data Mining. AAAI Press / The MIT Press, pp. 307–328
3. Higgins JPT, Green S, eds. (2006) Cochrane Handbook for Systematic Reviews of Interventions 4.2.6 [updated September 2006]. In: The Cochrane Library, Issue 4, 2006. Chichester, UK: John Wiley & Sons, Ltd.
4. Hájek P, Havránek T (1978) Mechanising Hypothesis Formation - Mathematical Foundations for a General Theory. Springer, Berlin Heidelberg New York
5. Lín V, Rauch J, Svátek, V (2002) Contend-based Retrieval of Analytic Reports. In: Schroeder M, Wagner G(ed.) Rule Markup Languages for Business Rules on the Semantic Web. Sardinia: ISWC, pp. 219-224.
6. Lín V, Rauch J, Svátek, V (2002) Analytic Reports from KDD: Integration into Semantic Web. In: ISWC 2002. Cagliari : University of Cagliari,p. 38.
7. Lín V, Rauch J, Svátek, V (2002) Mining and Querying in Association Rule Discovery. In: Klemettinen, M. et al (ed.).: Knowledge Discovery in Inductive Databases - KDID '02. University of Helsinki, 2002, pp. 97–98.
8. Matheus, J. et al: Selecting and Reporting What is Interesting: The KEFIR Application to Healthcare Data in Fayyad, U. M. et al.: Advances in Knowledge Discovery and Data Mining. AAAI Press / The MIT Press, 1996, 495–515
9. Rauch J (1997) Logical Calculi for Knowledge Discovery in Databases. in Proc. Principles of Data Mining and Knowledge Discovery, Springer-Verlag, pp. 47–57
10. Rauch J, Šimůnek M (2005) An Alternative Approach to Mining Association Rules. In: Lin T Y, Ohsuga S, Liau C J, and Tsumoto S (eds) Data Mining: Foundations, Methods, and Applications, Springer-Verlag, 2005, pp. 219 – 238
11. Rauch J, Šimůnek M: GUHA Method and Granular Computing. In: Hu, X et al (ed.). Proceedings of IEEE conference Granular Computing. 2005, pp. 630–635.
12. Rauch J, Šimůnek Milan, Lín Václav (2005) Mining for Patterns Based on Contingency Tables by KL-Miner First Experience. In: Lin T Y et all (Eds.) Foundations and Novel Approaches in Data Mining. Berlin. Springer-Verlag, pp. 155–167.
13. Rauch J: Project SEWEBAR – Considerations on Semantic Web and Data Mining. To appear in proceedings of 3rd Indian International Conference on Artificial Intelligence (IICAI-07)
14. Svátek V, Rauch J, Ralbovský M (2006) Ontology-Enhanced Association Mining. In: Ackermann et al: Semantics, Web and Mining. Berlin : Springer, pp. 163-179.
15. Strossa P. Černý Z, Rauch J (2005) Reporting Data Mining Results in a Natural Language. In: Lin, T. Y., Ohsuga, S., Liau, C. J., Hu, X. (ed.): Foundations of Data Mining and Knowledge Discovery. Berlin : Springer, pp. 347–362
16. Šimůnek M (2003) Academic KDD Project LISp-Miner. In Abraham A et al (eds) Advances in Soft Computing – Intelligent Systems Design and Applications, Springer, Berlin Heidelberg New York
17. Tomečková M: Minimal data model of the cardiological patient - the selection of data. Cor et Vasa, Vol. 44, 2002, No. 4 Suppl., p. 123 (in Czech)

Using Taxonomic Background Knowledge in Propositionalization and Rule Learning *

Monika Žáková, Filip Železný

Czech Technical University
Technická 2, 16627 Prague 6, Czech Republic
zakovm1@fel.cvut.cz, zelezny@fel.cvut.cz

Abstract. Knowledge representations using semantic web technologies often provide information which translates to explicit term and predicate taxonomies in relational learning. Here we show how to speed up the process of propositionalization of relational data by orders of magnitude, by exploiting such ontologies through a novel refinement operator used in the construction of conjunctive relational features. Moreover, we accelerate the subsequent search conducted by a propositional learning algorithm by providing it with information on feature generality taxonomy, determined from the initial term and predicate taxonomies but also accounting for traditional θ-subsumption between features. This information enables the propositional rule learner to prevent the exploration of useless conjunctions containing a feature together with any of its subsumees and to specialize a rule by replacing a feature by its subsumee. We investigate our approach with a propositionalization algorithm, a deterministic top-down propositional rule learner, and a recently proposed propositional rule learner based on stochastic local search. Experimental results on genomic and engineering data [2] indicate striking runtime improvements of the propositionalization process and the subsequent propositional learning.

1 Introduction

With the development of semantic web technologies and knowledge management using ontologies, increasing amounts of expert knowledge in important knowledge-intensive domains such as bioinformatics is becoming available in the form of ontologies and semantic annotations. One of the well known examples is the Gene Ontology[1]. However, semantic representation is becoming popular even in industrial use [2] for sharing and efficient searching of information in production enterprizes. Knowledge representation formalisms used to capture ontologies and semantic annotations are based on description logics, which have convenient properties with regard to complexity and decidability of reasoning [3].

* An extended version of a paper appearing in the ECML/PKDD'07 proceedings.
[1] http://www.geneontology.org

Inductive logic programming (ILP) aims at learning a theory in a subset of first-order logic from given examples, taking background knowledge into account. It has been considerably successful in various knowledge discovery problems such as in bioinformatics [4]. Standard ILP techniques cannot efficiently exploit explicit taxonomies on concepts and relations, which are typically available in semantic knowledge representations [2]. While in principle, any taxonomy can be encoded in background knowledge, there is good reason to view ontologies as meta-information, which can be directly exploited to guide the refinement operator used to search through the space of first-order rules.

We illustrate this statement through an example. The Gene Function Ontology declares a concept `binding` and its subconcept `protein_binding`. Such concepts are reflected by terms in ILP. It is possible to declare in background knowledge e.g.

```
subclass(binding, protein_binding).
geneFunction(G, F1) :- geneFunction(G, F2), subclassTC(F1, F2).
```

(where `subclassTC/2` is defined as the transitive closure of `subclass/2`). Unfortunately, in such an approach, for the following two exemplary clauses (hypotheses)

$C =$ `activeGene(G):- geneFuction(G, binding).`
$D =$ `activeGene(G):- geneFuction(G, protein_binding).`

it does not hold $C\theta \subseteq D$, so clause D is not obtained by applying a specialization refinement operator onto clause C. Similar reasoning applies to taxonomies on relations (predicates), which are also often present in ontologies (they are a standard part of the Resource Description Framework [5]). Informally, declaring taxonomies only in background knowledge has the consequence that some clause pairs with comparable generality are treated as incomparable by the clause search algorithm. This has significant implications to effiency of search, as we demonstrate in this work.

Recently, effort has been exerted to utilize the information available in ontologies out of the scope of the traditional ILP settings. There are 3 main approaches to this problem: introduce learning mechanisms into description logics [6], [7], hybrid languages integrating Horn logic and description logics [8] and learning in a more expressive formalism [9], [10]. Learning in description logics is useful mainly for the refinement of existing description hierarchies; however, here we are constrained to limitations of description logic and therefore e.g. unable to express formulas with a free variable. Therefore the works investigating learning in description logics are valuable especially in their results on dealing with the open-world assumption in learning and in transformations of description logics into some other subset of the first-order logic. Reasoning and learning in hybrid languages attempts to loosely couple descriptions of concept hierarchies in description logics with rules expressed in function-free Horn logic. Learning in hybrid languages is split into two phases, each utilizing well-known algorithms particular to each of the respective formalisms. Reasoning in hybrid languages is more complex than reasoning in both its constituent formalisms. E.g. even if

reasoning in the chosen subsets of both DL and HL formalisms separately is decidable, reasoning in the corresponding hybrid formalism may be undecidable. A growth in computational complexity is obviously also a deficiency of approaches using representation formalisms with expressivity exceeding that of first-order logic.

This paper concentrates on the problem of exploiting taxonomies in the framework of propositionalization of relational data by constructing relational features and subsequent learning from the propositionalized representation. This approach to relational learning has been paid significant attention in the last few years [11]. In particular, we demonstrate that explicit taxonomies can be exploited with significant runtime benefits at two different stages of this approach.

First, we exploit the term and predicate taxonomies in the process of relational feature construction conducted by the propositionalization algorithm, by adopting the formalism of sorted logic for feature representation and by adapting a refinement operator for first-order features to be co-guided by the taxonomies. The generated feature set plays the role of a boolean attribute set describing the examples from which the propositional algorithm learns.

Second, note that construction of features is implemented through systematic search using the mentioned refinement operator. In the search space, some elements comply to the notion of a feature [1] and they are used as such. Naturally, some of the resulting features are specializations of other resulting features. In contrast to state-of-the-art logic-based propositionalization systems [11], we explicitely store the information that a feature has been obtained by specializing another feature. The propositional algorithm then receives, apart from the propositionalized data matrix, also the resulting feature taxonomy information. This efficiently enables it to prevent the exploration of a conjunction containing a feature together with any of its subsumees and to specialize a rule by replacing a feature by its subsumee.

The first step above assumes that relation and term taxonomies are available beforehand, e.g. from ontology information. In the second step, the feature taxonomy is co-determined by these relation and term taxonomies, but also by standard θ-subsumption among first-order formulas. Consequently, the feature taxonomy is created and can be exploited whether or not relation and term taxonomies were available.

2 Method

In this section we describe our methodological improvements in algorithms for propositionalization and subsequent propositional learning.

2.1 Features

Propositionalization can be understood as a transformation method, where a relational learning problem is compiled to an attribute-value problem, which one can solve using propositional learners [12, 11]. During propositionalization, *features* are constructed from the background knowledge, adressing structural properties of individuals. Each feature is defined as a clause in the form

$$f_i(X) \leftarrow Lit_{i,1}, ..., Lit_{i,n}$$

where the literals in the body are derived from the background knowledge and the argument in clause's head refers to an individual as an example identifier. The features then correspond to attributes which form the basis for columns in single-table (propositional) representations of the data. If the clause defining a feature is called for a particular individual and this call succeeds, the feature is set to "true" in the corresponding column of the given example; otherwise it is set to"false". Recently several propositionalization systems have been proposed. Examples include: RSD [1] and SINUS [13] among others.

Propositionalization systems differ in their constraints applied to select which clauses of the form above form an admissible feature. Our approach stems from the constraints used in the RSD system. While the exact details and their motivations can be found in [1], the general conditions making a conjunction of literals a correct feature body can be briefly stated as follows.

1. Each literal in the conjunction corresponds to a predicate from a declared set of predicates. A predicate declaration also specifies for each argument whether it is an *input* or an *output* argument.
2. Every variable appearing at an input argument in a literal must also appear at an output argument in a preceding literal in the conjunction or in the feature head. Every variable appearing at an output argument in a literal is distinct from all variables in the literal and all preceding literals.
3. The conjunction is not a literal-wise union of two or more conjunctions which themselves are correct feature bodies.
4. The conjunction has at most l literals, for a prescribed $l > 0$.

The third condition prevents the formation of redundant features as it is assumed that the learner employed subsequently on the propositionalized representation (such as a conjunctive rule or decision tree learner) is itself able to conjugate two or more constructed features. To simplify the explanations to follow, we will also assume that all literals $Lit_{i,j}$ are non-negated atoms.

2.2 Sorted Logic

The first difference of the current approach to propositionalization from RSD is in dealing with terms. In RSD, a predicate declaration assigns a type symbol to each argument, from a finite set of type symbols. On top of the constraints enumerated above, a further condition then dictates that in a correct feature body, a single variable may not appear simoultaneously in two arguments with different types.

The present approach replaces the notion of type with that of *sort* borrowed from the formalism of sorted logic, which is suitable for encoding term taxonomies. Sorted logic contains in addition to predicate and function symbols also a disjoint set of sort symbols. A sort symbol denotes a subset of the domain called a sort [14].

A *sorted variable* is a pair, $x : \tau$, where x is a variable name and τ is a sort symbol. Semantically, a sorted variable ranges over only the subset of the domain

denoted by its sort symbol. The semantics of universally-quantified sorted formulas can be defined in terms of their equivalence to ordinary formulas: $\forall x : \tau\phi$ is logically equivalent to $\forall x : \neg\tau(x) \vee \phi'$ where ϕ' is the result of substituting x for all free occurrences of $x : \tau \in \phi$.

A *sort theory* Σ is a finite set of formulas containing function formulas and subsort formulas. A *function formula* has the form

$$\forall x_1, \ldots, x_n \tau_1(x_1) \wedge \ldots \wedge \tau_n(x_n) \rightarrow \tau(f(x_1, \ldots, x_n)) \tag{1}$$

where, in this paper, we constrain ourselves to $n = 0$, thus reducing function formulas to the form $\tau(f)$ reflecting that constant f is of sort τ. A *subsort formula* has the form

$$\forall x \tau_1(x) \rightarrow \tau_2(x) \tag{2}$$

reflecting that τ_1 is a direct subsort of τ_2. It is required that the directed graph corresponding to the subsort theory is acyclic and has a single root denoted *univ*.

For a sort theory Σ, a *Σ-sorted substitution* is a mapping from variables to terms such that for every variable $x : \tau$, it holds that $\Sigma \models \forall x \tau(t)$ where t is $(x : \tau)\theta$. Informally, this is a substitution that does not violate the sort theory.

In the present propositionalization approach, terms in features are constants or sorted variables. Backround knowledge consists of an ordinary first-order theory and a sort theory Σ. A *declaration* for a predicate of symbol π and arity n has the form

$$\pi(m_1\tau_1, \ldots, m_n\tau_n)$$

where $m_i \in \{+, -\}$ denotes whether i-th argument is an input (+) or an output (-), as defined in Sec. 2.1. Besides the constraints stated in Sec. 2.1, correct features must respect the sort relationships. Formally, a literal *Lit* may appear in a feature only if there is a declaration $\pi(m_1\tau_1, \ldots, m_n\tau_n)$ and a Σ-sorted substitution θ such that $\pi(\tau_1, \ldots, \tau_n)\theta = Lit$.

Example. Assume a learning task where examples are genes and background knowledge consists of the gene function ontology and further of gene-gene interaction data. Classification may e.g. distinguish genes expressed in various biological situations. The following predicate declarations may then be stated:

```
geneFunction(+gene, -function).
interacts(+gene, -gene).
```

The background knowledge consists of a first-order theory, such as

```
geneFunction(il2ra, 'interleukin-2 receptor activity').
interacts(cd4, il2ra).
```

and a sort theory, which, for technical convenience, is represented using auxiliary predicates `is_a/2` (for function formulas) and `subsort/2` (for subsort formulas), for example:

```
is_a(il2ra, gene).
subsort('interleukin-2 receptor activity', 'receptor activity').
subsort('receptor activity', function).
```

Then the following exemplary features

```
f0(G:gene) :- interacts(G:gene, H:gene)
f1(G:gene) :- geneFunction(G:gene, F:'receptor activity').
```

are correct (in the second case, `'receptor activity'` is a subsort of `'function'`), and true for gene `il2ra`, whereas the following expression

```
f2(G:gene) :- geneFunction(G:gene, F:function), interacts(G:gene,
H:gene).
```

is not a correct feature, despite its compliance with the sort theory. This is due to condition 3 in Section 2.1; clearly, the body of `f2` is a union of two correct feature bodies. In other words, `f2(X)` is logically equivalent to `f0(X)` $\wedge$ `f1(X)` for any X. Note however, that `f2` can be *specialized by refinement* into a correct feature

```
f3(G:gene) :- geneFunction(G:gene, F:function), interacts(G:gene,
H:gene), geneFunction(H:gene, F:function).
```

expressing that gene `G` interacts with another gene with whom it shares some function. Clearly, `f3` cannot be expressed as a conjunction of any two or more correct features. *(End of example).*

Next we turn attention to the refinement operator through which features are constructed.

2.3 Refinement

We have adapted the *sorted downward refinement* from [14], which accounts for term taxonomies, to further account for the earlier defined feature constraints and predicate declarations used in propositionalization, and for a further kind of taxonomy – the *relation taxonomy* – often available in ontology data. This taxonomy is encoded through meta-predicates in the form

`subrelation(`$pred_1/n$`,` $pred_2/n$`).`

providing the explicit meta-information that goal $pred_1(Arg_1, \ldots, Arg_n)$ succeeds whenever goal $pred_2(Arg_1, \ldots, Arg_n)$ succeeds, i.e. $pred_1$ is more general. The directed graph corresponding to the entire set of the `subrelation/2` statements (where direction is such that edges start in the more general goal) is assumed to be a forest. The set of its roots is exactly the set of predicates declared through the predicate declarations defined in the previous section. It is assumed that the non-root predicates inherit the argument declarations from their respective roots.

As feature heads are fixed in our propositionalization framework, we are concerned with refinement of their bodies, i.e. conjunctions. We will use the notion of an *elementary Σ-substitution*. Its general definition can be found in [14], however, adapted to our framework, the definition simplifies.

An elementary Σ-substitution for a sorted conjunction C is $\{x : \tau_1\} \rightarrow \{x : \tau_2\}$ where $\{x : \tau_1\}$ occurs in C and Σ contains the subsort formula $\forall\psi\tau_2(\psi) \rightarrow \tau_1(\psi)$ for some variable ψ. If $\{x : \tau_2\}$ already occurs in C, then x is deterministically renamed[2] to a variable not occurring in C. Unlike in [14], we can disregard the case of substituting a sorted variable by a function (as we work with function-free features) and, similarly to RSD [1], we neither allow to unify two distinct variables (an equality theory can be defined instead in background knowledge).

Let C be a conjunction, possibly empty, of non-negated atoms where any term is either a constant or a sorted variable, Σ be a sort theory, and Δ a set of predicate declarations. We define the *downward Δ, Σ-refinement*, written $\rho_{\Delta,\Sigma}(C)$, as the smallest set such that:

1. For each θ that is an elementary Σ-substitution for $C, \rho_{\Delta,\Sigma}(C)$ contains $C\theta$.
2. Let $\pi(m_1\tau_1, \ldots, m_n\tau_n)$ be a declaration in Δ such that for each i for which $m_i = +$, C contains a variable (denote it x_i) of sort τ_i' which equals or is a subsort of τ_i. Let further $\{x_i | m_i = -\}$ be a set of distinct variables not appearing in C. Then $\rho_{\Delta,\Sigma}(C)$ contains $C \wedge \pi(x_1 : \upsilon_1, ..., x_n : \upsilon_n)$, where $\upsilon_i = \tau_i'$ if $m_i = +$ and $\upsilon_i = \tau_i$ otherwise.
3. Let C contain a literal $pred_1(x_1\tau_1, \ldots, x_n\tau_n)$ and let $pred_2$ be a direct subrelation of $pred_1$. Then $\rho_{\Delta,\Sigma}(C)$ contains C', which is acquired by replacing $pred_1(x_1\tau_1, \ldots, x_n\tau_n)$ with $pred_2(x_1\tau_1, \ldots, x_n\tau_n)$ in C.

This definition of downward refinement is adapted to our feature construction framework and differs from the refinement operator defined in [14] in several respects, even if no `subrelation` information is assumed. First, we use a simplified notion of the *elementary Σ-substitution* as explained above. Second, our feature bodies are conjunctions of non-negated atoms, and thus the range of our $\rho_{\Delta,\Sigma}(C)$ is reduced in comparison to [14], which refines clauses and therefore also produces negated atoms. Thirdly, the range of $\rho_{\Delta,\Sigma}(C)$ is further reduced because a new atom is added to C only if its input variables can be matched to existing output variables in C, while respecting sort relationships. Lastly, while [14] adds a new atom to C always with all arguments being a variable assigned the root-sort *univ*, we instead 'initiate' each variable with the sort defined for it in the corresponding predicate declaration, which usually are proper subsorts of *univ*. This fact obviously reduces the total volume of search space traversed by the refinement operator.

