Gerson Zaverucha Vítor Santos Costa
Aline Paes (Eds.)

ILP 2013
Late Breaking Papers

Late Breaking Papers of the 23rd International Conference on Inductive Logic Programming (LBP-ILP)

Rio de Janeiro, Brazil, August 28th to 30th, 2013.

CEUR-WS.org/Vol-1187

http://ceur-ws.org/Vol-1187/.

urn:nbn:de:0074-1187-5

Editors' addresses:

Gerson Zaverucha

COPPE - Federal University of Rio de Janeiro
Department of Computer Science and Systems Engineering
Caixa Postal 68511
Rio de Janeiro, RJ, Brazil
21941-972

gerson@cos.ufrj.br

Vítor Santos Costa

Department of Computer Science
Faculty of Sciences
University of Porto
Rua do Campo Alegre, 1021/1055
4169 - 007 PORTO
Portugal

vsc@dcc.fc.up.pt

Aline Paes

Fluminense Federal University
Institute of Computer Science
Rua Passo da Pátria, 156, Bloco E, 3o andar, São Domingos, Niterói, RJ, Brazil,
24210-240

alinepaes@ic.uff.br

Preface

This volume contains the *Late Breaking Papers* of *ILP 2013: the 23rd International Conference on Inductive Logic Programming* held on August 28-30, 2014 in Rio de Janeiro, Brazil. The ILP conference series, started in 1991, is the premier international forum on learning from structured data. Originally focusing on the induction of logic programs, it broadened its scope and attracted a lot of attention and interest in recent years. The conference now focuses on all aspects of learning in logic, multi-relational learning and data mining, statistical relational learning, graph and tree mining, relational reinforcement learning, and other forms of learning from structured data.

This edition of the conference solicited three types of submissions:

1. long papers (12 pages) describing original mature work containing appropriate experimental evaluation and/or representing a self-contained theoretical contribution.
2. short papers (6 pages) describing original work in progress, brief accounts of original ideas without conclusive experimental evaluation, and other relevant work of potentially high scientific interest but not yet qualifying for the above category.
3. papers relevant to the conference topics and recently published or accepted for publication by a first-class conference such as ECML/PKDD, ICML, KDD, ICDM, etc., or journals such as MLJ, DMKD, JMLR, etc.

We received 42 submissions, 18 long, 21 short submissions, and 3 previously published papers. Each submission was reviewed by at least 3 program committee members. The short papers were evaluated on the basis of both the submitted manuscript and the presentation at the conference. Accepted papers presenting work in progress, i.e., reports on ongoing research are collected in this volume.

The conference program included 3 invited talks. Professor Jure Leskovec introduced ongoing work on *Exploring the Structure of On-Line Networks and Communities*. Social interactions of hundreds of millions of people on the Web create massive digital traces, which can naturally be represented, studied and analyzed as massive networks of interactions. By computationally analyzing such network data we can study phenomena that were once essentially invisible to us: the social interactions and collective behavior of hundreds of millions of people. In his talk he discussed how computational perspectives and mathematical models can be developed to abstract online social phenomena like: How will a community or a social network evolve in the future? What are emerging ideas and trends in the network? How does information flow and mutate as it is passed from a node to node like an epidemic?

Professor Hendrik Blockeel discussed *Lifted variable elimination: faster correct inference in probabilistic-logical models*. He started from an intriguing observation, that first-order logic allows inference on the level of variables, that is, we can reason about an object's properties without knowing the object. This boosts inference efficiency. It is not yet clear to what extent probabilistic inference can, similarly, be "lifted" to the level of logical variables. In recent years, many results have been obtained that contribute towards solving this

question. A number of them were discussed in his talk, focusing on intuition rather than technical detail. He discussed how variable elimination, perhaps the simplest approach to probabilistic inference, can be lifted by identifying and exploiting particular kinds of symmetry in a probabilistic-logical model. He also discussed a number of theoretical and experimental results, both positive and negative, that provide insight into the circumstances under which lifting is (not) possible.