Confined to 12 pages, we skip the proof that, under very general assumptions on Δ, the defined refinement operator is (i) finite, (ii) complete, in that all correct features (as defined in Sections 2.1 and 2.2) up to variable renaming are enumerated by its recursive closure, whenever the initial C in the recursive application of $\rho_{\Delta,\Sigma}(C)$ is the empty conjunction, and also (iii) non-redundant, in that $\rho_{\Delta,\Sigma}(C_1) \cap \rho_{\Delta,\Sigma}(C_2) = \{\}$ if $C_1 \neq C_2$. However, the operator is not neccessarily correct, in that all its products would be correct feature bodies. In

[2] That is, we do not allow several elementary substitutions differing only in the chosen renaming.

Propositionalize$(\Delta, B, \Sigma, E, l)$: **Given**, a set Δ of predicate declarations, a first-order theory (background knowledge) B, a sort theory Σ, a set of unary ground facts (examples) $E = \{e_1, \ldots, e_m\}$ and a natural number l; **returns** a set $\{f_1, \ldots, f_n\}$ of constructed features, each with at most l atoms in the body, an elementary subsumption matrix **E**, an exclusion matrix **X**, and an attribute-value matrix **A** where $\mathbf{A}_{i,j} = 1$ whenever f_i is true for e_j and $\mathbf{A}_{i,j} = 0$ otherwise.

1. $n = 0$; *Agenda* = a single element list $[(C, 0)]$, where C = empty conjunction;
2. If *Agenda* = []: go to 10
3. $(Curr, Parent) :=$ **Head**$(Agenda)$; $Tail :=$ **Tail**$(Agenda)$
4. If **Nonempty**$(Curr)$ and **Undecomposable**$(Curr)$:
5. $\quad n := n + 1$; $f_n =$ **AddFeatureHead**$(Curr)$;
6. $\quad \mathbf{E}_{n,Parent} = 1$; $Parent = n$;
7. $\quad \mathbf{A}_{n,1\ldots l} =$**Coverage**$(Curr, E, B, \Sigma, \mathbf{A}_{Parent,1\ldots l})$
8. $Rfs := \rho_{\Delta,\Sigma}(Curr)$; $Rfs := \{(Cnj, Parent) | Cnj \in Rfs, |Cnj| \leq l\}$
9. *Agenda* := **Append**(Rfs, Tail); go to 2
10. $\mathbf{X} =$ **Closure**$(\mathbf{E})$
11. **Return** $f_1, \ldots, f_n$, **E**, **X**, **A**

Fig. 1. A skeleton of the algorithm for propositionalization through relational feature construction using the sorted refinement operator $\rho_{\Delta,\Sigma}$.

particular, it may produce a conjunction violating condition 3 in Sec. 2.1, such as the decomposable expression $f2$ shown in Sec. 2.1. However, as we have illustrated, a correct feature body may be obtained by refining a decomposable conjunction. For this reason, we deliberately keep the refinement incorrect in this sense and conduct a further non-decomposability check in the feature construction algorithm (Sec. 2.5).

2.4 Feature Taxonomy

During the recursive application of the refinement operator, a feature generality taxonomy becomes explicit. For purposes of enhancing the performance of the propositional learning algorithm applied subsequently on the propositionalized data, we pass the feature taxonomy information to the learner through two boolean matrices.[3] Assume that features $f_1, \ldots f_n$ have been generated with corresponding conjunctive bodies $b_1, \ldots b_n$. The *elementary subsumption matrix* **E** of n rows and n columns is defined such that $\mathbf{E}_{i,j} = 1$ whenever $b_i \in \rho_{\Delta,\Sigma}(b_j)$ and $\mathbf{E}_{i,j} = 0$ otherwise. The *exclusion matrix* **X** of n rows and n columns is defined such that $\mathbf{X}_{i,j} = 1$ whenever $i = j$ or $b_i \in \rho_{\Delta,\Sigma}(\rho_{\Delta,\Sigma}(\ldots \rho_{\Delta,\Sigma}(b_j)\ldots))$ and $\mathbf{X}_{i,j} = 0$ otherwise. Sec. 2.5 shows how the matrices are instantiated during feature construction and Sec. 2.6 shows how they are utilized by the propositional learner.

2.5 Feature Construction Algorithm

A skeleton of the propositionalization algorithm is shown in Fig. 1. The algorithm is a depth-first search generally similar to the feature constructor of RSD

[3] While alternative data structures are of course possible for this sake, the elected binary matrix form requires modest space for encoding (our implementation uses one byte for each 8 matrix elements) and also is conveniently proccessed in the propositional algorithm implementation.

[1]. The main difference lies in using the novel sorted refinement operator $\rho_{\Delta,\Sigma}$ and also in creating the matrices **E** and **X** storing the generality taxonomy of constructed features. Some of the bold-faced functions have self-explaining names, others require comments. In particular, the **Undecomposable** procedure checks Condition 3 described in Sec. 2.1, through a method used in RSD and detailed in [1]. The **AddFeatureHead** forms a feature clause by formally attaching a head to the body, which consists of the constructed conjunction $Curr$. The **Coverage** procedure verifies the truth value of a conjunction for all examples in E returning a vector of Boolean values. The verification is done by a transformation of the sorted conjunction $Curr$ to an ordinary first-order conjunction as explained in Sec. 2.2 and then using a standard resolution procedure against a Prolog database consisting of B and Σ. Note that for efficiency, only nonempty undecomposable conjunctions, rather than all products of the refinement, are subjected to the coverage check. Again, for efficiency, the procedure obtains the coverage $\mathbf{A}_{Parent,1\ldots l}$ of the closest ancestor (subsuming) conjunction whose coverage was tested: any example i such that $\mathbf{A}_{Parent,i}$ is false can be left out of testing as it makes the current conjunction neccessarily false as well. The **Closure** procedure computes the transitive closure of the elementary subsumption relation captured in **E** in the manner described in Sec. 2.4, and represents the closed relation analogically in matrix **X**, in which it further sets $\mathbf{X}_{i,i} = 1$ for all $1 \leq i \leq n$.

2.6 Rule Learning

We have adapted two rule learning algorithms to account for the feature taxonomy information provided by the propositionalization algorithm.

The first algorithm stems from the rule inducer of RSD [1]. It is based on a heuristic general-to-specific deterministic beam search for the induction of a single propositional conjunctive rule for a given target class, and a cover-set wrapper for the induction of the entire rule set for the class. The core part of the algorithm is a procedure for refinement (specialization) of a propositional conjunction. Given a set of features $F = \{f_1, \ldots f_n\}$, the standard algorithm refines a conjunction C of features into the set $\{C \wedge f_i | f_i \in F, f_i \notin C\}$. In our enhanced version, the algorithm is provided with the elementary subsumption matrix **E** and the exclusion matrix **X**. Using these matrices it can prevent the useless combination of a feature and its subsumee within the conjunction, and specialize a conjunction by replacing a feature with its elementary (direct) subsumee. Formally, the enhanced algorithm refines a conjunction $C = f_{k_1} \wedge \ldots f_{k_p}$ into the set

$$\{C \wedge f_i | f_i \in F, \mathbf{X}_{i,j} = 0 \; \forall f_j \in C\} \cup \tag{3}$$

$$\{f_{k_1} \wedge \ldots f_{k_{r-1}} \wedge f_i \wedge f_{k_{r+1}} \ldots \wedge f_{k_p} | 1 \leq r \leq p, \mathbf{E}_{i,k_r} = 1\} \tag{4}$$

Furthermore, we have similarly enhanced the stochastic local DNF search algorithm first introduced in [15] and later transferred into the propositionalization framework by [16]. This algorithm conducts search in the space of DNF

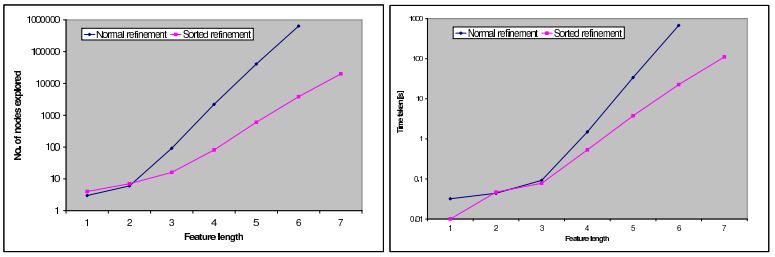


Fig. 2. Sorted refinement vs. standard refinement on CAD data. Left: Nodes explored Right: Time taken. (Experiments exceeding 1000s were discarded)

formulas, i.e. it refines entire propositional rule sets. Refinement is done by local, non-deterministic DNF term changes, which are detailed in [15]. In our version, the **X** matrix is used to prevent the combination of a feature and its subsumee within a DNF term.

3 Experimental Results

We designed experiments to assess the runtime impact of (i) the novel taxonomy-aware refinement operator in the propositionalization process, and (ii) the exploitation of the feature-taxonomy information in subsequent propositional learning.

We conducted tests in two domains. The first concerns genomics, where we used data and language declarations from [17]. The nature of this learning task has been illustrated in the Introduction and in Sec. 2.2. The second is concerned with learning from product design data. Here the examples are semantically annotated CAD documents. We used the same learning setting and ontology data as in [2].

Figures 2 and 3 illustrate on log scale the number of conjunctions searched (left) and the time spent on search (right) to enumerate all conjunctions true for at least 80% examples, for increasing maximum conjunction size l. Here, we distinguish the sorted refinement operator using a special sort theory Σ encoding the taxonomy information, against the standard refinement operator, which treats the taxonomy information only as part of background knowledge. While in both cases exactly the same set of conjunctions is produced, an order-of-magnitude runtime improvement is observed for the 'taxonomy-aware' operator.

Tables 1 and 2 show in turn the runtime spent of inducing a rule set by two algorithms (top-down and stochastic) through 10-fold cross validation in two scenarios: in the first, no feature taxonomy information is used by the algorithms, in the second, feature taxonomy is exploited as described in Sec. 2.6. A significant speedup is observed when feature taxonomy is used without compromising the predictive accuracy.

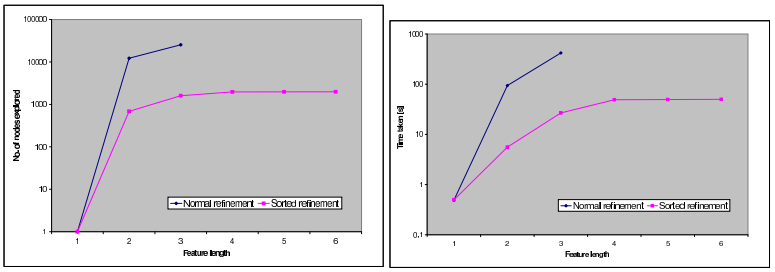


Fig. 3. Sorted refinement vs. standard refinement on Genomic data. Left: Nodes explored Right: Time taken. (Experiments exceeding 1000s were discarded)

Table 1. Propositional rule learning from CAD data

Algorithm	Time taken	Predictive accuracy
Top-down	0.223 ± 0.0785	0.6599 ± 0.2111
Top-down, feature taxonomy	0.061 ± 0.0221	0.6599 ± 0.2214
SLS	0.6268 ± 1.4544	0.6154 ± 0.1770
SLS, feature taxonomy	0.2758 ± 0.83	0.6109 ± 0.1864

4 Conclusions

In this work we have proposed principled methods to exploit term, predicate and feature taxonomies to increase the performance of propositionalization and subsequent propositional learning. The significance of our work is supported by three factors: (i) order-of-magnitude runtime improvements with no sacrifice in predictive accuracy, (ii) the practical value and common use [11] of the propositionalization strategy to relational machine learning, which was the target of our methodological enhancements, and (iii) the increasing volumes of semantic knowledge representations providing explicit taxonomies. In future work, we plan to extend the scope of meta-information exploitable by refinement operators beyond taxonomy information. For example, principled methods are needed to deal with meta-knowledge such as "relation R is a function" or "binary relation R is symmetrical," etc.

Acknowledgements. This research was supported by the FP6-027473 Project SEVENPRO. The authors would like to thank Peter Flach and Simon Rawles for valuable advice concerning term taxonomies in ILP.

References

1. Železný, F. and Lavrač, N.: Propositionalization-based relational subgroup discovery with RSD. Machine Learning 62: 33-63, 2006.

Table 2. Propositional rule learning from Genomic data

Algorithm	Time taken	Predictive accuracy
Top-down search	0.991 ± 0.6548	0.787 ± 0.13
Top-down, feature taxonomy	0.335 ± 0.1927	0.755 ± 0.07
SLS	2.963 ± 2.587	0.787 ± 0.13
SLS, feature taxonomy	1.908 ± 1.689	0.755 ± 0.071

2. Žáková, M., Železný, F., Garcia-Sedano, J. A. et al.: Relational Data Mining Applied to Virtual Engineering of Product Designs. In *Proc. of the 16th Conference on ILP*, Springer, 2007.
3. Baader, F. and Calvanese, D. and McGuinness, D. et al. (Eds.): The Description Logic Handbook: Theory, Implementation and Applications, Cambridge University Press, 2003.
4. King, R. D. et. al.: Functional genomic hypothesis generation and experimentation by a robot scientist.*Nature*, 427:247–252, 2004.
5. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004. Available at http://www.w3.org/TR/rdf-schema/.
6. Badea, L. and Neinhuys-Cheng, S.-W.: A Refinement Operator for Descriptionn Logics. In Cusens, J. and Frich, A. (eds.): Inductive Logic Programming, Vol. 1866 of Lecture Notes in Artificial Intelligence, pp. 40-59, Springer-Verlag, 2000.
7. Kietz, J.-U.: Learnability of Description Logic Programs,Inductive Logic Programming: 12th International Conference, ILP 2002, Sydney, Australia, LNCS Volume 2583/2003, Springer, 2002.
8. Donini, F.M., Lenzerini, M. et. al.: AL-log: Integrating Datalog and Description Logics. Journal of Intelligent Information Systems, 10(3):227 252, 1998.
9. Lloyd, J.W.: Logic for Learning: Learning Comprehensible Theories from Structured Data, Series: Cognitive Technologies, Springer, 2003.
10. Kifer, M. and Lausen, G. and Wu, J.: Logical Foundations of Object-Oriented and Frame-Based Languages, Journal of ACM, vol. 42, 1995, pp. 741-843.
11. Mark-A. Krogel, Simon Rawles, Filip Železný, Peter A. Flach, Nada Lavrač, and Stefan Wrobel. Comparative evaluation of approaches to propositionalization. In *Proc. of the 13th ILP*, volume 2835 of *LNAI*, pages 197–214. Springer, 2003.
12. Nada Lavrač and Peter A. Flach. An extended transformation approach to inductive logic programming. *ACM Trans. on Comp. Logic*, 2(4):458–494, 2001.
13. N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, 1994.
14. Frisch, A.: Sorted downward refinement: Building background knowledge into a refinement operator for ILP. In ILP, LNAI 1634:104-115. Springer, 1999.
15. Ulrich Rückert and Stefan Kramer. Stochastic local search in k-term dnf learning. In *Proc. of the 20th ICML*, pages 648–655, 2003.
16. A. Paes, G. Zaverucha, F. Železný, D. Page, A. Srinivasan. ILP through Propositionalization and k-Term DNF Learning. In *Proc. of the 16th Conference on ILP*, Springer, 2007.
17. I. Trajkovski , F. Železný, J. Tolar, N. Lavrač. Relational Subgroup Discovery for Descriptive Analysis of Microarray Data. In Procs of the 2nd Int Sympos on Computational Life Science, Springer, 2006.

Part II

Web Mining 2.0 Papers

Mining Music Social Networks for Automating Social Music Services*

Claudio Baccigalupo and Enric Plaza

IIIA - Artificial Intelligence Research Institute
CSIC - Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra, Catalonia (Spain)
Vox: +34-93-5809570, Fax: +34-93-5809661
Email: {claudio,enric}@iiia.csic.es

Abstract. Community-driven services compile data provided by the community members, for instance playlists in Web 2.0 music sites. We show how this data can be analysed and knowledge about sequential associations between songs and artists can be discovered. While most of this kind of analysis focus on (symmetric) similarity measures, we intend to discover which songs can "musically follow" others, focusing on the sequential nature of this data in a database of over 500,000 playlists. We obtain a song association model and an artists association model, we evaluate these models comparing the results with other similarity-based analysis, and finally we show how these models can be used to automatically schedule sequences of songs in a social Web radio service.

1 Introduction

In our view, the most interesting data in community-driven services (also called social network Web sites or Web 2.0 applications) are those provided by the community members themselves, the reason being this data would simply be unavailable (or inexistent) if they were not provided by the components of such communities. An example is found in music-related Web communities that allow people to share their personal playlists, which may have been compiled for different purposes (e.g., to listen while jogging, working, partying).

The success of many Web-based communities is also related to the creation of services which target the *social nature* of their public, that is, services that exploit knowledge about people in relation to other people, rather than about individuals. An example of such a service, described in this paper, is that of *Web social radios*. In a Web social radio, a group of people listens at once to the same stream of music, on the same Web radio channel. This service is interestingly different from the so-called Web *personalised* radios, that generate music streams individually tailored for every single person, who will listen in isolation.

* This research is supported in part by a MyStrands scholarship and by MID-CBR project TIN2006-15140-C03-01.

Poolcasting is a Web social radio we have developed, that *customises* the music of each channel for its *group* of listeners, combining a Case-Based Reasoning approach [3] with a knowledge discovery process over a Web 2.0 community. In this paper we focus on the knowledge discovery process needed to construct a model of "musical association" among songs and among artists. The knowledge discovery process has the goal of determining meaningful associations of songs and artists to insure that the sequence of songs streamed into a radio channel is not only customised for the audience group, but also contains a "meaningful" musical sequence, as if it were compiled by a human DJ. This knowledge discovery process has been applied to data the Web 2.0 music community MyStrands has collected from its users. In particular, we have accessed the dataset of playlists the members of MyStrands have shared on the community Web page. By analysing this database of playlists, we were able to find out, for each song, which other songs are more suitable to follow in a good sequence of music.

Several researchers assume that playlist merely contain "similar music"[8]. However, our assumption is not that songs in a playlist are "similar", but that when songs co-occur sequentially in one playlist, they musically flow one after the other, that is they "sound well together" *one after the other* in some specific context (e.g., jogging, working, partying). Although we ignore the implicit "meaning" of a given playlist, we assume that the sequence in which the songs are ordered "makes sense", so that if two songs are contiguously one after the other in a playlist, it somehow "makes sense" to play them in this order (the same assumption is made in [1]). Although this assumption may not hold for *each* playlist, applying this hypothesis to a large dataset of playlists implies that we can extract information about songs that are associated when they co-occur together in many playlists; in particular when they co-occur contiguously and in the same order.

2 User provided playlist data

There are different online sources for playlists. The most relevant ones are Web-based communities dedicated to share playlists: The Art of the Mix, Fiql, GoFish, UpTo11, WebJay, and MyStrands[1]. We work with MyStrands since we have direct access to the user playlists, although these can also be accessed using the Web API OpenStrands, available for developers.

Dynamic nature of playlists. A common thing for Web 2.0 applications is the dynamic nature of user-created content, that can be continuously updated (consider for example *wikis* and *blogs*). Indeed, MyStrands users can publish their personal playlists on the community Web page and later access them to add/remove songs or entire playlists. This fact increases the quality of the data, for new songs are easily entered into playlists, without any hard technical obstacle for users. The larger the number of available playlists, the best the quality of the discovered knowledge.

[1] Their Web pages are respectively: http://artofthemix.org, http://fiql.com, http://gofish.com, http://upto11.net, http://webjay.org, and http://mystrands.com.

Contentless nature of playlists. A playlist on the Web contains only a sequence of "referrers" to some songs, not the actual songs. Unfortunately, there is no such a thing as an universal ID that allows to univocally represent a song in a playlist. Moreover, new songs appear everyday: hence, the problem of correctly identifying *which* songs are included in a playlist. Since we work with MyStrands, we solve this problem by using MyStrands identifiers that are automatically assigned to every song in their catalogue (6 millions and growing).

Social nature of playlists. Playlists are published by the members of the MyStrands community in two different ways: either via the Web page, by manually adding a track after the other to a *current* playlist, or via the MyStrands media player plug-in, that allows to publish playlists directly from the media player (iTunes or Windows Media Player). The dataset of playlists that we have used is a "static snapshot" of all of these user playlists, taken on March 7, 2006, and containing 599,565 playlists.

There are some properties about the MyStrands members, songs and playlists that are worth mentioning: (1) members are 65% male, 35% female, are 32 years old in average (standard deviation: 10 years), and come from a number of countries (United States 41%, Spain 23%, United Kingdom 5%, Canada 3%, Germany 3%, others 25%); (2) the genres of the songs are unevenly distributes since Rock has 58%, and the rest are R&B 7%, Electronic 6%, Latin 5%, Soundtracks 5%, Jazz 3%, Rap 2%, others 14%, and as a consequence we expect to find better results for the most frequent genres; (3) the average length of a playlist is 16.8 songs (standard deviation: 11 songs); very infrequent songs can be found in playlists as well as very popular songs, and this will be reflected in our results.

Noisy nature of playlists. To analyse different playlists from different users and/or different datasets, we need some uniformity mainly about two things: a) the format of a playlist b) the way to identify a track (or artist, or album). Concerning the first point, there exists a quite-standardised format of storing a playlist, via the XML Shareable Playlist Format (XSPF), and more and more communities (including MyStrands) are moving towards this format. Concerning the second point, this is an open issue; every community uses its own IDs to refer to a track, and the same track can sometimes be referred to with different IDs (e.g., a studio version and a live version of the same song). In our case, we basically skipped this problem of track identification by using the IDs that were provided by MyStrands. From their IDs, we had to exclude some indexes that were referring to *virtual* elements; for instance the ID that corresponds to "Various Artists", which cannot be considered as the *same* artist every time it occurs.

3 Discovering Associations by Usage

In this section we will analyse playlist data to discover *song association degrees* and later *artist association degrees*.

Let $s(X, Y) \in [0, 1]$ be the *song association degree* from a song X to a song Y. Counting just the frequency with which two songs appear together

in a collection of playlists is not sufficient to estimate their association degree, for some songs are quite rare, but still are strongly associated with other rare songs. One solution is to consider the association strength from song X to song Y as the conditional probability to find song Y, given a playlist that contains song X, i.e., $P(Y|X) = \frac{f(X,Y)}{f(X)}$, where $f(X)$ is the *popularity* of X (defined as the number of playlists where X appears). Notice that $P(X|Y) \neq P(Y|X)$: the relation is not symmetric. This measure is biased towards having high conditional probabilities with songs that are very popular. That is, $P(Y|X)$ may be high as a result of the fact that Y occurs very frequently and not because X and Y are strongly associated. We correct this problem dividing $P(Y|X)$ by a quantity that depends on the popularity of Y: if Y is very popular (say, more than the average), the association degree is decreased, otherwise it is increased; the exact degree of scaling depends on the playlists and on the distribution of popularity among songs. The following formula takes into account these factors to compute the association between two songs X and Y:

$$\frac{f(X,Y)}{f(X) \cdot (f(Y)/\overline{f})^{\beta}} \tag{1}$$

where $\overline{f}$ is the average song popularity, and β is a parameter that takes a value in $[0, 1]$; when $\beta = 0$, the function is identical to $P(Y|X)$.

We improve this measure by taking into account how far apart two songs are in a playlist, and their relative order. This can be done using a "sliding window" that lists a certain number of consecutive songs in a playlist: if two songs co-occur inside this window, they are considered to be associated, otherwise not. In this way, songs that are common but not specifically associated will not co-occur often relatively to the total number of their occurrences.

We make three assumptions: 1) the farther two songs occur in a playlist, the smaller is their association; 2) if two songs are separated by more than a threshold of $\delta \geqslant 1$ songs in a playlist, their association is null; 3) any song X is more associated to the songs it follows in a playlist than to the songs it precedes. This last point can be explained as follows: since our final goal is to program a radio channel by selecting one song *after* the other, and since the order between songs can be meaningful (e.g., the end of a track mixes into the beginning of the next one), we endeavour to preserve it.

Let $\mathcal{Q}$ be a collection of playlists and $q \in \mathcal{Q}$ be one of these playlists, $q = (X_1, X_2, \dots)$. Let X and Y be two songs; we denote as $d(q, X, Y)$ the distance that separates them in q, e.g., $d(q, X_i, X_j) = j - i$. If either X or Y does not occur in q, $d(q, X, Y) = \infty$. The songs X and Y are associated in q if $d(q, X, Y) \leqslant \delta$; formally we define their *song association degree in q* as:

$$w(q, X, Y) = \begin{cases} 0 & \text{if } |d(q, X, Y)| > \delta \\ 1/|d(q, X, Y)| & \text{if } |d(q, X, Y)| \leqslant \delta \wedge d(q, X, Y) > 0 \\ \alpha/|d(q, X, Y)| & \text{if } |d(q, X, Y)| \leqslant \delta \wedge d(q, X, Y) < 0 \end{cases}$$

where $\alpha \in [0, 1]$ is a parameter to assign higher associations to post-occurrences than to pre-occurrences. Finally, to estimate the *song association degree* between

X and Y, we substitute in Eq. 1 the numerator with $\sum_{q \in \mathcal{Q}} w(p, X, Y)$. That is, rather than accumulating 1 for each playlist q where X and Y co-occur, we accumulate $w(q, X, Y)$, which equals 1 only if Y occurs contiguously after X in q, otherwise $0 \leqslant w(q, X, Y) < 1$:

$$s(X, Y) = \frac{\sum_{q \in \mathcal{Q}} w(q, X, Y)}{f(X)(f(Y)/\overline{f})^{\beta}} \quad . \tag{2}$$

From the MyStrands dataset of playlists, we obtain an average song popularity $\overline{f}$ of 37. We set parameters $\alpha = 0.75$, $\beta = 0.5$, $\delta = 3$ and discard any song that occurs just once, as well as associations within the same artist, for their obviousness. The result is a set of 112,238 distinct songs that have a non-null association with some other song. For instance, the top associated tracks found for *Smoke On The Water* (Deep Purple) are: *Space Truckin'* (VV.AA.), *Cold Metal* (Iggy Pop), *Iron Man* (Black Sabbath), *China Grove* (The Doobie Brothers), *Crossroads* (Eric Clapton), *Sunshine Of Your Love* (Cream), *Wild Thing* (Jimi Hendrix).

We have analysed the same collection of MyStrands playlists to discover knowledge about artist association. Given a playlist $q = (X_1, X_2, \dots)$ and two artists A and B, we denote as $a(X_i)$ the artist of the song X_i, and we denote as $d'(q, A, B)$ the minimum distance that separates a song of A and a song of B in q, e.g., if $a(X_i) = A$ and $a(X_j) = B$, $d'(q, A, B) = j - i$. If q does not contain both a song from A and a song from B, then $d'(q, A, B) = \infty$. We define the *artist association degree in q* from A to B as: $w'(q, A, B) = \frac{1}{|d'(q,A,B)|}$ if $|d'(q, A, B)| \leqslant \delta'$, otherwise $w'(q, A, B) = 0$. Notice that the order is not important when we deal with artists. To estimate the *artist association degree* from any artist A to any artist B $s'(A, B)$ we use an approach similar to the one used for the song association degree using w' instead of w as in Eq. 2:

$$s'(A, B) = \frac{\sum_{q \in \mathcal{Q}} w'(q, A, B)}{f'(A)(f'(B)/\overline{f'})^{\beta}}$$

where $f'(A)$ is the number of playlists where any song by A appears (artist popularity), and $\overline{f'}$ is the average artist popularity. From the MyStrands dataset we obtain an average artist popularity $\overline{f'}$ of 235. Using $\delta' = 2$ as the maximum distance and $\alpha = 0.75$, $\beta = 0.5$, we discover that 25,881 artists have an association with some other artist.

3.1 Parameters

In the previous section, we have introduced some parameters and assigned them some specific values for our experiments and evaluations.

The value $\beta = 0.5$ was decided after several experiments, in order to obtain a nice mix of more and less popular tracks/artists in the associations. For instance, the top associated artists found for Abba when $\beta = 0.5$ are: Agnetha Faltskog, A-Teens, Chic, Gloria Gaynor, The 5th Dimension, Andy Gibb, Olivia Newton-John, Rose Royce, KC & The Sunshine Band, and The Bee Gees. Notice that

the first two names (Agnetha Faltskog and A-Teens) are not very popular, but are very much associated with Abba: the first was their lead singer, the second is a cover band of Abba. As the sequence continues, more popular names appear, still associated with Abba, but in a weaker degree.

The value $\alpha = 0.75$ was decided because we want to favour post-occurrences rather than pre-occurrences, albeit not in excess, for two reasons. The first is that it is *sometimes* useful to maintain the order of two songs (e.g., the first mixes into the second one, the first and the second are two consecutive movements of the same theme, etc.), but for some genres and songs the order is not so important. The other reason is that we assume that songs in user playlists are implicitly ordered, but this is not always the case, for some users build playlists just as unordered sets of songs.

4 Evaluation

In this section we compare the *associations* found using our system with the *degrees of similarity* provided by other community-based music services: All Music Guide, Yahoo! Music, Last.fm, MyStrands and MusicSeer[2].

All Music Guide (AMG) has handcrafted contributions by expert editors, while at Yahoo the recommendations are generated from end user feedback [4]. Last.fm similar artists focus on overall listening habits, based on people's listening history. MyStrands uses our same data, but a different technique[3], while MusicSeer follows two distinct techniques: one is based on a survey about related artists[4], the other is based on a set of playlists collected from the community Art Of The Mix.

Tables 1 and 2 compare our results for two songs with those of Yahoo! Music, the only community that deals with song-to-song similarity. Tables 3 and 4 show a comparison of our artist association model with the "most similar artists" provided by MyStrands, AMG, Last.fm and Yahoo! Music. The four artists compared are Abba, John Williams, Destiny's Child, and Frank Sinatra. When available, the results of MusicSeer are also reported.

Which observations can be done from this examples? First, we point out that All Music Guide can be seen as the base referrer, for its associations are compiled by hands by a group of experts. Nevertheless, we can observe how other automatically-compiled results are not so different from human-compiled ones; for instance our technique, MyStrands, Yahoo! Music and Last.fm, all return Dean Martin as the top result for Frank Sinatra. Also notice that our technique cannot be directly compared with other ones, because we are not looking for "similarity" but for musical association. In our case, for instance, the order of songs in a mined playlist is important, while it is not important in the case of MusicSeer playlists analysis.

[2] Their Web pages are respectively: http://allmusic.com, http://music.yahoo.com, http://last.fm, http://mystrands.com, and http://musicseer.org.

[3] For details: http://blog.recommenders06.com/wp-content/uploads/2006/09/shur2.pdf

[4] For details: http://www.musicseer.org/projects/musicsim/musicseer.org/results/

Table 1. Comparison of associated songs for *Strangers In The Night* (Frank Sinatra).

Poolcasting	*Up, Up and Away* (The 5th Dimension), *Message To Michael* (Dionne Warwick), *Whatever Happens, I Love You* (Morrissey), *Sugar Baby Love* (Rubettes), *Move It On Over* (Ray Charles), *It Serves You Right To Suffer* (John Lee Hooker), *Blue Angel* (Roy Orbison), *I Am What I Am* (Shirley Bassey), *Rain* (Jose Feliciano)
Yahoo! Music	*Mr. Tambourine Man* (The Byrds), *Don't You Want Me* (Human League), *I'm A Believer* (The Monkees), *Good Vibrations* (The Beach Boys), *Stay* (Shakespeare's Sister), *The House Of The Rising Sun* (The Animals), *Oh Pretty Woman* (Roy Orbison), *Working My Way Back To You* (The Spinners), *Never Ever* (All Saints)

Table 2. Comparison of associated songs for *Smoke on the water* (Deep Purple).

Poolcasting	*Space Truckin'* (VV.AA.), *Cold Metal* (Iggy Pop), *Iron Man* (Black Sabbath), *China Grove* (The Doobie Brothers), *Crossroads* (Eric Clapton), *Sunshine Of Your Love* (Cream), *Wild Thing* (Jimi Hendrix), *Song For Jeffrey* (The Rolling Stones), *Money For Nothing* (Dire Straits)
Yahoo! Music	*School's Out* (Alice Cooper), *Slow Ride* (Foghat), *I Can't Drive 55* (Sammy Hagar), *Rock You Like A Hurricane* (Scorpions), *White Room* (Cream), *We're An American Band* (Grand Funk Railroad), *Rock And Roll All Nite* (Kiss), *Purple Haze* (Jimi Hendrix), *All Right Now* (Free)

In general, our technique suffers less than MyStrands from the problem of over-popular artists (e.g., MyStrands returns Green Day for John Williams). Although Soundtrack is not a main genre in the dataset we have used, the associated artists we found for John Williams are closely related (indeed, most of them are other soundtrack music composers). We can observe that our technique often returns the highest association degrees for tracks/artists which are not very popular but are very strictly associated, followed by other artists which are less related but more popular. This is mainly because of the value of β, and is a desirable property for our technique to be applied in contexts where, at each moment, the next song has to be chosen from a restricted set of songs.

Finally, notice that our technique is capable to spot out (as the most associated artist) an artist which indeed is strongly associated with the previous one, although rare. For instance, our technique returns Agnetha Faltskog as the strongest associated artist with Abba, or Kelly Rowland for Destiny's Child. These are good associations, for these two are precisely the lead singers of Abba and Destiny's Child, so the relation holds. Every other music service simply ignores this kind of relationship, presenting a set of "similar artists" which are also known by the public. Our technique, on the other hand, prefers to put first in the list those artists that are *not* so famous to the generic public but that are strongly associated. This is good in the sense that the user can get to discover *new songs and artists*, not just new relations of similarity between songs/artists.

5 Social Web radio with Poolcasting

The advantage of considering *ordered sequences* of songs in our mining technique is useful when the association degrees that we discover are applied to generate good *sequences* of songs, in different contexts. Poolcasting is a social Web radio

Table 3. Comparison of associated artists for Abba.

Poolcasting	Agnetha Faltskog, A-Teens, Chic, Gloria Gaynor, The 5th Dimension, Andy Gibb, Olivia Newton-John, Rose Royce, KC & The Sunshine Band, The Bee Gees
MyStrands	Donna Summer, Madonna, Gloria Gaynor, Cyndi Lauper, Blondie, Kool & The Gang, Elton John, The B-52 s, Michael Jackson, Diana Ross
AMG	Ace of Base, Gemini, Maywood, Bananarama, Lisa Stansfield, Gary Wright, Roxette, Animotion, Clout
Yahoo! Music	The Bee Gees, The Carpenters, The Beatles, Foreigner, Whitney Houston, Madonna, Michael Jackson, Elton John, Cher, Chicago
Last.fm	The Bee Gees, Madonna, Cher, Kylie Minogue, Boney M., Michael Jackson, Gloria Gaynor, Village People, Donna Summer, Britney Spears
MusicSeer Survey	Ace of Base, Bee Gees, Blondie, Spice Girls, Olivia Newton-John, Beach Boys, Roxette, Cyndi Lauper, Backstreet Boys, Donna Summer
MusicSeer Playlists	Bee Gees, Blondie, Cyndi Lauper, Queen, Cat Stevens, Cher, Beach Boys, Donna Summer, Olivia Newton-John, Phil Collins

Table 4. Comparison of associated artists for John Williams.

Poolcasting	Williams & London Symphony Orchestra, Meco, Danny Elfman, Williams & Boston Pops, John Carpenter, London Theatre Orchestra, John Barry, Hollywood Studio Orchestra, Elmer Bernstein/RPO Pops, Spectrum
MyStrands	Danny Elfman, Vangelis, Hollywood Studio Orchestra, Erich Kunzel, Green Day, Gorillaz, Weird Al Yankovic, John Barry, Queen, Eminem
AMG	John Barry, Jerry Goldsmith, Elmer Bernstein, Howard Shore, Erich Korngold
Yahoo! Music	Patrick Doyle, James Horner, James Galway, Danny Elfman, Howard Shore, Hans Zimmer, London Symphony Orchestra, Enya, Frank Sinatra, Josh Groban
Last.fm	Howard Shore, Hans Zimmer, James Horner, Danny Elfman, Klaus Badelt, Harry Gregson-Williams, Jerry Goldsmith, Alan Silvestri, Patrick Doyle, Ennio Morricone

Table 5. Comparison of associated artists for Destiny's Child.

Poolcasting	Kelly Rowland, City High, Ciara, Fantasia, Christina Milian, Beyonce, Ashanti, Girls Aloud, 3LW, Dru Hill
MyStrands	Ciara, Pussycat Dolls, Usher, Beyonce, Nelly, 50 Cent, Mariah Carey, Chris Brown, Gwen Stefani, Eminem
AMG	Mariah Carey, Jennifer Lopez, Aaliyah, Xscape, Ginuwine, Deborah Cox, Kelly Price, Faith Evans, Brandy, Usher
Yahoo! Music	Cruel Story Of Youth, Jessica Simpson, Ryan Cabrera, Ashlee Simpson, Faith Evans, Nick Lachey, Vitaly Romanov, Janet Jackson
Last.fm	Beyoncé, Mariah Carey, Jennifer Lopez, Usher, Aaliyah, Rihanna, TLC, Ciara, Ashanti, Christina Aguilera

Table 6. Comparison of associated artists for Frank Sinatra.

Poolcasting	Dean Martin, Sammy Davis Jr., Judy Garland, Bing Crosby, The California Raisins, Tony Bennett, Louis Prima, Rosemary Clooney, Nat "King" Cole, Ella Fitzgerald
MyStrands	Dean Martin, Billie Holiday, Nat "King" Cole, Perry Como, Ella Fitzgerald, Andy Williams, Tony Bennett, Etta James, Bing Crosby, Diana Krall
AMG	Dean Martin, Vic Damone, Dick Haymes, Sarah Vaughan, Nat King Cole, Dinah Washington, Mel Tormé, Ella Fitzgerald, Tony Bennett, Jo Stafford
Yahoo! Music	Dean Martin, Tony Bennett, Bing Crosby, Nat King Cole, Elvis Presley, The Beatles, Norah Jones, Ella Fitzgerald, Louis Armstrong, Michael Bublé
Last.fm	Dean Martin, Louis Armstrong, Nat King Cole, Bing Crosby, Ella Fitzgerald, Tony Bennett, Bobby Darin, Michael Bublé, Billie Holiday, Sammy Davis, Jr.
MusicSeer Survey	Eric Clapton, Billy Joel, Elton John, Elvis Costello, Elvis Presley, Van Morrison, John Lennon, Bob Dylan, Nine Days, Ozzy Osbourne
MusicSeer Playlists	Elvis Presley, Elton John, John Denver, Abba, Whiskeytown, Beatles, Billy Joel, Bob Marley, Eric Clapton, Everly Brothers

service that automatically generates the content of radio channels using the *music pool* of the users connected to the service. The *music pool* is the collection of all songs the users have in their music players (e.g. iTunes); when a song is selected for a channel, it is uploaded to the Poolcasting server and streamed back to that channel listeners. A channel is defined by a set of conditions (e.g. genre = 'Rock' and year > 1990) and a set of listening users.