Professor William W. Cohen discussed *Learning to Construct and Reason with a Large Knowledge Base of Extracted Information*. Carnegie Mellon University's "Never Ending Language Learner" (NELL) has been running for over three years, and has automatically extracted from the web millions of facts concerning hundreds of thousands of entities and thousands of concepts. NELL works by coupling together many interrelated large-scale semi-supervised learning problems. In this talk he discussed some of the technical problems the group encountered in building NELL, and some of the issues involved in reasoning with this sort of large, diverse, and imperfect knowledge base. Professor Cohen presented joint work with Tom Mitchell, Ni Lao, William Wang, and many other colleagues.

The General Chair was Gerson Zaverucha, the Program Chairs were Gerson Zaverucha and Vítor Santos Costa, and the Local Chair was Aline Paes. We would like to thank the guest speakers for coming to ILP'13 and for their availability during the Conference. The conference was kindly sponsored by FAPERJ, the Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro through grant E-26/101.541/2010. The Universidade Federal do Rio de Janeiro (UFRJ) generously supported ILP'13 by allowing us to use the conference venue, Casa da Ciência. We would like to thank its helpful staff: Camila Costa, Angela Monteiro and Claudia Pereira. We would like to thank Maria de Fatima Cruz Marques for her valuable suggestions. Vítor Santos Costa was supported by the grant SIBILA, NORTE-07-0124-FEDER-000059, and the FCT grants ADE, PTDC/EIA-EIA/121686/2010, and ABLe, PTDC/EEI-SII/2094/2012 (FCOMP-01-0124-FEDER-029010). The Machine Learning journal supported research in this area by opening a special issue on ILP'13. Springer Verlag will publish the ILP'13 main proceedings, and CEUR is publishing the late breaking papers. We would like to thank Easychair.org for supporting submission handling. Last, but not least, we would like to thank the Local Organizing Committee: Kate Revoredo and Fernanda Baião helped throughout in the organization, and Roosevelt Sardinha created and maintained the web-site.

June, 2014
Rio de Janeiro and Porto

Gerson Zaverucha
Vítor Santos Costa
Aline Paes

Organization

General Chair

Gerson Zaverucha	COPPE – Universidade Federal do Rio de Janeiro

Program Chairs

Gerson Zaverucha	COPPE – Universidade Federal do Rio de Janeiro
Vítor Santos Costa	CRACS/INESC-TEC & DCC-FCUP

Local Chair

Aline Paes	Universidade Federal Fluminense

Local Organizing Committee

Kate Revoredo	Universidade Federal do Estado do Rio de Janeiro
Fernanda Baião	Universidade Federal do Estado do Rio de Janeiro
Roosevelt Sardinha	COPPE - Universidade Federal do Rio de Janeiro

Program Committee

Erick Alphonse	LIPN - UMR CNRS 7030
Annalisa Appice	Dipartimento di Informatica, Università di Bari
Hendrik Blockeel	K.U. Leuven
Ivan Bratko	University of Ljubljana
Rui Camacho	LIACC/FEUP University of Porto
James Cussens	University of York
Luc De Raedt	Katholieke Universiteit Leuven
Saso Dzeroski	Jozef Stefan Institute
Nicola Fanizzi	Dipartimento di Informatica, Università di Bari
Stefano Ferilli	Università di Bari
Peter Flach	University of Bristol
Nuno Fonseca	CRACS-INESC Porto LA & EMBL-EBI
Paolo Frasconi	Università degli Studi di Firenze