In addition to the song and artist associations models that we described in the previous sections, Poolcasting takes into account the listening habits of the users, analysing the songs in their personal libraries: which tracks/artists are contained, how they were rated, how many times they were played. The scheduling of songs in a channel is determined by a Case-Based Reasoning system (CBR) that uses the association models we described as *background knowledge*, and considers each user library and its listening habits data as an *individual case base*. The goal here is not to personalise a radio for an individual person, but to customise each channel dynamically for the actual *group of listeners* at each moment in time. The CBR system is described in [3].

Poolcasting has to select one of the songs available in the music pool at each moment; for this purpose each song is evaluated combining both song association degree and artist association degree. In this combination, song association degree has more importance, but if no song-associated song can be found, Poolcasting looks for artist-associated choices. In brief, we assume that, given the last song Y played on a channel, the following songs are related and can be played on the same channel after Y (with decreasing relevance) using four layers of decision:

1. songs Z that have a strong song association degree $s(Y, Z)$ with Y;
2. songs Z that have a strong song association degree $u(Y, Z)$ with songs from the artist of Y (where $u(Y, Z)$ is the average song association degree from every song whose artist is $a(Y)$ to Z);
3. songs Z that have a strong song association degree $v(Y, Z)$ with songs from artist associated with the artist of Y (where $v(Y, Z)$ is the average song association degree from every song whose artist is associated with $a(Y)$ to Z, combined with the relative artist association degree);
4. songs Z whose artist has a strong artist association degree with the artist of Y (using artist association $s'(a(Y), a(Z))$).

Figure 1 shows an example, where a song has to be played on a channel after Abba's *Waterloo*: if a song Z can be selected using $s(Y, Z)$ at the first level, Z will be the next played song; if none can be found in the music pool, $u(Y, Z)$ is used at the second level to find Z; if not, the system proceeds to layers three and four with $v(Y, Z)$ and $s'(a(Y), a(Z))$. This intuition is implemented as the following aggregation function $r(Y, Z) = s(Y, Z) + \epsilon u(Y, Z) + \epsilon^2 v(Y, Z) + \epsilon^3 s'(a(Y), a(Z))$, where ϵ may vary in $[0, 1]$ (currently $\epsilon = 0.5$). In this way every song Z in the music pool can be assigned a relevance value $r(Y, Z)$ that represents how good it would be to schedule Z after Y.

Table 7 shows an example of the combination of these four factors to evaluate which is the best song to play on a channel after Abba's *Super Trouper*, when the music pool is made of the songs *Take On Me* (A-Ha), *Listen To Your Heart*

Fig. 1. Four layered factors to select next song after Abba's *Waterloo*.

(Roxette), *The Look Of Love* (ABC), and *I'm So Excited* (The Pointer Sisters).

In Poolcasting, the selection of which song to schedule on a channel in a particular moment is taken in two consecutive steps. First, $r(Y, Z)$ is used to build a list of possible candidates. Then, the music preferences of those users that are currently listening to the channel are analysed and combined, to pick the candidate that most satisfy the listeners [3]. This combination gives more weight to those users who were less satisfied with the last tracks selected to be played on that channel.

The musical preferences of each listener are inferred by Poolcasting both implicitly and explicitly. Implicitly, because Poolcasting analyses the personal music library of each participant, and infers that the higher the rating assigned to a song and the higher the play count, the stronger the preference of a user for that song. Explicitly, because with the Web interface users can evaluate the proposed songs, thus stating how much they like or dislike a specific song.

Table 7. Combining factors to find which song to schedule after Abba's *Super Trouper*.

Z_i $a(Z_i)$	$f(Z_i)$ $f(a(Z_i))$	$s(Y, Z_i)$	$u(Y, Z_i)$	$v(Y, Z_i)$	$s'(a(Y),$ $a(Z_i))$	$r(Y, Z_i)$	rank
Take On Me (A-Ha)	1341 1937	$\frac{0.942}{10^3}$	$\frac{0.574}{10^3}$	$\frac{0.324}{10^3}$	$\frac{2.817}{10^3}$	$\frac{1.662}{10^3}$	2º
Listen To Your Heart (Roxette)	184 642	$\frac{2.548}{10^3}$	$\frac{0.841}{10^3}$	$\frac{1.119}{10^3}$	$\frac{0.265}{10^3}$	$\frac{3.281}{10^3}$	1º
The Look Of Love (ABC)	237 878	0	$\frac{1.807}{10^3}$	$\frac{0.852}{10^3}$	$\frac{0.944}{10^3}$	$\frac{1.234}{10^3}$	3º
TI'm So Excited (The Pointer Sisters)	278 1149	0	$\frac{1.063}{10^3}$	$\frac{0.114}{10^3}$	$\frac{0.428}{10^3}$	$\frac{0.614}{10^3}$	4º

6 Related Work

MusicSeer endeavours to extract artist associations from user playlists. As described in [8]: "we gathered over 29,000 playlists from The Art of the Mix, a website that serves as a repository and community center for playlist hobbyists (www.artofthemix.org). After filtering for our set of 400 artists, we were left with some 23,000 lists with an average of 4.4 entries". Notice the difference in the size of experimental data between their approach and our approach.

Playlists are not the only musical data mined from the Web that has been used to discover song/artist relations. For instance [13] and [14] consider that when the *names* of two artists co-occur together in many Web pages, then they are related.

Another text-based approach is [10], that retrieves playlists from the Web in form of radio programs or users' music compilations, then tries to identify titles extracting text, in order to apply a co-occurrence analysis to assess similarity between artists. Our analysis of the occurrences of songs in playlists is different, in that we take into account the ordering of the sequences. For this reason our approach is more related to the analysis of the occurrences of words in phrases [7, 2]. In particular, we deal with sequence of songs, where the order is relevant: co-occurrences [9], and in particular post-occurrences [12].

7 Conclusions and Future Work

We have shown how knowledge discovered from a Web-based music community (song and artist association models) can be used in conjunction with individual user data to provide a customised service in the form of several automated radio channels. Poolcasting is thus an example of using data and knowledge provided from social networking for a service (automated radio channels) that is also social in nature. In this paper, we have focused on the knowledge discovery process that acquires the association models conducive to determine affinities of songs that are to be played in sequence. Poolcasting is currently being tested in our local network. We plan to deploy it to the Internet; actually this would require us to pay the copyright fees normally applied to commercial Web radios.

Our focus on sequential ordering information could be used, as we plan to do in future work, for other tasks in addition to Poolcasting. A straightforward application is a media player plug-in that generates playlists for a single user, considering her library of songs as the only music pool from which songs have to be selected. Another application is generating "channels for parties", as done in PartyStrands[5]; in this context the association models would work the same way, but we would need to change the way in which data from individual profiles are acquired.

In this paper we have shown how to calculate associations from *one* song to another; this approach could be expanded to look for contiguous sequential

[5] http://partystrands.com

patterns: sequences of two or more songs that appear frequently in the dataset of playlists and as such can be considered strongly associated.

References

1. Andric A. & Haus G. (2006). Automatic playlist generation based on tracking user's listening habits. *Multimedia Tools and Applications*, vol. 29(2), pp. 127–151.
2. Ahonen-Myka H. (1999). Finding All Frequent Maximal Sequences in Text. In *ICML-99 Workshop: Machine Learning in Text Data Analysis*, Mladenic D. & Grobelnik M. (eds.), pp. 11-17.
3. Baccigalupo C. & Plaza, E. (2007). A Case-Based Song Scheduler for Group Customised Radio. In *Proc. of the 7th Int. Conf. on Case-Based Reasoning (ICCBR '07)*, Weber R.O. & Richter M.M. (eds.), LNAI 4626, pp. 433–448.
4. Baumann S. & Hummel O. (2003). Using Cultural Metadata for Artist Recommendations. In *Proceedings of the Int. Conf. on WEB Delivering of Music (WEDELMUSIC '03)*, pp. 138–141.
5. Brown B., Sellen A. & Geelhoed G. (2001). Music Sharing as a computer supported collaborative application. In *Proc. of the 2001 European Conf. on Computer Supported Cooperative Work (ECSCW '01)*, pp. 179–198.
6. Ellis D.P.W., Whitman B., Berenzweig A. & Lawrence S. (2002). The Quest for Ground Truth in Musical Artist Similarity. In *Proceedings of the 3rd Int. Symposium on Music Information Retrieval (ISMIR '02)*, pp. 13–17.
7. Heylighen F. (2001). Mining Associative Meanings from the Web: from Word Disambiguation to the Global Brain. In *Proceedings of the International Colloquium: Trends in Special Language & Language Technology*, Temmerman R. & Lutjeharms M. (eds.), pp. 15–44.
8. Logan B., Ellis D.P.W. & Berenzweig A. (2003). Toward Evaluation Techniques for Music Similarity. In *Proceedings of the 4th Int. Symposium on Music Information Retrieval (ISMIR '03)*, pp. 81–85.
9. Mannila H., Toivonen H. & Verkamo A.I. (1995). Discovery of Frequent Episodes in Event Sequences. *Data Mining and Knowledge Discovery*, vol. 1(3), pp. 259-289.
10. Pachet F., Westermann G. & Laigre D. (2001). Musical Data Mining for Electronic Music Distribution. In *Proceedings of the Int. Conf. on WEB Delivering of Music (WEDELMUSIC '01)*, pp. 101–106.
11. Schedl M., Knees P. & Widmer G. (2005) Discovering and Visualizing Prototypical Artists by Web-based Co-occurrence Analysis. In *Proceedings of the 6th Int. Symposium on Music Information Retrieval (ISMIR '05)*, pp. 21–28.
12. Sudkamp T. (2004). Co-occurrence, interest, and fuzzy events. In *Fuzzy Information, 2004. Processing NAFIPS*, vol.2, pp. 508–513.
13. Whitman B. & Lawrence S. (2002). Inferring Descriptions and Similarity for Music from Community Metadata. In *Proceedings of the 2002 Int. Computer Music Conf. (ICMC '02)*, pp. 591–598.
14. Zadel M. & Fujinaga I. (2004). Web Services for Music Information Retrieval. In *Proceedings of the 5th Int. Symposium on Music Information Retrieval (ISMIR '04)*, pp. 478-483.

Dynamic Item Weighting and Selection for Collaborative Filtering

Linas Baltrunas and Francesco Ricci

Free University of Bozen-Bolzano
Domeninkanerplatz 3, Bozen, Italy
{lbaltrunas,fricci}@unibz.it

Abstract. User-to-user correlation is a fundamental component of Collaborative Filtering (CF) recommender systems. In user-to-user correlation the importance assigned to each single item rating can be adapted by using item dependent weights. In CF, the item ratings used to make a prediction play the role of features in classical instance-based learning. This paper focuses on item weighting and item selection methods aimed at improving the recommendation accuracy by tuning the user-to-user correlation metric. In fact, item selection is a complex problem in CF, as standard feature selection methods cannot be applied. The huge amount of features/items and the extreme sparsity of data make common feature selection techniques not effective for CF systems. In this paper we introduce methods aimed at overcoming these problems. The proposed methods are based on the idea of dynamically selecting the highest weighted items, which appear in the user profiles of the active and neighbor users, and to use only them in the rating prediction. We have compared these methods using a range of error measures and we show that the proposed dynamic item selection performs better than standard item weighting and can significantly improve the recommendation accuracy.

1 Introduction

The World Wide Web, interconnecting a myriad of information and business services, has made available to on-line users an over abundance of information and very large product catalogues. Hence, users trying to decide what information to consult or what products to select may be overwhelmed by the number of options that they can potentially access. Collaborative Filtering (CF) is a recommendation technique which emulates a simple and effective social strategy called "word-of-mouth" and is now largely applied in Web 2.0 platforms. In CF personalized recommendations for a target user are generated using opinions (item ratings) of users having similar tastes to that of the target user [1]. A CF system represents users with their ratings on a set of items (ratings vectors). When requested to generate a recommendation for a target user, a CF system first selects a set of similar users according to a similarity/correlation measure computed on their ratings vectors. Then, it generates rating predictions for items not rated yet by the target user. Finally the system recommends the items with the highest predicted rating.

Neighborhood formation is an important step of CF [1], and this paper concentrates on feature weighting and feature selection methods, aimed at improving the recommendation accuracy by tuning the user-to-user similarity metric. In fact, CF can be seen as an instance-based learning approach where users are instances described by their feature vectors, where the product ratings play the role of features. Hence, the rationale for the application of feature weighting/selection techniques in CF is that some items may be more important than others in the similarity computation. This paper reviews and compares feature weighting methods. Moreover, we introduce three dynamic feature selection methods that are suited to the specific characteristics of data used by CF prediction.

Usually, in standard classification and regression learning, feature selection is preferred to feature weighting. In fact, feature selection decreases dimensionality of data, speeds up the learning process and could improve model interpretability [5]. But, the application of feature selection to CF rating prediction and recommendation faces two major problems. Firstly, real world data exploited in CF systems contains a huge amount of features, and secondly it is extremely sparse. In this paper we show how both problems can be tackled exploiting a rather simple dynamic feature selection approach based on feature weights computed by a number of alternative methods. In practice, features with largest weights are selected on-line, every time a rating prediction is required, depending on the target user, target item, i.e., the item whose rating prediction is sought, and the neighbor user whose similarity is needed. Our experiments show that dynamic feature selection is more effective than feature weighting.

The paper is organized as follows. Section 2 overviews some related work. Methods used to improve user-to-user similarity are discussed in section 3. Section 4 describes the feature weighting methods we used in our study, and Section5 describes how dynamic feature selection methods can be exploited in a CF. All the proposed methods are evaluated in Section 6, and Section 7 draws the conclusions and presents future work.

2 Related Work

Feature weighting is a well studied problem in Machine Learning, and Instance-Based Learning in particular. A number of studies reported accuracy improvements when features are weighted in the instance-to-instance distance computation (see [9] for a survey). In the context of CF, feature weighting raises some notable problems. The huge amount of features and data sparsity makes most methods inefficient. Breese et al. [2] adapted the idea of inverse document frequency [7] for feature weighting in CF. The key idea of that approach, called Inverse user Frequency, is that universally liked and known items do not give a lot of information about the true user preferences, therefore, the weights for such commonly rated items should be decreased. The approach showed better performances than the unweighted version of the method. The same idea of decreasing the weights of commonly liked items was implemented by using variance weighting [3]. Here, items with bigger variance are supposed to better distinguish

users' tastes, therefore, they receive larger weights (in user-to-user similarity). But, the authors report that this method slightly decreases the Mean Absolute Error (MAE) with respect to the non-weighted version.

In [10] Yu et al. introduced Information Theoretical approaches for feature weighting. They showed that Mutual Information and Entropy based methods perform better than Inverse User Frequency and the baseline CF. Moreover, Mutual Information gains 4.5% accuracy improvement and performs better (even when trained on a small number of instances) than the baseline CF. They also report that Inverse User Frequency, differently from an earlier report, decreases the accuracy. Since these papers are contradicting we decided to test ourselves the performance of some of these methods. [4] presents another automatic weighting schema for CF systems. This method tries to find weights for different items that bring each user closer to the similar users and further from dissimilar users. The method uses ideas from model-based approaches and obtains a reduction of MAE.

Feature selection algorithms were studied in Machine Learning for several decades [9]. Kohavi or Langley However, the huge search space of thousands and millions of items, and the fact items/features must be selected depending on the target item prediction, makes them hard to apply to CF. For these reason feature selection has not been applied to CF.

3 Improving the Similarity

The user-to-user similarity is the core computation step in user-based CF [1]. Similarity is used in the neighborhood formation and in the final rating prediction. The two most popular measures of user-to-user similarity are: the cosine of the angle, formed by the rating vector of the two users; and the Pearson Correlation Coefficient (PCC). PCC is preferred when data contains only positive ratings, and has been shown to produce better results in such cases [2]. PCC among users x and y is defined as:

$$PCC(x,y) = \frac{\sum_i (v_{xi} - \bar{v_x})(v_{yi} - \bar{v_y})}{\sqrt{\sum_i (v_{xi} - \bar{v_x})^2 \sum_i (v_{yi} - \bar{v_y})^2}}$$

where v_{xi} denote the rating given by user x to item i, and $\bar{v_x}$ is the mean of the ratings of user x. The sum runs over all the items i that both users rated.

Feature weighting and feature selection are two methods that have been largely used in instance-based learning approaches to improve the prediction accuracy of classifiers [9]. In fact, a user-based CF system can be described as a collection of instance-based classifiers, one for each item whose rate is to be predicted. Given a target item (class) and a user whose rating must be predicted, the user's ratings on all the other items provide the instance description (predictive features or items). In this perspective, the rating prediction step of a CF system can be described as a classification or regression learning problem, i.e., one classification/regression problem for each item's rating prediction. The similarity measures we introduced above are based on user preferences, i.e. item

ratings. Hence, these items can be regarded as user's features, and the ratings are the feature values. In the rest of this paper we shall call features of the user the items managed by the CF system, and feature values the ratings assigned by the user to the corresponding item.

Feature weighting methods assign to each instance's feature a weight that measures how important is the feature in the overall instance-to-instance similarity. This translates in the CF case to weights assigned to items used to balance the importance of predictive items in the user-to-user similarity.

Feature selection methods operate in a different way. Instead of providing a weight for each feature, they decide whether a feature must be used or not in the similarity computation. In fact, feature selection could be seen as a particular case of feature weighting, where feature selection uses binary valued weights. We have applied both approaches to a weighted version of the PCC user-to-user similarity measure:

$$WPCC(x,y,j) = \frac{\sum_i (w_{ji}(v_{xi} - \bar{v_x}))(w_{ji}(v_{yi} - \bar{v_y}))}{\sqrt{\sum_i (w_{ji}(v_{xi} - \bar{v_x}))^2 \sum_i (w_{ji}(v_{yi} - \bar{v_y}))^2}}$$

where j denotes the target item of the rating prediction, and w_{ji} is the weight of the (predictive) item i when making a prediction for j. In feature weighting methods, the weight could be any real number, however, in this paper we use only positive weights. Feature selection is here implemented by assigning a weight equal to 1 for the item/feature that is selected, and 0 otherwise.

The role of the target item j needs some further explanation. In fact this gives to a feature weighting or selection method the flexibility to assign to each predictive item a weight that depends on the target item whose rating prediction is sought. Hence, the weights used for predicting the rating of item j (for all users) can be different from those used for making predictions on another item j'. In this way, we can exactly encode in a weight how much two items are correlated, and what is the role that an item should play in neighborhood formation when a prediction is sought for the second one. Without such a flexibility we could only express knowledge about how important an item would be for all rating predictions. In fact, there are some examples of item weighting approaches that do not make this distinction and compute the absolute importance of an item. An example of this approach is the variance weighting method [3] that will be explained shortly.

For efficiency reasons, in all the feature weighting and feature selection algorithms that we shall describe later, we first compute all the weights and later on we use these stored values in the user-to-user similarity computation. In practice, to store all the weights, we need an $M \times M$ matrix of weights, where M is the cardinality of the item set. In other words, one vector of weights of size M is used for each item.

4 Items Weighting Methods

Feature weighting tries to estimate how much a particular feature is important for deciding about the class membership of instances. In CF, item weights can be

learned while exploring training data consisting of user ratings, or using external information associated with the items. Based on this observation we classify the methods we are going to present in two groups. The methods in the first group determine item weights using statistical approaches and are aimed at estimating statistical dependencies within rating data. The second group of methods uses item descriptions to estimate item dependencies and finally infer the weights. In the rest of this paper we will consider Random, Variance, Mutual Information, Tag label based, and PCC based feature weighting approaches.

Random. The first method is the baseline and uses a random item weighting. Random weights in $[0, 1]$ are selected for each target item and predictive item combination.

Variance. Variance method is based by an observation originally made by [3], that knowing a user's ratings on certain items could be more valuable than other ratings in discerning a user's interests. For example, if all users rate a particular item with the maximum rating value, this information is not much valuable to determine a user preferences and therefore should not be exploited in the user-to-user correlation (similarity). Hence, the variance method gives higher weights to items with higher variance among the ratings provided by the users to that item:

$$w_{ji} = w_i = \frac{\sum_{u=1}^{n} (v_{ui} - \bar{v_i})^2}{n-1}$$

here $\bar{v_i}$ is the mean of the ratings of item i, and n is the number of users in the database. Variance feature weighting method uses only information on a single item, the item that is weighted. All the methods that we're presenting next explore internal relations between predictive items and target item.

IPCC. The first method in this group uses PCC as measure of dependency between two vectors, which represent the ratings of all users for the two items. Since PCC ranges from -1 to 1, and we are using only positive weights we transform PCC to IPCC as follow, to take values between 0 and 1:

$$w_{ji} = \frac{\frac{\sum_u (v_{ui} - \bar{v_i})(v_{uj} - \bar{v_j})}{\sqrt{\sum_u (v_{ui} - \bar{v_i})^2 \sum_u (v_{uj} - \bar{v_j})^2}} + 1}{2}$$

here u runs on all the users in the database that have rated both i and j, and $\bar{v_i}$ is the mean of item i ratings. The usage of Pearson Correlation Coefficient between items to determine item weights has not been used in previous researches.

Mutual Information. Mutual Information measures the information that a random variable provides to the knowledge of an other. In CF we compute the mutual dependency between target item variable and predictive item variable (the values are the ratings). Mutual Information of two random variables X and Y is defined as :

$$I(X;Y) = \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

The above equation can be transformed into the $I(X;Y) = H(X) + H(Y) - H(X,Y)$, where $H(X)$ denotes the entropy of X, and $H(X,Y)$ is the joint entropy of X and Y. Using the above formula we compute weights as:

$$w_{ji} = -\sum_{r=1}^{5} p(v_i = r) \log p(v_i = r) - \sum_{r=1}^{5} p(v_j = r) \log p(v_j = r) + \sum_{r'=1}^{5} \sum_{r''=1}^{5} p(v_i = r', v_i = r'') \log p(v_i = r', v_i = r'')$$

Here the probabilities are estimated with frequency counts, hence for instance $p(v_i = r) = \frac{\#users\ who\ rated\ i\ as\ r}{\#users\ who\ rated\ i}$ is the probability that the item i is rated with value r, and r is running through all rating values (in our data set this is 5).

Tag weighting. All the previous methods exploit statistics of the users rating data to compute the feature weights. The last method we present here computes weights using items' description. The general idea is to increase the weight w_{ji} if the description of the target item j and the predictive item i are similar. In Tag weighting we increase the weight w_{ji} if the target item j is labelled with a tag that is also used to tag the predictive item i. In the movie recommendation dataset that we are going to use for our experiments, movies are tagged with movie genres (a movie can be tagged with more than one genre). Hence, we make the assumption that the larger the number of common tags (genres), the higher is the dependency. The weight of the predictive item i for a prediction of the ratings of item j is given by:

$$w_{ji} = 1 + \frac{\#\ co-labeled\ classes\ between\ i\ and\ j}{\#classes}$$

We note that [8] introduces the idea of using domain specific information to selectively choose the items to be used in the user-to-user correlation.

5 Item Selection

Feature selection is a technique commonly used in machine learning to filter out irrelevant or noisy features in classification learning tasks. While in classical classification learning each instance can belong to a single class, in CF rating prediction, given an active user (instance), the goal is to make a rating prediction for the largest number of items in the database. Therefore, instead of a single classification problem for each user (instance) we have a classification problem for each target user and target item pair. Hence, as we mentioned above, we would like to have one set of relevant items (predictive items) for each target item. The optimal feature selection problem requires to conduct a search procedure in the space of the subsets of the features [5]. Applying this to a recommender system scenario would require to conduct a search procedure for every target item (the item playing the role of the class to be predicted) and

for a large number of features , i.e., the predictive items. Conversely, we propose to use a dynamic approach that reuses the information provided by the feature weighting methods to select, for each target item and user pair, an appropriate set of predictive items. Hence, first we compute feature weights, using one of the methods described above, and then we filter out irrelevant features, i.e., the predictive items with the smallest weights, for any given target item. Therefore, every method used for feature weighting can be transformed into one or more corresponding feature selection method depending on how the weights are used to select items.

In the rest of the section we describe three different feature selection methods for CF systems. We will explain all three methods using an example of user-item rating matrix showed bellow:

	$i_1(w_{t1}=0.1)$	$i_2(w_{t2}=0.2)$	$i_3(w_{t3}=0.3)$	$i_4(w_{t4}=0.4)$	$i_5(w_{t5}=0.5)$	i_t
u_1	5	3	2	1	?	6
u_2	4	2	4	?	5	?

The table consist of two user and six items. Question marks indicate unknown ratings. Let us assume that we want to predict user's u_2 rating for the item i_t. Moreover, suppose that we have computed item weights beforehand, using one item weighting algorithm, and weights for all the items are showed in the first line in the brackets. For example, the weight of item i_1 for predicting the target item i_t is $w_{t1} = 0.1$

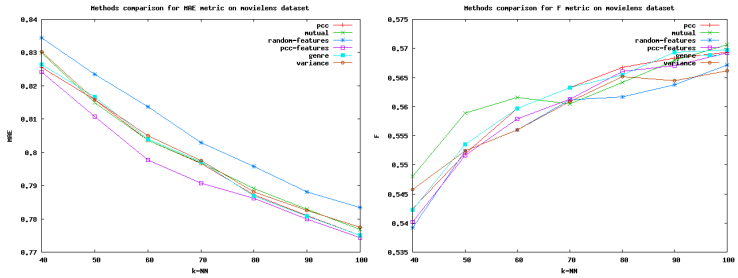
The first method, called **best-f-per-target**, selects the f items (predictive items) with highest weights. In other words, from the set of weights $W = \{w_{t1}, \ldots, w_{tM}\}$ the algorithm selects the f items with highest weights. Using the example above, If f is set to 3, then best-f-per-target would select items 5, 4 and 3. Best-f-per-target does not take into account, if the target user, or the neighbor user express any rating for the selected items. Therefore, some users can have no available ratings for the selected items.

To overcome this problem, we propose a second method, called **best-f-per-user** that selects the best f items, according to the weights, that are present in the target user's profile. Using the same example, in this case best-f-per-user would select the items 5, 3 and 2. Hence, in best-f-per-user predictive items are selected on a user base and are kept stable for all the predictions for a user.

The last method extends the idea of selecting the best items that are actually presents in the user profiles even further. The method **best-f-per-overlap** selects the f items with largest weights that are present in both the target user profile and in the neighbor user profile. In the example above, method best-f-per-overlap would select items 3, 2, and 1. Hence, in best-f-per-overlap the items used in each rating prediction change for every target item and user pairs.

6 Experimental Evaluation

This section evaluates feature weighting and feature selection methods for CF. All methods compute the set of feature weights and later use them in WPCC to



(a) MAE measure (b) F measure

Fig. 1. Performance of item weighting methods.

compute user-to-user correlation. To generate the prediction, WPCC is used in both, computing k nearest neighbors of the target users, and also to determine the predicted rating in the prediction step:

$$v^*_{xj} = \bar{v_x} + \frac{\sum_{y \in N(k,x)} WPCC(x,y,j) \times (v_{yj} - \bar{v_y})}{\sum_{y \in N(k,x)} |WPCC(x,y,j)|}$$

here the sum runs on the k-nearest neighbors of the user x, $N(k,x)$.

In our implementation, as previously done in [8], we do not take into account neighbors which have less than six overlapping rated items. In our experiments we used the MovieLens dataset, containing 100K ratings, for 1682 movies by 943 users. The data sparsity is 96%, and each rating can vary in the range $\{1,2,3,4,5\}$.

To evaluate the proposed methods the full dataset was randomly divided into train (80K) and test (20K) cases. We used the train dataset to learn the weights and also to make a prediction for the test ratings. We evaluated the performance of the proposed methods with a wide range of error measures.

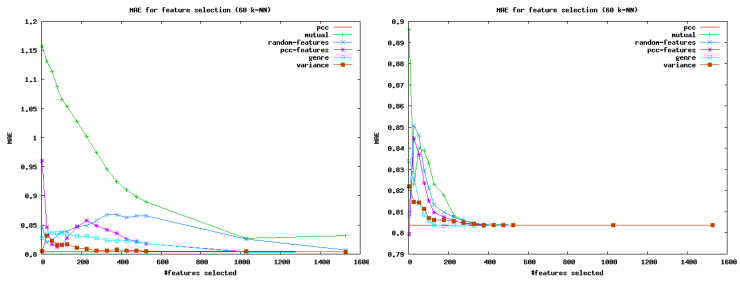
To measure the accuracy we used: MAE, High MAE, F measure, precision and recall [3]. To compute F, precision and recall measures, we considered items worth recommending (relevant items) only if their ratings were 4 or 5. Since, we are interested in recommending only the top items, we propose to modify the MAE error measure to see how an algorithm performs on predicting the highest ratings. We defined High MAE measure as the MAE obtained only on ratings with values 4 and 5.

In the first experiment we evaluated all feature weighting methods with a varying number of nearest neighbors. The baseline prediction method is the standard unweighed CF using PCC as user-to-user similarity measure (named *pcc* in all figures). Figure 1 shows the performances of the six proposed weighting methods with the number of neighbors ranging from 40 to 100.

It is clear from these results that there is no single best method for all error measures and nearest neighbors (for lack of space we do not show the results obtained with all the error measures). Some methods perform better than other in specific situations and sometimes they perform even worse than the standard CF. As expected, random feature selection performs worst and degrades the base line unweighed approach (Figure 1(a).) Considering only MAE we can observe that feature weighting based on IPCC performs better than other feature weighting or baseline method up to 80 neighbors. With 60 nearest neighbors it gets up to 1.5% improvement, and random-feature weighting loses 1.2% accuracy. In fact, the performance gains even for the best performing feature weighting methods are small. We should mention that in our experiments we did not observe any improvement for Mutual Information weighting, contradicting what is reported in [10]. That could be due to the fact that we used different datasets and different missing value interpretation. In [10] Yu et al. used the EachMovie dataset and discarded users, who have rated less than 20 items. Moreover, in our implementation when computing entropy we skipped undefined summation terms like $p(a)log(p(a))$ when, $p(a) = 0$. The interpretation of undefined terms is not clear in [10]. Considering F measure, it is clear that there is no single method, which outperforms all other methods. Random feature weighting is still the worst, however, the difference is not large. Here, differently from MAE measure, Mutual Information performs well, especially when the number of nearest neighbors is small.

We need to mention that, the weighting methods which depend on the target item, suffer from several drawbacks. First of all, the memory complexity is quadratic in the number of items, because they use $M \times M$ matrix to store the weights (M is the number of items). This makes such methods not easily applicable for big datasets with a lot of features/items. Another problem, is related to the computation of the nearest neighbors for a given user. To speed up this computation, most of the implementations index the nearest neighbors or cache the user-to-user similarity values for most of the active users. When the feature weights depend on the target item caching is hardly possible. In fact, in the worst case, in order to cache the user neighborhood we would have to store one neighborhood for each of the item and user pairs, as the user-to-user similarity uses weights that depends on the target item.

The second experiment evaluates the first item selection method we proposed, i.e., best-f-per-target. To compute the weights we used the five feature weighting methods described in Section 5. In figure 2(a) the performance of all these methods are depicted. We note that in general all the methods performed worse than the baseline, i.e., standard CF without item selection. The motivation is that users tend to rate only a small fraction of the total available items and the number of co-rated items between two users can be very small and even null. Selecting items, even if relevant, decreases the number of co-rated items even more. Imagine for instance, that there are two users that are perfectly correlated but have co-rated a few items. One user could be used to predict ratings of second user. But, when we select a small number of items we have a very



(a) "best-f-target" (b) "best-f-user"

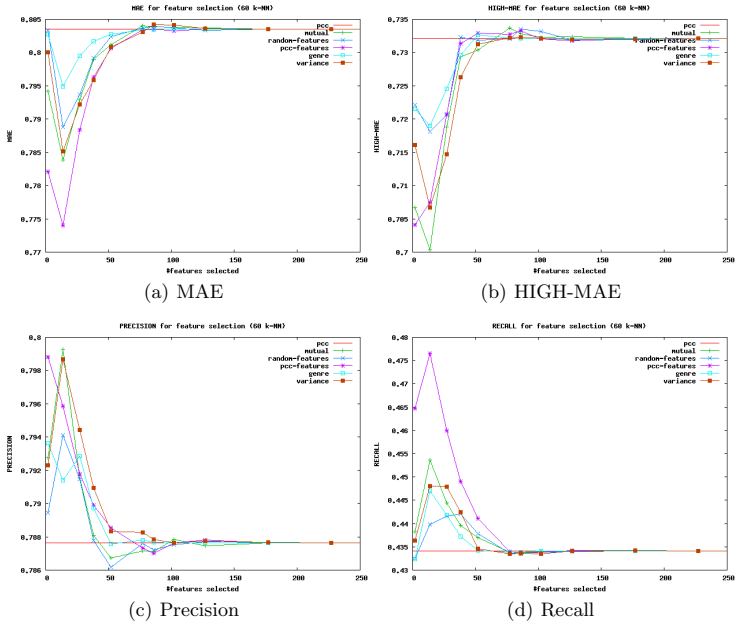
Fig. 2. MAE for two item selection methods.

small chance that these users will overlap on the selected items. Therefore, using this simple item selection procedure, many good neighbors could be discarded degrading the prediction accuracy. Anyway it is important to note that using a smaller number of items the accuracy can be pretty close to the standard CF system and when one uses more and more items the accuracy converge to that of the standard CF method.

In the second experiment we tested the best-f-per-user method . The accuracy (see Figure 2(b)) loss is smaller than best-f-per-target, and a faster convergence to the accuracy of the standard CF is obtained with very few items. However in general we do not really improve the accuracy. We observe that with 300 items we obtain the same accuracy of the baseline method. It can be explained by the fact, that virtually no users rate more than 300 items and much less ratings are needed to compute a reliable prediction.

In the final experiment we evaluated the best-f-per-overlap method. Figure 3 shows results for different accuracy metrics. Here, selecting a small number of items, we can observe a significant improvement of all the accuracy measures. If we select a larger and larger number of features every method results in the same accuracy as the base line method. We note that there is no single best performing method and the winner depends on the particular error measure we use. In fact, for MAE we decrease the error up to 3.7% using PCC based feature weighting method and for Recall measure the improvement is up to 9%. Random feature selection method and genre labelling method are the worst, however, they can also improve the performance of the baseline method.

It is worth noting that item selection based on best-f-overlap makes the user-to-user similarity much faster to compute since it is based on a small number of overlapping items. In fact, with only 14 overlapping items the best performances can be achieved. Moreover, for the same reason, it is possible to explain to the user her correlation with her neighbors and increase trust on the system prediction providing transparency on the recommendation mechanism.



(a) MAE (b) HIGH-MAE

(c) Precision (d) Recall

Fig. 3. Performance of "best-f-overlap" item selection methods.

7 Conclusions

In this paper we compared a number of feature weighting methods, some original and some previously proposed in the literature. We introduced IPCC based weighting method and adopted tag weighting method to CF. Furthermore we introduced three feature selection methods applicable to CF prediction that are based on feature weighting. We evaluated the proposed methods along with a range of error measures and varying the number of nearest neighbors. The main conclusion of this work is that there is no single best feature weighting method, and accuracy improvements using weighting are very small. Considering instead the item selection methods, we show that dynamic best-f-per-overlap outperforms item weighting and the two other item selection methods. In summary, best-f-per-overlap, using a small number of items, can achieve significant improvements for all considered error measures. This result is important because it shows that CF user profiles contain a lot of redundancy and even if the dataset is sparse the information is not uniformly distributed among users. This work also shows that CF performances can be improved with a careful selection of

the item ratings, i.e., acquiring ratings for certain items can greatly improve the performance of the recommendation algorithm.

In the future we will compare feature weighting methods with other techniques such as instance (user) selection methods. As we noticed above, we observed that there is no single method outperforming the others in all cases. Therefore, we want to design a meta learning approach that tunes user-to-user similarity according to a number of contextual and reliability indicators.

The computation of the feature weights is an expensive step in all considered algorithms. Moreover, we need to recompute the weights when new ratings are registered to the system. At the moment we are working on a feature weighting method based on the RELIEF [6] feature estimator in order to avoid feature weights recomputation. Using this method with every new rating we could gradualy adapt the overal weighting schema.

References

1. G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749, 2005.
2. J. S. Breese, D. Heckerman, and C. M. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In G. F. Cooper and S. Moral, editors, *UAI*, pages 43–52. Morgan Kaufmann, 1998.
3. J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR*, pages 230–237. ACM, 1999.
4. R. Jin, J. Y. Chai, and L. Si. An automatic weighting scheme for collaborative filtering. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 337–344, New York, NY, USA, 2004. ACM Press.
5. R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.
6. M. Robnik-Šikonja and I. Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Mach. Learn.*, 53(1-2):23–69, 2003.
7. G. Salton and M. J. Mcgill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.
8. T. K. Shlomo Berkovsky and F. Ricci. Cross-domain mediation in collaborative filtering. In C. Conati, K. McCoy, and G. Paliouras, editors, *LNAI*, volume 4511 of *Lecture Notes in Artificial Intelligence*, pages 355–359. Springer, 2007.
9. D. Wettschereck, D. W. Aha, and T. Mohri. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artif. Intell. Rev.*, 11(1-5):273–314, 1997.
10. K. Yu, X. Xu, M. Ester, and H.-P. Kriegel. Feature weighting and instance selection for collaborative filtering: An information-theoretic approach*. *Knowl. Inf. Syst.*, 5(2):201–224, 2003.

Using Term-matching Algorithms for the Annotation of Geo-services

Miha Grcar[1], Eva Klien[2]

[1] Jozef Stefan Institute, Dept. of Knowledge Technologies, Jamova 39,
1000 Ljubljana, Slovenia
miha.grcar@ijs.si
[2] Institute for Geoinformatics, Robert-Koch-Str. 26–28,
48149 Münster, Germany
klien@uni-muenster.de

Abstract. This paper presents an approach for automating semantic annotation within service-oriented architectures that provide interfaces to databases of spatial-information objects. The automation of the annotation process facilitates the transition from the current state-of-the-art architectures towards semantically-enabled architectures. We see the annotation process as the task of matching an arbitrary word or term with the most appropriate concept in the domain ontology. The term matching techniques that we present are based on text mining. To determine the similarity between two terms, we first associate a set of documents [that we obtain from a Web search engine] with each term. We then transform the documents into feature vectors and thus transition the similarity assessment into the feature space. After that, we compute the similarity by training a classifier to distinguish between ontology concepts. Apart from text mining approaches, we also present two alternative techniques, namely hypothesis checking (i.e. using linguistic patterns such as "$term_1$ is a $term_2$" as a query to a search engine) and Google Distance.

Keywords: geo-services, semantic annotation, text mining, search engine querying, machine learning, term matching

1 Introduction and Motivation

This paper presents an approach for automating semantic annotation within service-oriented architectures that provide interfaces to databases of spatial-information objects. The automation of the annotation process facilitates the transition from the current state-of-the-art architectures towards semantically-enabled architectures. The techniques presented in this paper are being developed in the course of the European project SWING[1] which deals with introducing semantics into spatial-data infrastructures to support discovery, composition, and execution of geo-services.

[1] Semantic Web Services Interoperability for Geospatial Decision Making (FP6-026514) <http://www.swing-project.org>

In SWING, semantic annotation is understood as the process of establishing explicit links between geographic information that is served via OGC[2] services and the vocabulary defined in the domain ontology (i.e. the vocabulary of a specific geo-informatics community). Once the bridge between the two sides is established, the domain ontology can be employed to support all sorts of user tasks.