Tamas Horvath	University of Bonn and Fraunhofer IAIS
Katsumi Inoue	NII, National Institute of Informatics
Nobuhiro Inuzuka	Nagoya Institute of Technology
Andreas Karwath	University of Mainz
Kristian Kersting	University of Dortmund
Ross King	University of Manchester
Ekaterina Komendantskaya	School of Computing, University of Dundee
Stefan Kramer	University of Mainz
Nada Lavrač	Jozef Stefan Institute
Francesca Alessandra Lisi	Università degli Studi di Bari "Aldo Moro"
Donato Malerba	Dipartimento di Informatica, Università di Bari
Stephen Muggleton	Department of Computing, Imperial College London
Sriraam Natarajan	University of Indiana
Ramon Otero	University of A Coruña
Aline Paes	Universidade Federal Fluminense
C. David Page	University of Wisconsin – Madison
Bernhard Pfahringer	University of Waikato
Ganesh Ramakrishnan	IIT Bombay
Jan Ramon	K.U.Leuven
Oliver Ray	University of Bristol
Fabrizio Riguzzi	University of Ferrara
Celine Rouveirol	LIPN, Université Paris 13
Chiaki Sakama	Wakayama University
Claude Sammut	University of New South Wales
Jude Shavlik	University of Wisconsin – Madison
Takayoshi Shoudai	Department of Informatics, Kyushu University
Ashwin Srinivasan	IBM India Research Lab
Alireza Tamaddoni-Nezhad	Imperial College, London
Tomoyuki Uchida	Hiroshima City University
Christel Vrain	LIFO - University of Orléans
Stefan Wrobel	Fraunhofer IAIS & Univ. of Bonn
Akihiro Yamamoto	Kyoto University
Filip Zelezny	Czech Technical University

Additional Reviewers

B

Bellodi, Elena
H

Heras, Jonathan
M

Manine, Alain-Pierre
S

Sato, Taisuke

Contents

Incremental Construction of Complex Aggregates: Counting over a Secondary Table

Clément Charnay[1], Nicolas Lachiche[1], and Agnès Braud[1]

ICube, Université de Strasbourg, CNRS
300 Bd Sébastien Brant - CS 10413, F-67412 Illkirch Cedex
{charnay,nicolas.lachiche,agnes.braud}@unistra.fr

Abstract. In this paper, we discuss the integration of complex aggregates in the construction of logical decision trees. We review the use of complex aggregates in TILDE, which is based on an exhaustive search in the complex aggregate space. As opposed to such a combinatorial search, we introduce a hill-climbing approach to build complex aggregates incrementally.

1 Introduction and Context

Relational data mining deals with data represented by several tables. We focus on the typical setting where one table, the primary table, contains the target column, *i.e.* the attribute whose value is to be predicted, and has a one-to-many relationship with a secondary table. A possible way of handling such relationships is to use complex aggregates, *i.e.* to aggregate the objects of the secondary table which meet a given condition, using an aggregate function on the objects themselves (*count* function) or on a numerical attribute of the objects (*e.g. max*, *average* functions). For instance, we may want to classify molecules. Molecules have atoms. They can be represented as a table of molecules and a table of atoms, with a foreign key in the table of atoms indicating the molecule it belongs to. Then, the class of a molecule may depend on the comparison between the average of the charge of the carbon atoms of the molecule and some threshold value. This example also shows what a complex aggregate relies on: an aggregate function (here the *average*), a condition to select the objects to aggregate (here we select only the *carbon* atoms), the attribute to aggregate on (here the charge of the atoms), an operator and a threshold to make a comparison with the result of the aggregation.

Previous work showed that the expressivity of complex aggregates can be useful to solve problems such as urban blocks classification. [1,2] introduced complex aggregates in propositionalisation. But their use increases the size of the feature space too much, and they cannot be fully handled. This is the reason why we focus on introducing them in the learning step. To our knowledge, only one relational learner, TILDE [3], has implemented complex aggregates, but its exhaustive approach is not adapted to too complex problems. The main motivation for our work is to elaborate new heuristics to handle complex aggregates in

relational models. This article presents a logical decision tree learner which uses complex aggregates to deal with secondary tables, using a hill-climbing heuristic to build them incrementally. We introduce this heuristic in the context of logical decision trees, but it could be applied to other approaches. In this article, we focus on counting over a secondary table, *i.e.* the aggregate function considered will be the *count* function.