The main purpose of this paper is to present data mining techniques that facilitate the annotation process. The annotation process can be seen as the task of matching an arbitrary word or term with the most appropriate concept in the domain ontology. Most of the term matching techniques that we present are based on text mining (see Sections 4.1–4.2). To determine the similarity between two terms, we first associate a set of documents with each term. To get the documents, we query search engines, on-line encyclopaedias, dictionaries, thesauri, and so on (query being the term in question). We then transform the documents into feature vectors. By doing so, we transition the similarity assessment into the feature space. Several text mining approaches are at hand to compute the similarity in the feature space – either by computing centroids or by training classifiers to distinguish between ontology concepts. Apart from the techniques based on document similarity, we also present two alternative techniques, namely hypothesis checking (i.e. using linguistic patterns such as "$term_1$ is a $term_2$" as a query to a search engine; see Section 4.3) and Google Distance (see Section 4.4). All the techniques are demonstrated on a toy example from the domain of mineral resources.

2 Related Work

Several knowledge discovery (mostly machine learning) techniques have been employed for ontology learning tasks in the past [6]. Text mining seems to be a popular approach to ontology annotation because the text mining techniques are shown to produce relatively good results.

We reference much of the related work from the corresponding sections. In the context of text mining we discuss centroid computation and classification [1] (see Section 4.1.1), Support Vector Machines [11] (see Section 4.1.2), *k*-NN, and classification in general [8] (see Section 4.1.2). Apart from the text learning techniques we also deal with linguistic patterns introduced by Hearst [7] (see Section 3.3), and Google Distance [2] (see Section 4.4).

3 Baseline for the Annotation of Geo-services

Normally, geo-data is served by a database of spatial-information objects through a standardized interface. In SWING, we use OGC-defined standard interfaces, namely Web feature services (WFS) [9], to access spatial-information objects. Web feature services are required to implement the capability to describe objects (termed "features") that they serve (see Fig. 1, the left-hand side). These descriptions

[2] **O**pen **G**eospatial **C**onsortium <http://www.opengeospatial.org>

(schemas) contain the definition of each available class of objects in a similar fashion as a data structure is defined in an object-oriented modeling or programming language: the schema provides the class name and its attributes; each attribute is described by its own name and the corresponding data type.

On the other hand we have real-word entities such as trees, rivers, minerals, quarries, and so on. These are modeled as axiomatized concept definitions in the form of a domain ontology that captures a specific view on the world (see Fig. 1, the right-hand side). The core idea is to employ the domain ontology for the discovery and composition of Web feature services, and also for the retrieval of spatial-information objects (i.e. for the invocation of Web feature services).

In order to support these user tasks, we need to establish a bridge between the WFS schema on one side and the domain ontology on the other. The process of establishing this link is called annotation. We see the annotation as a two-step process. The first step is a simple syntactic translation from a WFS schema description into the appropriate ontology language. We use WSML [3] as the ontology-description language in SWING. We call a WFS described in WSML a "feature-type ontology" (FTO). FTOs do not differ much from the original WFS descriptions apart from being formulated in a different description language.

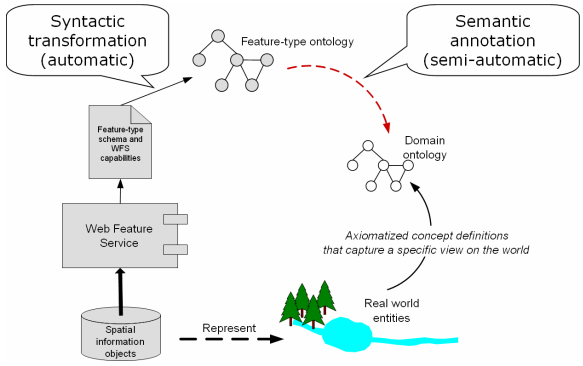


Fig. 1. The two-step semantic annotation process.

The first step thus establishes the syntactic compatibility between a WFS and the domain ontology (i.e. both descriptions are put into the same ontology-description language). However, the two descriptions are not yet semantically interlinked. The second step thus associates concepts and properties from FTOs with the domain ontology concepts. This process is described in the following section.

4 Automating the Annotation Process

Let us first define the problem of mapping one concept to another in more technical terms. We are given a feature-type ontology (FTO) concept as a single textual string (e.g. OpenPitMine) and a domain ontology which is basically a directed graph in which vertices represent concepts and edges represent relations between concepts. Each concept in the domain ontology is again given as a single textual string (e.g. D:Quarry[3]). The task is now to discover that OpenPitMine is more closely related to D:Quarry as to for instance D:Legislation or D:Transportation. Also important to mention is that every FTO concept has a set of attributes. Each attribute is given as a single textual string (e.g. OpenPitMine.SiteName) and has its corresponding data type (the data type is not expected to provide much guidance in the annotation process since it is usually simply *string*). Concepts in the domain ontology can similarly be described with the surrounding concepts, e.g. D:Quarry-hasLocation-QuarryLocation[4].

A straightforward approach would be to try to compare strings themselves. Even by taking attribute strings into the account coupled with some heuristics we cannot hope for good results – this can serve merely as a baseline.

In the following we present several promising approaches that use alternative data sources (mostly the Web) to discover mappings between concepts. We limit ourselves to a scenario where attributes are not available (i.e. we are given merely an FTO concept and a set of domain ontology concepts). The task is to arrange domain ontology concepts according to the relatedness to the FTO concept. In the examples we will use OpenPitMine as the observed FTO concept and domain ontology concepts D:Quarry, D:Legislation, and D:Transportation.

In Section 4.1 we first introduce the idea of concept comparison by populating concepts with (textual) documents that reflect semantics of these concepts. To enable the realization of these ideas in the context of SWING we first need to resolve the fact that the concepts are not a-priori populated with documents. Section 4.2 presents two promising techniques of using a Web search engine (in our particular case: Google) to acquire the "missing" documents. Sections that follow (4.3 and 4.4) present two alternative ways of using the Web for the annotation. Rather than dealing with documents, these approaches deal with term co-occurrences and linguistic patterns, respectively.

4.1 Comparing Documents to Determine Concept Similarity

Suppose we have a set of documents assigned to a concept and that these documents "reflect" the semantics of the concept. This means that the documents are talking about the concept or that the domain expert would use the concept to annotate (categorize) these documents.

[3] With prefix D: we denote concepts that belong to the domain ontology.

[4] This denotes a domain ontology concept named Quarry with "attribute" QuarryLocation which is linked to the concept via the hasLocation relation.

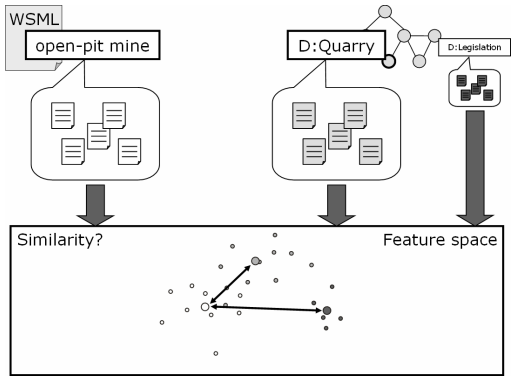


Fig. 2. Transitioning the similarity assessment into the feature space by transforming the documents into the corresponding *tfidf* feature vectors. This figure also illustrates how the comparison of centroids (discussed later on in Section 4.1.1) can be used to conclude that OpenPitMine (represented with white documents/dots) is associated with D:Quarry (represented with light gray documents/dots) stronger than with D:Legislation (represented with dark gray documents/dots). This conclusion is based on the fact that the white centroid is closer to the light gray centroid than to the dark gray centroid. The centroids are represented with the larger dots.

In such cases we can compute the similarity between two concepts. We are given an FTO concept (in our case OpenPitMine) and several domain ontology concepts (in our case D:Quarry, D:Transportation, and D:Legislation) with their corresponding document sets (see Fig. 2). We first convert every document into its bag-of-words representation, i.e. into the *tfidf* representation [6]. A *tfidf* representation is actually a sparse vector of word-frequencies (compensated for the commonality of words – this is achieved by the *idf* component – and normalized). Every component of a *tfidf* vector corresponds to a particular word in the dictionary. With "dictionary" we refer to all the different words extracted from the entire set of documents. If a word does not occur in the document, the corresponding *tfidf* value is missing – hence the term "sparse vector". Each of these vectors belongs to a certain concept – we say that the vector is *labelled* with the corresponding concept. This gives us a typical supervised machine learning scenario. In the following subsections we present three different approaches to concept-concept similarity computation using different machine learning approaches.

Comparing Centroids to Determine Concept Similarity. A centroid is a (sparse) vector representing an (artificial) "prototype" document of a document set. Such prototype document should summarize all the documents of a given concept. There are several ways to compute the centroid (given *tfidf*s of all documents in the corresponding set). Some of the well-known methods are the Rocchio formula,

average of vector components, and (normalized) sum of vector components. Of all the listed methods, the normalized sum of vector components is shown to perform best in the classification scenario [1]. In the following we limit ourselves to the method of normalized sum. We first represent documents of a particular concept C as normalized *tfidf* vectors $\vec{d}_i$. Now we compute the centroid as given in Eq. 1.

$$\vec{c} = \frac{1}{\|\vec{c}\|} \sum_{d_i \in C} \vec{d}_i \ . \tag{1}$$

Having centroids computed for all the concepts, we can now measure similarity between centroids and interpret it as similarity between concepts themselves (we are able to do this because a centroid summarizes the concept it belongs to). This is illustrated in Fig. 2. Usually we use cosine similarity measure [6] to measure similarity between two centroid vectors.

Table 1. The performance of some of the discussed algorithms on our toy example. In the case of the centroid-to-centroid similarity computation, the numbers represent cosine similarity between the terms; in the case of the Normalized Google Distance, the numbers represent the distance measure, and in the case of *k*-NN, the sum of all the cosine similarities measured between an FTO document and a domain ontology document from the corresponding neighborhood. The results are given for two different contexts: the general context and the context of "extracting material" (note that the contextualization is not applicable if Google definitions are used as the data source). Top 30 search results are considered when querying the search engine. Only English definitions and English search results are considered. In the case of *k*-NN, *k* is set dynamically by taking all the documents within the cosine similarity range of less than (or equal to) 0.06 into account. The number that represents the strongest association between the corresponding two terms is emphasized.

		open pit mine	
		general context	"extracting material"
Centroid Google search	quarry	**0.11**	**0.39**
	legislation	0.01	0.09
	transportation	0.02	0.05
Centroid Google definitions	quarry	**0.39**	N/A
	legislation	0.01	N/A
	transportation	0.05	N/A
k-NN Google search	quarry	**0.61**	**2.82**
	legislation	0	0.43
	transportation	0	0.10
k-NN Google definitions	quarry	**3.17**	N/A
	legislation	0	N/A
	transportation	0.35	N/A
NGD	quarry	**0.02**	1.92
	legislation	0.42	3.55
	transportation	0.50	**1.71**

Employing Classification to Determine Concept Similarity. We already mentioned that every *tfidf* vector is *labelled* with the corresponding concept and that this gives us a typical supervised machine learning scenario. In a typical supervised machine learning scenario we are given a set of training examples. A training example is actually a labelled (sparse) vector of numbers. We feed the training examples to a classifier which builds a model. This model summarizes the knowledge required to automatically assign a label (i.e. a concept) to a new yet unlabelled example (we term such unlabelled examples "test examples"). This in effect means that we can assign a new document to one of the concepts. We call such assignment (of a document to a concept) classification.

How do we use classification to compare two concepts? The approach is quite straightforward. We take the documents belonging to a particular FTO concept (in our case the documents of OpenPitMine) and strip them of their label thus forming a test set. Now we assign each of these documents to one of the domain ontology concepts (i.e. we classify each of the documents to one of the domain ontology concepts). The similarity between an FTO concept and a domain ontology concept is simply the number of FTO-concept documents that were assigned to that particular domain ontology concept. Many classifiers assign the same document to all the concepts at the same time but with different probabilities or confidence. We can compute the sum of these probabilities/confidence values instead of simply counting the documents.

There are many different classifiers at hand in the machine learning domain. Herein we discuss a very popular classifier – Support Vector Machine (SVM). In addition we also discuss how the same task is performed with the *k*-nearest neighbors (*k*-NN) algorithm which has the property of a "lazy learner". The latter means that *k*-NN does not build a model out of the training examples – instead, it uses them directly to perform classification.

Classification with SVM. In its basic form, SVM is able to classify test examples into only two classes: positive and negative. We say that SVM is a binary classifier. This means that training examples must also be only of the two kind: positive and negative. Since examples are vectors, we can see them as points in a multi-dimensional space. The task of SVM is to find such hyper-plane that most of the positive training examples lie on one side of the hyper-plane while most of the negative training examples lie on the other side. Formally, SVM is an optimization problem that can be solved optimally. Recently it has been shown that this can actually be done in linear time for linear kernels [10], which is quite a breakthrough regarding the usefulness and quality of SVM.

Even though SVM is binary, we can combine several such classifiers to form a multi-class variant of SVM. Several multi-class variants are discussed and evaluated in [4].

Classification with k-NN. We already mentioned that *k*-NN is one of the "lazy learners" which means that it does not build a model out of training examples. It performs the classification of a document by finding *k* most similar documents of all the documents that belong to the domain ontology concepts. The similarity between the document and a domain ontology concept can be computed as the number of documents (from the set of *k* most similar documents) that belong to that domain ontology concept. Instead of simply counting the documents we can compute the sum of the corresponding cosine similarities. As an alternative to defining a constant

neighborhood size, we can set *k* dynamically by taking all the documents within the cosine similarity range of less than (or equal to) a predefined threshold into account.

4.2 Google Definitions and Contextualized Search Results

"If Google has seen a definition for the word or phrase on the Web, it will retrieve that information and display it at the top of your search results. You can also get a list of definitions by including the special operator 'define:' with no space between it and the term you want defined. For example, the search 'define:World Wide Web' will show you a list of definitions for 'World Wide Web' gathered from various online sources." (excerpt from Google Help Center <http://www.google.com/help/features.html#definitions>)

Googlebots crawl the Web all the time. In their expeditions they gather terabytes of data which is then processed in order to discover information that is potentially of particular interest to Google users (such as products for sale on-line, weather forecast, travel information, and images). One of such separately maintained information repositories are the definitions of words or phrases as found on the Web.

Google definitions can be used to compensate for the missing document instances – each definition (known by Google) can be seen as one document. In this way we can "populate" concepts with documents and then perform the mapping (i.e. the annotation) as already explained in Section 4.1.

To get back to our example, if we populate concepts OpenPitMine, D:Quarry, D:Transportation, and D:Legislation with document instances and then compare OpenPitMine (which is an FTO concept) to the domain ontology concepts (i.e. the other three concepts), we get centroid-to-centroid similarities as shown in Table 1. Since it is hard to find definitions for *n*-grams such as "open pit mine" (i.e. 3 or more words in a composition), we additionally query Google for the definitions of "pit mine" and "mine", weighting the contribution of these definitions less than the one of the initial composed word (if the "complete" definition exists, that is).

There are still some issues that need to be considered when using this approach. For one, a word can have several meanings, i.e. its semantics depends on the context (or the domain). Google does not know in which context we are searching for a particular definition – it thus returns all definitions of a particular word or phrase it keeps in its database. "Mine", for instance, can be defined either as "excavation in the earth from which ores and minerals are extracted" or as "explosive device that explodes on contact". It is important to somehow detect the documents that do not talk about the geospatial domain and exclude them from the annotation process.

Note that we can also populate concepts with Google search results (in contrast or even in addition to populating it with definitions). In this case we can put the search term into a context by extending it with words or phrases describing the context. For example: to populate concept OpenPitMine in the context of "extracting materials" with documents, we would query Google for "open pit mine extracting materials" and consider for instance the first 50 search results. Centroid-to-centroid similarities for this approach are also shown in Table 1.

4.3 Using Linguistic Patterns for Hypothesis Checking

We can use a Web search engine to estimate the truthfulness of a hypothesis given as a statement in a natural language. If we query Google for "quarry is an open pit mine", it returns 13 hits (at the time of writing this paper). We also get 3 hits for the query "mine is a quarry". In contrast, we do not get any hits for queries "quarry is a transportation" or vice versa, and "quarry is a legislation" or vice versa. We can check for synonymy between any two words (or even *n*-grams) w_1 and w_2 with this same pattern expressed as a template: "w_1 is a w_2" or "w_2 is a w_1". Hearst [7] introduced several such patterns for the acquisition of hyponyms. These patterns are thus called Hearst patterns.

Intuitively it seems that synonymy is the relation that is most suitable for the annotation task because we can infer similarity between two concepts from the "truthfulness of synonymy" (expressed for instance as the number of Google search results when "checking" the synonymy hypotheses) between these two concepts. However, hyponymy can be used to extend the set of synonymy hypotheses. The idea is to actually populate concepts with instances (in the true ontological sense) and then try to find synonymies between these instances. This is particularly useful in cases when the hypotheses checking on concepts (their string representations, more accurately) fails or yields inconsistent results. The system called KnowItAll [5] uses Hearst patterns and a set of Web search engines to populate concepts with instances.

4.4 Google Distance

Word similarity or word association can be determined out of frequencies of word (co-)occurrences in text corpora. Google Distance [2] uses Google to obtain these frequencies. Based on two requirements, namely (1) if the probability of word w_1 co-occurring with word w_2 is high then the two words are "near" to each other and vice versa, and (2) if any of the two words is not very common in the corpus, the distance is made smaller, the authors came up with the equation given in Eq. 2. They call it Normalized Google Distance (NGD).

$$\mathrm{NGD}(w_1, w_2) = \frac{\max\{\log f(w_1), \log f(w_2)\} - \log f(w_1, w_2)}{\log M - \min\{\log f(w_1), \log f(w_2)\}} \,. \tag{2}$$

In Eq. 2, f(w) is the number of search results returned by Google when searching for w (similarly f(w_1,w_2) is the number of search results returned by Google when searching for pages containing both terms), and M is the maximum number of pages that can potentially be retrieved (posing no constraints on language, domain, file type, and other search parameters, Google can potentially retrieve around 10 billion pages).

It is also possible to put NGD computation into a context. This can be done simply by extending Google queries (the ones that are used to obtain frequencies) with words that form the context. Note that in this case M must be determined as the number of returned search results when searching for the words that form the context. The performance of NGD on our toy example is evident from Table 1.

We believe that NGD is not really the best way to search for synonymy because synonyms generally do not co-occur. It is more a measure of relatedness or association – nevertheless it can be tried out for the SWING annotation task. Also note that any other search engine that reports the total number of search results can be used instead of Google.

4 A Preliminary Experiment

We tested some of the presented methods on a dataset from the domain of minerals. We obtained 150 mineral names together with their synonyms. To list just a few: acmite is a synonym for aegirite, diopside is a synonym for alalite, orthite is a synonym for allanite, and so on (this dataset can be found at http://www.csudh.edu/oliver/chemdata/minsyn.htm). The mineral names were perceived as our domain ontology concepts while the synonyms were perceived as the feature-type ontology concepts. For each of the synonyms, the selected algorithms were used to sort the mineral names according to the strength of the association with the synonym in question. We measured the percentage of cases in which the correct mineral name was in the top 1, 3, 5, and 10 names in the sorted list. In other words, we measured the precision of each of the algorithms according to the top 1, 3, 5, and 10 suggested mineral names.

We employed 16 algorithms altogether: 7 variants of *k*-NN, 5 variants of the centroid classifier, and 4 variants of NGD. We varied the context and the data source (either Google definitions or Google search engine). We also varied whether the order of words in a term matters or not (if the order was set to matter then the term was passed to the search engine in quotes). Top 30 search results were considered when querying the search engine. Only English definitions and English search results were considered. In the case of *k*-NN, *k* was set dynamically by taking all the documents within the cosine similarity range of less than (or equal to) 0.06 into account. The final outcome of a *k*-NN algorithm was computed as a sum of all the cosine similarities measured between a synonym document and a mineral name document from the corresponding neighborhood. Table 2 summarizes the results of the experiment. The best performing algorithm is emphasized in the table.

5 Conclusions

This paper presents several techniques for automatic annotation in which "external" data sources (such as the Web) are used to compensate for the missing textual documents corresponding to concepts.

According to the preliminary experiment presented in Section 5, the presented techniques have a very good potential. From the results it is evident that – at least for the dataset used in the experiment – it is not beneficial to limit the search to Wikipedia (a free Web encyclopedia available at http://www.wikipedia.org) or Google definitions. However, it proved useful to perform the search in the context of "minerals". Also important to notice is that *k*-NN outperforms the centroid classifier

when not put into a context. However, when the search is performed in the context of "minerals", the centroid classifier outperforms *k*-NN. This occurs because the documents, gathered by the contextualized search, are less heterogeneous. Consequently the centroid is able to summarize the topic (which has explicitly to do with "minerals") easier than if the documents were taken from the general context. Even more, the centroid cuts off the outliers (i.e. the noise) by averaging vectors' components. On the other hand, when dealing with more heterogeneous set of documents from the general context, the centroid is "shifted" towards irrelevant (sub)topics (i.e. other than "minerals") which results in poorer performance.

Table 2. The results of the preliminary experiment.

				Precision [%]			
Algorithm	Data source	Context	Quotes	Top 1	Top 3	Top 5	Top 10
k-NN	Google srch.	general	no	82.67	90	92	93.33
Centroid	Google srch.	general	no	78	91.33	92	94
k-NN	Google def.	general	no	70	76	77.33	79.33
Centroid	Google def.	general	no	76	77.33	78.67	79.33
k-NN	Google srch.	"site: wikipedia.org"	no	43.33	60.67	70	76
k-NN	Google srch.	"minerals"	no	86.67	97.33	98.67	100
Centroid	Google srch.	"minerals"	no	91.33	96.67	98.67	99.33
NGD	Google srch.	general	no	8.67	18	21.33	30.67
NGD	Google srch.	"minerals"	no	12.67	21.33	29.33	42.67
k-NN	Google srch.	general	yes	80.67	89.33	91.33	93.33
Centroid	Google srch.	general	yes	78	91.33	93.33	94.67
k-NN	Google srch.	"site: wikipedia.org"	yes	27.33	38.67	39.33	42
k-NN	Google srch.	"minerals"	yes	88	98	99.33	100
Centroid	**Google srch.**	**"minerals"**	**yes**	**93.33**	**98.67**	**99.33**	**100**
NGD	Google srch.	general	yes	16	26	36.67	54.67
NGD	Google srch.	"minerals"	yes	11.33	22.67	36.67	58

These conclusions lead us to an idea of how to improve the presented algorithms. We believe that it would be beneficial to first cluster the documents and then perform the classification on each cluster separately. The final classification result would then be obtained by maximizing or averaging the per-cluster classification results.

Also noticeable from Table 2, as we suspected, NGD is not very successful in detecting synonymy. Furthermore, it is much slower that the other presented algorithms as it is querying the Web to determine term co-occurrences. Last but not least, we can see that the best performing algorithms perform slightly better if the

search query is put into quotes (i.e. if the order of words in the term that represents the query is set to matter).

The focus of our future work will be on the implementation of the SVM-based term matching algorithm and the implementation of the clustering of documents. We have high hopes for these two technologies. We also lack the implementation of the hypothesis checking; however, we believe that using linguistic patterns for string matching will be inferior to the text mining approaches. The next step will be to go beyond string matching and also consider the neighborhood of a concept similarly to [12] (provided that the concept is a part of an ontology or a similar structure).

The implemented technologies will soon become a part of OntoBridge, an open-source system for semi-automatic data-driven ontology annotation (i.e. for mapping or mediation). OntoBridge will be one of the deliverables of the European project SWING.

Acknowledgments. This work was partially supported by the IST Programme of the European Community under SWING – Semantic Web Services Interoperability for Geospatial Decision Making (FP6-026514) and PASCAL Network of Excellence (IST-2002-506778).

References

1. Cardoso-Cachopo, A., Oliveira L., A.: Empirical Evaluation of Centroid-based Models for Single-label Text Categorization. INSEC-ID Technical Report 7/2006 (2006)
2. Cilibrasi, R., Vitanyi, P.: Automatic Meaning Discovery Using Google (2004)
3. de Bruijn, J.: The Web Service Modeling Language WSML (2005)
4. Duan, K.-B., Keerthi S., S.: Which is the Best Multiclass SVM Method? An Empirical Study. In LNCS vol. 3541/2005, Springer Berlin Heidelberg (2005)
5. Etzioni, O., Cafarella, M., Downey, D., et al.: Web-scale Information Extraction in KnowItAll (Preliminary Results). In Proceedings of WWW2004, New York, USA (2004)
6. Grcar, M., Klien, E., Fitzner, D., I., Maué, P., Mladenic, D., Grobelnik, M.: D4.1: Representational Language for Web-service Annotation Models. Project Report FP6-026514 SWING, WP 4, D4.1 (2006)
7. Hearst, A., M.: Automatic Acquisition of Hyponyms from Large Text Corpora. In Proceedings of COLING-92 (1992)
8. Mitchell, T. M.: Machine Learning. The McGraw-Hill Companies, Inc. (1997)
9. Open Geospatial Consortium: Web Feature Service Implementation Specification, Version 1.0.0 (OGC Implementation Specification 02-058) (2002)
10. Thorsten, J.: Training Linear SVMs in Linear Time. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 217–226, PA, USA (2006)
11. Vapnik, V.: Statistical Learning Theory. Wiley, New York (1998)
12. Gligorov, R., Aleksovski, Z., ten Warner, K., van Harmelen, F.: Using Google Distance to Weight Approximate Ontology Matches. In Proceedings of WWW 2007, Banff, Alberta, Canada (2007)

Using context models and models for contextually instantiated social relations for mobile social computing services *(Invited Talk)*

Georg Groh

Technische Universität München, München, Germany
grohg@in.tum.de

Abstract. Social network analysis and models for social structures have gained substantial interest in connection with Web 2.0, communities and other social computing paradigms. While numerous platforms provide means to manage personal social networks of simple kinds, few approaches have been investigated that aim at modelling instantiations of social relations and subsequently using these models for services which are socially- *and* context-aware at the same time. In contrast to the simple models of relations which always represent an average with respect to contextual parameters such as time and space, we will investigate models for describing the instantiations of these relations in time and space and discuss ideas for heuristic methods for identifying these instantiated relations algorithmically.
These models can be used for a broad spectrum of context-aware mobile services in the fields of Contextual Social Awareness, Contextual ocial Recommenders and Information Exchange as well as Context-sensitive Authorization and we will suggest ideas of how such services can be designed to effectively use instantiated social relation models.

A Hybrid Approach Towards Wrapper Induction

Stefan Raeymaekers and Maurice Bruynooghe

K.U.Leuven, Dept. of Computer Science, Celestijnenlaan 200A, 3001 Leuven, Belgium
{stefanr,maurice}@cs.kuleuven.ac.be

Abstract. The approaches to learn wrappers for extraction from semi-structured documents (like HTML documents) are divided into string based ones, and tree based ones. In previous papers we have shown that tree based approaches perform much better and need less examples than string based approaches, but have the disadvantage that they can only extract complete text nodes, whereas string based approaches can extract within text nodes. In this paper we propose a hybrid approach that combines the advantages of both systems. We compare this approach experimentally with a string based approach on some sub node extraction tasks.

1 Introduction

Wrappers that extract information from web pages are very useful to process data that is only available as a HTML document. The induction of wrappers from examples is an active research field, as manually crafting these wrappers is a tedious job. Moreover, they regularly require maintenance as a change in the templates of a site often invalidates the wrappers.

In a string based approach [1–6, 9, 10, 13], the document is mostly viewed as a sequence of tokens and markup tags from which a subsequence gets extracted. This is done by learning to recognize the *start* and *end* boundaries of the target substring. These boundaries are always between two tokens.[1] The markup tags define an implicit tree structure on the document. In the string representation, the relations in this tree are hidden, as in the flattened tree, parent or sibling nodes of a given node are separated by the tokens and tags that make up the subtrees under its (preceding) siblings. This renders the induction task more difficult.

Tree based approaches [8, 12] view the document as a tree, preserving the tree relations. In [11] we compare some state of the art string based approaches with state of the art tree based approaches and conclude that the latter are much more performant. They need less examples to induce a perfect wrapper, and the induction time is often orders of magnitudes lower. A limitation of these methods though is that they operate on the nodes of the tree and hence can only extract complete nodes. Tasks that require sub node extraction are out of scope. The common reply on this issue is that the tree based approach is a first step in a two level approach in which the sub node extractions are performed in a second step.

In this paper we show that this is indeed a viable approach. We investigate ways to extend a tree based approach with a string based approach as a second step. We

[1] Using characters instead of tokens is an overkill, as the target values do usually not contain half a token, and with tokens, the higher granularity speeds up the learning phase.

implemented one possibility and compared this hybrid approach experimentally with a state of the art string based approach.

As string based approach we have chosen the STALKER system [10]. This system reaches relatively good results by adopting a hierarchical approach. The learning task is split up in simpler tasks that are learned separately. We use the same system, without the hierarchical approach as an extension to the tree based system. There is no need for a hierarchical approach as the sub node learning task in the second step, are most of the times easier then the top level tasks in the hierarchical approach.

As tree based approach we use our system [12] based on (k, l)-contextual tree languages. In [11], an extended version of [12] this system is shown to have superior performance over both state of the art string and tree based systems.

The rest of the paper is organized as follows. In Section 2, we give an overview of the STALKER system, the string based approach we have chosen. Section 3 contains a summary of our wrapper induction with (k, l)-contextual tree languages. In Section 4 we discuss the main contribution of this paper, the extension to enable sub node extraction. Section 5 describes the experimental setup and relates the results. We conclude in Section 7. Below we introduce Example 1. This example will be used as a running example in the coming sections.

Example 1. A web site running a restaurant guide, allows for a search for restaurants based on parts of their name. The resulting list of restaurants is returned as a web page constructed from a fixed template. In Figure 1, a possible outcome is shown for a search on 'china'. From this web page we can extract the following fields: the name of the restaurant(N), its type(T), the city(C) where it is located, and a phone number(P). For each restaurant we could also extract the url(L) from the link (leading to more detailed address information). And from the top sentence, the search term(S) that generated the page can be found. Note that the occurrence of the search term in the name is rendered in italic, while the land code of the phone number is in bold.

2 String Based Approach

In this section we describe the STALKER system [10]. We start with explaining its hierarchical approach. Then we give the semantics of the extraction rules, and we conclude with the induction algorithm.

2.1 Hierarchical Extraction

In contrast with other string based methods, STALKER implements a hierarchical extraction approach. An *Embedded Catalog (EC)* describes the structure of the data. This is a tree structure where the leaves are fields, and the internal nodes either tuples or lists. Figure 2 shows the EC for Example 1. Note that the EC formalism might not be expressive enough to represent some more complex data structures. To extract a specific field, first the parent has to be extracted, and the extraction rules are then applied on the subsequence extracted for the parent. To extract the values of the 'City' field of Example 1, first the subsequence containing the search term and the list of restaurants is extracted.

a)
```
<html><body>
   <b>Restaurant Guide: search results for <i>china</i></b>
   <p><a>New<i>China</i>Town (chinese)</a>
      Brussels<b>Tel: +32</b>(0)2 345 67 89</a></p>
   <p><a>Royal<i>China</i>(chinese)</a>
      Leuven<b>Tel: +32</b>(0)16 61 61 61</a></p>
   <p><a><i>China</i>Garden (chinese)</a>
      Amsterdam<b>Tel: +31</b>(0)20-4321234</a></p>
</body></html>
```

b)

Fig. 1. Restaurant Guide (Example 1): a) HTML code; b) screen shot.

Then the complete list of restaurants is extracted. Then the individual restaurants are extracted. And finally from the subsequences for each restaurant, the 'City' field is extracted. The advantage of this approach is that complex extraction tasks are split into easier problems. Disadvantages are that more examples are needed to learn rules for every level of the hierarchy[2], and that errors in the different levels will accumulate.

2.2 Rules

To extract a subsequence from a sequence of tokens, the STALKER system uses a start and an end rule, to find the boundaries of that subsequence. The start rules are executed in forward direction from the beginning of the sequence, the end rules are executed in backward direction. A STALKER rule is either a simple rule or a disjunction of simple rules. In the latter case the boundary is given by the first simple rule that does not fail. A simple rule is a list of so-called landmarks. A landmark is a sequence pattern consisting of tokens and/or wildcards. On execution, the rule searches for a part of the sequence that matches the first landmark. From the end of this part the search for the second landmark is started, and so on. The boundary that is finally returned is either the end or the beginning of the part that matched the last landmark. Which one is indicated by a modifier; SkipTo or SkipUntil for respectively the end or the beginning (or BackTo and BackUntil for rules in the other direction). When the search for a landmark reaches the

[2] To learn list extraction, each example should consist of two consecutive elements of the list.

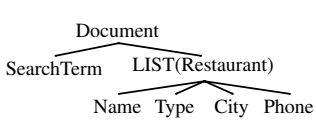


Fig. 2. Embedded Catalog for the restaurant guide example (Example 1).

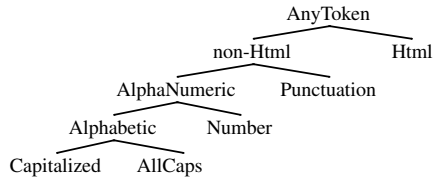


Fig. 3. Wildcard hierarchy. A token that matches a wildcard of a given type, will also match the wildcards of the ancestors of that type.

end/beginning of the sequence, the rule is said to fail. STALKER uses multiple types of wildcards that form a type hierarchy. This hierarchy is shown in Figure 3.

Example 2. Consider the first subsequence extracted for the tuple 'Restaurant':

```
New <i> China </i> Town ( chinese ) </a> Brussels <b>
Tel : + 32 </b> ( 0 ) 2 345 67 89
```

The rule *SkipTo(*`</a>`*)* applied on this sequence returns the position at the end of the first occurrence (and single occurrence in this example) of the tag '`</a>`', hence at the beginning of 'Brussels'. The rule *BackTo(*`<b>`*)* goes backward and returns the position at the end of 'Brussels'. Hence these rules are a start and end rule for the 'City' field.

For the rule *SkipUntil(*AnyToken*)* we see that the first token of the restaurant sequence is matched by the wildcard 'AnyToken'. As the modifier is 'until', the beginning of that token is returned. This is the beginning of 'New' for the above sequence. The rule *BackTo(*`</a>`*) BackTo(*`'('`*)* goes backward to the position before the first matching '`</a>`' token, and then continues going backward from there on until the first `'('` encountered. The position between 'Town' and '(' will be returned. Therefore these rules can be used to extract the 'Name' field.

To extract the sub-sequences for the tuple 'Restaurant' from the list of restaurants (or the sequence extracted for that list), we use the startrule and endrule repeatedly. The first start boundary though coincides with the start boundary of the list (and the last end boundary coincides with the end boundary of the list). The startrule *SkipTo(*`<p><a>`*)* returns the position at the end of the first occurrence of these two consecutive tags. The endrule for this extraction task is: *BackTo(*`</p>`*)*.

2.3 Induction Algorithm

The STALKER induction algorithm starts from a set of positive examples (each consisting of a sequence wherein boundaries of a subsequence are given). As long as this set is not empty, a new simple rule is learned, those examples covered by this rule are removed from the set, and that rule is added to the disjunction of rules that will be the final result. The algorithm to learn a simple rule chooses one *seed* example (the shortest example in the set) to guide the induction, the other examples are used to test the quality of candidate rules. The algorithm does not search the entire rule space for the best rule.

In each loop it takes two rules from a given set of rules, one is the best solution in that set, the other is the best refiner. Some heuristic rules are designed to define a ranking (best solution and best refiner) over a set of rules. This ranking is based on properties of the rules, and on the number and quality of the extractions of each rule on the other examples. The refinements of the best refiner, together with the best solution gives the new rule set for the next iteration. This loop continues until a perfect solution is found (one that either extracts correctly from an example or fails on that example) or until all refinements fail. The initial set of candidate rules are single landmark rules, with each landmark a single token or wildcard (occurring in the seed). The refinement step will either extend one of the landmarks of a rule with an extra token or wildcard (the extended landmark has to match within the seed), or add a new single token/wildcard landmark somewhere in the rule (the token or wildcard has to occur in the seed).

3 Tree Based Approach

In this section we define the notion of (k,l)-Contextual Tree Languages, as a subclass of the regular tree languages. In contrast with the whole class of regular languages, this subclass can be learned from positive examples only. The intuition behind (k,l)-contextual tree languages is fairly straightforward. At the base is a parameterized deconstruction of a tree into its building blocks called (k,l)-forks. These are subparts of the tree with maximally k consecutive children and a maximal depth of l. A tree belongs to a given language iff its (k,l)-forks all belong to the representative set of building blocks for that language. To learn a (k,l)-contextual tree language from examples, the (k,l)-forks of these examples are collected into a representative set for the learned language.

We start with formal definitions of (k,l)-forks and (k,l)-contextual tree languages. We then show how tree languages can be used to represent wrappers, and how to learn the parameters.

3.1 Preliminary Definitions

An alphabet Σ is a finite set of symbols. The set $T(\Sigma)$ of all finite, unranked trees with nodes labelled by elements of Σ can be recursively defined as $T(\Sigma) = \{f(w) \mid f \in \Sigma, w \in T(\Sigma)^*\}$. We denote $f(\epsilon)$, with ϵ the empty sequence, by f. The subtrees of a tree are inductively defined as $sub(f(t_1,\ldots,t_n)) = \{f(t_1,\ldots,t_n)\} \cup \bigcup_i sub(t_i)$. A *tree language* is any subset of $T(\Sigma)$. The set of *(k,l)-roots* of a tree $f(t_1,\ldots,t_n)$ is the singleton $\{f\}$ if l=1; otherwise, it is the set of trees obtained by extending the root f with $(k,l-1)$-roots of k successive children of t (all children if $k > n$). Formally:

$$R_{(k,l)}(f(t_1,\ldots,t_n)) = \begin{cases} \text{if } l=1 \text{ then } \{f\} \\ \text{if } l>1 \text{ and } k>n \text{ then } f(R_{(k,l-1)}(t_1),\ldots,R_{(k,l-1)}(t_n)) \\ \text{else } \bigcup_{p=1}^{n-k+1} f(R_{(k,l-1)}(t_p),\ldots,R_{(k,l-1)}(t_{p+k-1})) \end{cases}.$$

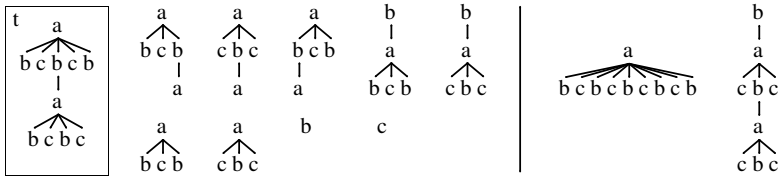
In this formula, $f(S_1,\ldots,S_n)$, denotes the set $\{f(s_1,\ldots,s_n) \mid s_i \in S_i\}$. In a similar notational extension, $R_{(k,l)}(T)$ denotes $\bigcup_{t\in T} R_{(k,l)}(t)$, the (k,l)-roots of a set T of trees. Finally, a (k,l)-*fork* of a tree t is a (k,l)-root of any subtree of t. Thus, the set of (k,l)-forks of t can be written as $R_{(k,l)}(sub(t))$ and we denote it by $F_{(k,l)}(t)$. Then the (k,l)-forks of a set of trees T are defined as $F_{(k,l)}(T) = \bigcup_{t\in T} F_{(k,l)}(t)$.