The rest of this paper is organized as follows. In Sect. 2, we review the use of complex aggregates in TILDE. In Sect. 3, we describe our heuristic to explore the complex aggregate space. Finally, in Sect. 4, we detail future work.

2 TILDE and Complex Aggregates

TILDE [3] is a first-order decision tree learner. It uses a top-down approach to choose, node by node, from root to leaves, the best refinement to split the training examples according to their class values, using gain ratio as a metric to guide the search. TILDE relies on a language bias: the user specifies the literals which can be added in the conjunction at a node. In this relational context, to deal with secondary tables, the initial version of TILDE introduces new variables from these secondary tables using an existential quantifier.

Then, TILDE has been extended [4] to allow the use of complex aggregates, and a heuristic has been developed to explore the search space [5]. This heuristic is based on the idea of a refinement cube, where refinement space for the aggregate condition, aggregate function and threshold are the dimensions of the cube. This cube is explored in a general-to-specific way, using monotone paths along the different dimensions: when a complex aggregate (a point in the refinement cube) is too specific (*i.e.* it fails for every training example), the search does not restart from this point.

However, the implementation does not allow more than two conjuncts in the aggregate query "due to memory problems" [6, p. 32], which limits the search space. Numerical attributes are handled by a comparison to a threshold in problems such as geographical ones. It is possible to discretize numerical attributes beforehand and to define predicates to make those comparisons between the values of the attributes and the thresholds given by the discretization. Nevertheless, enumerating all the combinations of thresholds over all the numerical attributes takes space in memory, and hence the approach is not tractable. To summarize, this combinatorial approach which handles complex aggregates in TILDE has limitations. We intend to overcome these limitations by not trying to explore the search space exhaustively, but by finding a heuristic to explore the search space and to build complex aggregates incrementally.

3 Incremental Construction of Complex Aggregates with Hill-Climbing

Our goal is to build a logical decision tree, like TILDE, which uses complex aggregates to deal with secondary tables. To explore the refinement cube of com-

plex aggregates, we choose to use a hill-climbing method. This section details the method. We use the general notation "$function(condition)\{operator\}threshold$" to refer to a complex aggregate.

3.1 Refinement of Complex Aggregates

When testing an aggregate, we make a refinement to find the best aggregate, and a hill-climbing is performed from a starting aggregate. In this paper, we limit ourselves to the *count* function to aggregate secondary tables. The starting conditions are detailed below. We then try to refine it with hill-climbing, allowing a non-strict climbing: if there is no strictly improving refinement, we allow picking a refinement with the same score as in the previous step, if it has not been visited before. To achieve that, we store all previous refinements chosen in the hill-climbing path. From the current aggregate, there are several ways to modify it:

- Firstly, given the examples, we compute the possible results of the aggregate function, which will serve as possible thresholds. To these values, we add one threshold depending on the operator: strictly lower than the other values if the operator is $\leq$ (so that the complex aggregate is true for none of the examples) and strictly higher if the operator is $\geq$. Such thresholds are chosen to be respectively MAX_DOUBLE and its opposite. The current threshold is then set to the closest possible threshold if it is lower than the minimum or higher than the maximum of the possible thresholds. For instance, if the current refinement to try is *"the count of atoms in the molecule is less than or equal to MAX_DOUBLE"* and, in the training set, there are between 13 and 42 atoms in a molecule, the threshold will be immediately set to 42.
- Then, we can refine the aggregate by increasing or decreasing the threshold (among the possible thresholds).
- Other possibilities are to remove a literal from the aggregate condition, or to add one.