3.2 (k,l)-Contextual Tree Languages

Definition 1. *The (k,l)-contextual tree language based on the set G of trees is defined as* $L_{(k,l)}(G) = \{t \in T(\Sigma) \mid F_{(k,l)}(t) \subseteq G\}$.

As shown in [12], the language $L_{(k,l)}(F_{(k,l)}(E))$ is the most appropriate (k,l)-contextual tree language that can be learned from a set of positive examples E as it is the most specific (k,l)-contextual language that accepts all the examples. Generalization is controlled by the choice of the parameters; they determine the minimal granularity of the building blocks (the forks from the examples) that can be used in defining the language. Negative examples can be used to adjust the parameter values [12].

Example 3. Below we show graphically the $(3,3)$-forks of a tree t. The first three of these forks are the $(3,3)$-roots of t. Two trees from the language $L_{(3,3)}(F_{(3,3)}(\{t\}))$ are shown on the right.



3.3 Wrapper Induction

A marking of a tree $t \in T(\Sigma)$ is a function that maps a tree on a marked version of that tree $t' \in T(\Sigma_X)$, by replacing some of its nodes s with a marked equivalent s_X. The marked alphabet Σ_X is defined as $\Sigma_X = \Sigma \cup \{s_X \mid s \in \Sigma\}$. A correctly marked tree (with regard to the extraction task) is defined as the single marked version of a tree in which all target nodes, and no others, are marked, while a partially correct marked tree requires that only some of the target nodes, and no others, are marked. We represent our wrapper as a language that accepts only partially correct marked trees. During extraction, a node is extracted if after marking that single node the resulting tree is accepted by the wrapper language. The wrapper is learned from examples that consist of the document tree with exactly one of the target nodes marked.

In [11], (k,l)-contextual tree languages are used for the correct marking acceptor. To enhance the generalization power of the algorithm, a preprocessing of the text nodes and a filtering of the forks is added. During the preprocessing step, the text nodes (except elements of the distinguishing context) are replaced by wildcards. The use of distinguishing contexts is optional, and the set of contexts is learned from the given examples. Only the marked forks are used; they provide the local context needed to decide whether a node should be extracted or not, while the other forks describe the general structure of the document. The latter is not needed as we assume that all documents for a given task are generated from the same template.

Example 4. In Figure 4.a we show the tree (only a subtree due to space restrictions) of the document from Figure 1, with the target fields indicated beneath. Only the 'City'

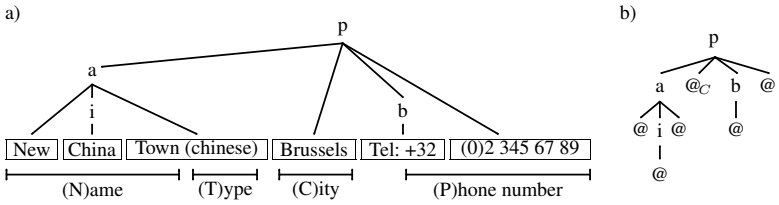


Fig. 4. a) A subtree of the document from Figure 1, containing the first restaurant. The different fields are indicated below the text leaves. b) The same subtree preprocessed, with the target node of the (C)ity field marked.

field can be extracted with the regular tree based approach, as it is the only one that occupies a single text node. Given the value 'Brussels' as example element of that field, we will mark the node containing 'Brussels', and perform the preprocessing step. The result (for the subtree of Figure 4.a) is shown in Figure 4.b. The learning algorithm collects now the (k, l)-forks that contain the marker from this tree. For $k = 2$ and $l = 1$, the resulting set is $\left\{ @_C \text{ , } \begin{array}{c} \text{p} \\ | \\ @_C \end{array} \right\}$, for $k = 2$ and $l = 2$, it is $\left\{ @_C \text{ , } \begin{array}{c} \text{p} \\ \text{a} \ @_C \end{array} \text{ , } \begin{array}{c} \text{p} \\ @_C \ \text{b} \end{array} \right\}$. The first wrapper will extract 'Brussels' and '(0)2 345 67 89' from the subtree in Figure 4.a, hence it is to general. The second wrapper extracts only 'Brussels' and is therefore a correct marking acceptor.

Another 'single node' field is the '(S)earch term'. A correct marking acceptor is obtained with the parameters $k = 1$ and $l = 3$: $\left\{ @_S \text{ , } \begin{array}{c} \text{i} \\ | \\ @_S \end{array} \text{ , } \begin{array}{c} \text{b} \\ | \\ \text{i} \\ | \\ @_S \end{array} \right\}$.

The induction algorithm is able to learn from positive examples only. But suitable values for the parameters k, l, and a boolean to turn the distinguishing contexts on or off have to be specified. Smaller values lead to more general acceptors, while larger values result in more specific ones. To obtain correct marking acceptors, while avoiding overfitting, one can search for the most general wrapper that rejects a set of given negative examples. In [12] an efficient algorithm is given to search through the parameter space.

All the above is integrated in an interactive system that starts induction from a single positive example. During the interaction, a user can apply the current wrapper on a document and can provide a negative example by selecting a false positive and a positive example by selecting a false negative. The wrapper is updated after each new example.

4 Hybrid Approach

This section describes how our tree based approach can be combined with a string based approach for extraction of fields that do not coincide with a single node.

4.1 Sub Node Fields

The term sub node field means that the text value from the field does not necessarily begin or end at a node boundary. The value can be a substring of a single text node, or could start and end in different nodes (the boundary nodes) in which case the value is the accumulation of the strings in the text nodes between the boundary nodes.

We define a spanning node for a given occurrence of a field as the first common ancestor of the two boundary nodes. In the case that the start node and end node are the same node, the spanning node is defined to be this node itself. These two cases are represented schematically respectively in Figure 5.a and b. We will refer to them as cases a and b. As can be seen in Figure 4, the 'Name' field is an example of case a, with as spanning node the 'a' node that spans over the start and end node. The 'Type' field in the same figure is an example of case b. Note that the boundary nodes are not necessarily at the same depth in the tree, as illustrated by the 'Phone' field. For this occurrence the spanning node is the 'p' node.

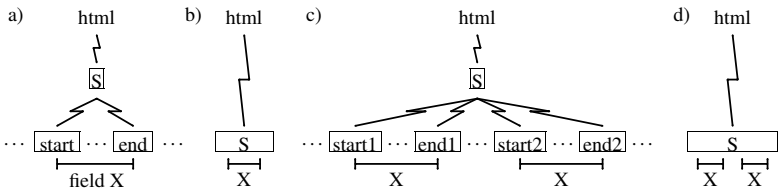


Fig. 5. Schematic representation of the different possible configurations in which an occurrence of a field can be found in a tree. The broken lines indicate an ancestor relation of one or more levels deep (The intermediate (irrelevant) nodes are left out).

In Example 1, all occurrences of a same field have different spanning nodes. It is possible though that different occurrences share the same spanning node. This case (case c) is represented in Figure 5.c. This case can also degenerate such that the boundary nodes and spanning node coincide. Hence multiple field values can be extracted from a single text node. This is illustrated in Figure 5.d, and we refer to it as case d.

4.2 Possible Approaches

We take two approaches to combine the tree based node extraction with a token sequence based subsequence extraction. A first approach is to extract (or learn to extract) the spanning node, and then extract (or learn to extract) the correct subsequence from the sequence obtained by flattening the subtree that starts at the spanning node. From this sequence we remove the initial (before the first text node) and trailing (after the last text node) mark up tags.

Example 5. In Example 2, the Name, Type, City, and Phone fields are all extracted from the sequence that is extracted for the Restaurant tuple (the previous level in the

hierarchy). In the first approach the spanning node 'a' is extracted, and sequence based extraction is then performed on the sequence defined by this spanning node:

```
New <i> China </i> Town ( chinese )
```

This sequence is smaller than in the hierarchical STALKER approach. The end rule *BackTo(*`'('`*)* suffices, as opposed to the *BackTo(*`</a>`*) BackTo(*`'('`*)* rule given in Example 2. For the Type field, the sequence is even smaller. The spanning node is a text node (case b). The City field simplifies to node extraction, no sequence extraction is needed. Only the Phone field, with spanning node 'p' will need sequence extraction from the same sequence as in the hierarchical STALKER approach. For the other fields, extraction and rule induction are performed on smaller sequences, leading to smaller and more correct rules, and a faster induction.

The second approach is to perform two node extraction tasks, one for the start node, and one for the end node. In a second step, the start boundary is retrieved from the start node, and the end boundary from the end node. Although two different sequences are used in this approach, these sequences are in general smaller than in the first approach.

On a case by case basis, we see that for case a, the second approach is better, because the sequence will perform better on smaller sequences. For case b, the second approach is overkill as there is no need to extract the same node twice. In case c and d, the first approach will not suffice with a single level sequence extraction. We need to use a limited hierarchical extraction that will do a list extraction of the multiple field values under the single spanning node. For case c, the second approach will be able to extract the different start and end nodes separately, and will not need a hierarchical sequence extraction. However, for extracting all targets in case d, the second approach also requires the use of hierarchical sequence extraction.

Overall, the second approach seems to be the preferable one. But having a look at real world extraction tasks, it turns out that Example 1, having two fields in case a, is a bit contrived. Indeed, the overwhelming majority of extraction tasks we looked at is either case b, or single node extraction without the need for sequence extraction. Extraction tasks situated in case c and d occur rarely. Hence the first approach is not too bad after all.

As the goal of the paper is to explore the viability of a hybrid scheme, it is sufficient to implement one approach and to compare it with the hierarchical STALKER system. We have chosen for the first approach, as it is a more straightforward extension to our existing system that is already able to extract a single node, so we only had to add a postprocessing step.

4.3 Interactive System

We have extended the system (that has a GUI), described in [12]. Instead of initially clicking on a single text node, the user selects a subsequence as the initial positive example. The system enters a loop in which it interacts with the user to improve the wrapper, until the user is satisfied. In each iteration, it induces a hypothesis based on the given positive and negative examples. The extraction results for the current document are visualized, and the user is invited to give counterexamples when the hypothesis is

not perfect. For false negatives, the user simply selects a *new positive example*. For false positives we distinguish two cases. Either an extraction is shown at a correct position, but the extraction itself is too big or too small. The user can then select the correct extraction, providing a *correction* to the system. Or the extraction is at a position without a target value in the neighborhood. The user can indicate this, providing a *new negative example*.

Internally, the spanning nodes of the example selections (both new positive examples and correction) are retrieved to find the set of positive node examples. The spanning nodes of the rejected extractions are collected to form the set of negative nodes. Based on these two sets, the induction algorithm for (k, l)-contextual languages, learns a set of parameters and the associated marked tree language for the node extraction.

Next, for a (new) positive example, the sequence under the spanning tree together with the selected field provides a (new) example for the STALKER induction algorithm.

Note that a new negative example requires only to learn again the extraction of the spanning node as the set of examples used by STALKER is preserved, given that STALKER only uses positive examples. A correction, on the other hand, will often not affect the position of the spanning node, in which case it only provides a new example for STALKER.

5 Experiments

In our experimental setup we want to compare the number of interactions by the user needed to learn a correct wrapper. For hierarchical STALKER this means that for every level the correct rules have to be learned. For every level the user has to give two initial examples, and extra corrections until no more mistakes can be found. For the hybrid approach, the user has to give a single initial example, and as many false positives, false negatives, and corrections as needed to learn a perfect wrapper. To simulate the user, we choose the annotated training set to find all mistakes, and take a random one to pass to the learning algorithm.

We use the WIEN data sets[3] for our comparison. We only used the sets that have a set of annotations included in the repository, and we left out some that were hard to represent in the STALKER embedded catalog formalism. Every data set has multiple fields. As we compare a single field extraction task. We split the tuple extraction task for every data set into several single field extraction tasks. Each task is referred to with the name of the original data set combined with the index of the field in the tuple. Some fields are contained in the 'href' attribute of an 'a' tag, or the 'src' attribute of an 'img' tag. In the tree based approach, the HTML-parser associates the attributes to the corresponding node. A trivial step can be added to retrieve these values. We decided to leave these tasks out, as they are skewed in favour of the tree based approach.

In Table 1 we show the averaged results of 30 runs on each data set. We give the induction time for the two approaches in column *ms*. For the hybrid approach we give the final k and l values, and the number of **P**ositive examples, **N**egative examples, and **C**orrections. For the hierarchical STALKER approach, we show the number of **P**ositive

[3] These are available at the RISE repository: http://www.isi.edu/info-agents/RISE/index.html.

examples, split over the different levels (starting on left with the top level). When we compare the total number of interactions (P+N+C and P summed over all levels), it is clear that the hybrid approach requires substantially less user interactions. The sequence extraction step in the hybrid approach, and the extraction in the final level of the STALKER approach extract the same text value. When we compare the number of positive examples needed to learn this last extraction (P+C compared with the last number in P), we see that this number is again smaller for the hybrid approach. This is because the tree based approach returns a much smaller sequence to extract from, as illustrated in Example 5.

Table 1. Comparison of the interactions needed to learn a perfect wrapper, between our hybrid approach, and the a sequence based approach (STALKER).

Data set	Hybrid				STALKER	
	P/N/C	k	l	ms	P	ms
s1-1	1/1/0	1	3	18	3/72.1/2.9	4442
s1-3	4/1.8/0	3	3	612	3/60.6/6.9	3651
s3-2	1/1/0	1	3	10	2.9/2.1/2.2	460
s3-3	1/0/0	1	2	3	2.5/2.1/2.3	316
s3-4	1/0/1.2	1	2	6	2.8/2.1/3	394
s3-5	1/0/4.9	1	2	48	2.8/2.1/5.4	554
s3-6	1/0/3.4	1	2	7	2.9/2/6.1	27520
s4-1	1/0/0	1	2	2	4.5/2.2/2.3	1136896
s4-2	1/1/0	2	3	33	4.7/3/2.1	1240828
s4-3	1/1/1	2	3	23	4.7/2.2/2	1420509
s4-4	1/1/1	2	3	22	4.8/2.7/2	1333724
s5-2	1/1/0	1	4	18	2.8/3.7/2.9	1136
s8-2	1/1/0	1	3	12	2.4/2/2.6	785
s8-3	1/1.2/0	2	3	39	2.3/2.1/2.9	675
s12-2	2/1.4/0	1	4	36	2.7/80.6/2.3	1394
s14-1	1.1/1/0.9	2	2	21	2/2.3/2.3	21
s14-3	1/0/0	1	2	3	2/2.2/2.4	21
s15-2	1/0/0	1	2	2	2.9/2.1/2.1	5
s19-2	1/1/0	2	2	13	2/2.1/2.1	85
s19-4	1/1/0	1	3	7	2/2/2	85

Data set	Hybrid				STALKER	
	P/N/C	k	l	ms	P	ms
s20-3	1/0/0	1	2	3	2/2/2.1	155
s20-4	1/1.4/0	2	3	192	2/2/3	165
s20-5	1/1.8/0	2	3	1067	2/2/2	158
s20-6	1/1.4/0	2	3	41	2/2/3.1	159
s23-1	1/1/0	2	3	35	2.6/3.1/4.1	606
s23-3	1/1/0	1	3	11	2.6/2.6/3.4	602
s25-2	1/1/0	1	3	5	2.6/5.1/3.0	82
s27-1	1/1.6/1	2	6	195	2.7/2.4	44
s27-2	1/1.3/1	2	6	190	2.7/2.8	46
s27-3	1/1.4/1	2	6	235	2.7/2.8	52
s27-4	1/1.2/1	2	6	348	2.6/6.7	3430
s27-5	1/1/1	2	5	125	2.7/2.5	33
s27-6	1/1/0	2	5	101	2.9/2.3	44
s30-2	2/1/0.7	1	3	14	2/2.8/2.6	8
s30-3	2/1/0	1	3	14	2/3/2	8
s30-4	2/1/0	2	2	50	2/3.3/2.5	12
s30-5	2/1/0.2	2	2	26	2/2.7/2.1	7
s30-6	2/1/0	2	2	49	2/3.1/2.8	9
s30-7	2/1/0	2	2	28	2/2.5/2.1	6
s30-8	2/1/0	2	2	25	2/2.6/2.6	6

6 Related Work

Another approach that allows to combine sequence based and tree based methods is described in [7]. A (set covering) meta learning algorithm runs the learning algorithms of different wrapper modules, evaluates their results and chooses the best resulting rules to add to the final solution. Some of these modules are defined to combine other modules to allow conjunctions or a multi level approach like ours. In contrast to our approach, the algorithm requires completely annotated documents (or at least a completely annotated part of the document).

7 Conclusion

Tree based methods have been shown to have a favorable performance over string based ones, on complete node extraction tasks, but have as limitation that they cannot extract values that have boundaries within text nodes [11]. In this paper we show that these methods are viable, when used as a first step, in a hybrid two step approach. We have shown that the tree based approach does a good job of narrowing down the sub node extraction task presented to the string based second step. This results in substantially faster learning while requiring substantially less user interactions.

The hybrid approach as presented in this paper will extract only a single field, instead of n-tuples (in contrast, STALKER can extract n-tuples as long as they can be represented in the embedded catalog formalism). Also in [11] we argue in a section about further work that it is more flexible to add a tuple aggregation procedure on top of a single field extraction approach. A practical approach and an empirical proof of the viability of this last approach still remains for further work.

References

1. M. E. Califf and R. J. Mooney. Relational learning of pattern-match rules for information extraction. In *AAAI/IAAI*, pages 328–334, 1999.
2. B. Chidlovskii, J. Ragetli, and M. de Rijke. Wrapper generation via grammar induction. In *Proc. 11th European Conference on Machine Learning (ECML)*, volume 1810, pages 96–108. Springer, Berlin, 2000.
3. D. Freitag. Information extraction from HTML: Application of a general machine learning approach. In *AAAI/IAAI*, pages 517–523, 1998.
4. D. Freitag and N. Kushmerick. Boosted wrapper induction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Innovative Applications of AI Conference*, pages 577–583. AAAI Press, 2000.
5. D. Freitag and A. McCallum. Information extraction with HMMs and shrinkage. In *AAAI-99 Workshop on Machine Learning for Information Extraction*, 1999.
6. C.-N. Hsu and M.-T. Dung. Generating finite-state transducers for semi-structured data extraction from the web. *Information Systems*, 23(8):521–538, 1998.
7. L. S. Jensen and W. W. Cohen. A structured wrapper induction system for extracting information from semi-structured documents. In *Proc. of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*, 2001.
8. R. Kosala, M. Bruynooghe, H. Blockeel, and J. V. den Bussche. Information extraction from web documents based on local unranked tree automaton inference. In *Intl. Joint Conference on Artificial Intelligence (IJCAI)*, pages 403–408, 2003.
9. N. Kushmerick, D. S. Weld, and R. B. Doorenbos. Wrapper induction for information extraction. In *Intl. Joint Conference on Artificial Intelligence (IJCAI)*, pages 729–737, 1997.
10. I. Muslea, S. Minton, and C. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Journal of Autonomous Agents and Multi-Agent Systems*, 4:93–114, 2001.
11. S. Raeymaekers, M. Bruynooghe, and J. V. den Bussche. Learning (k,l)-contextual tree languages for information extraction. Submitted to Machine Learning Journal.
12. S. Raeymaekers, M. Bruynooghe, and J. V. den Bussche. Learning (k, l)-contextual tree languages for information extraction. In *ECML*, pages 305–316, 2005.
13. S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272, 1999.

Author Index