The Starting Conditions Given this, if we refer to the maximum threshold as max, 2 starting conditions will be $count(true) \leq max$ and $count(true) \geq MAX\_DOUBLE$. From the point of view of the examples considered, they are opposite: if one succeeds for an example, the other will fail. The former is the most general (*i.e.* it succeds for every example), the latter the most specific. We then observe that refinements for one will have the same effect on information gain for the other (the final branch of the examples will be inverted between both), so on the training set, they are equivalent, and will be refined equivalently. In the end, we get two conditions $count(condition) \leq someThreshold$ and $count(condition) \geq nextThreshold$ which are opposite. To conclude on this point, considering both starting conditions is not necessary, since they will be refined following similar paths and will yield the same information gain after the hill-climbing process. Of course, the same reasoning applies for starting conditions $count(true) \leq -MAX\_DOUBLE$ and $count(true) \geq min$, this is the

reason why we consider only two starting conditions, arbitrarily with the operator $\geq$.

Moving in the Complex Aggregate Space We now discuss our method for refinement of the aggregate. If we add a literal to the aggregate condition, it will yield a specialization and less objects will be selected. The threshold range discussed above will not be the same, and the current threshold, associated to the previous, more general condition, will not be relevant since it may be too high. Hence, the refinement will yield a poor gain ratio and will not be chosen. For instance, in the training set, there are between 13 and 42 atoms in a molecule, but only between 5 and 20 carbon atoms. If the current aggregate states that "the count of atoms is less than or equal to 30", and we try to refine it to "the count of carbon atoms is less than or equal to 30", this last aggregate will be true for every example, yielding zero gain, and hence will not be chosen. Of course, the problem will be similar if we consider the other way, *i.e.* if we drop a literal from the aggregate condition, yielding a generalization.

To avoid modifying the aggregate condition without modifying the threshold, we do as follows. When modifying the aggregate condition, we consider the number of possible thresholds n_1 given by the current aggregate condition, and the number of thresholds n_2 given by the next (after modification) aggregate condition. We sort those two sets those thresholds in increasing order, such that the current threshold is in position c_1 with indices going from 0 to $n_1 - 1$, the next threshold chosen, in position c_2 between 0 and $n_2 - 1$ will be picked such that $\frac{c_2}{n_2-1}$ is closest to $\frac{c_1}{n_1-1}$. Mathematically: $c_2 = round(\frac{c_1 \cdot (n_2-1)}{n_1-1})$. Since we add a threshold to a list which already contains at least one element, there are always at least two possible thresholds and hence the case $n_1 = 1$ is not an issue.

3.2 Dealing with Empty Sets

We finally discuss a problem that will occur with aggregate functions other than *count*: the computation of a value for empty sets. Indeed, an aggregate condition might select no object, and aggregation over a numerical attribute is not possible in this case. For instance, how can the mean of the charge of the oxygen atoms in a molecule be computed when the molecule does not have any oxygen atom? Only the *count* function can deal in a natural way with empty sets, while another solution has to be chosen for numerical aggregate functions. Some possibilities to deal with this issue have been discussed in [7]:

- Fixing an arbitrary value as the result.
- Using a value depending on the aggregate condition, as close as possible to the values for examples for which the aggregate condition does not result in an empty set, or as far as possible.
- Failing the aggregate when the aggregate function cannot be applied.
- Discarding the aggregate from being chosen as a refinement if the aggregate function cannot be applied for at least one example.

In our opinion, the two first options are not easy to apply for every function. Indeed, for functions *min* or *max*, one can choose values as low or as high as possible so that the aggregate always succeeds or fails if the function cannot be applied directly. But for the *average* function, using a fixed value such as 0 is not relevant if the attribute can take both positive and negative values, neither is choosing positive or negative infinity. Moreover, the fact that the set to aggregate is empty can be meaningful, and by assigning a result to the aggregate function we lose this significance. The third option also gives to the empty set a meaning we do not necessarily want it to have: the failure of the aggregate should mean the inequality between the result of the aggregation and the threshold is wrong. However, this is not the meaning of the empty set. Actually, it is implied by the failure of the existential quantifier for the aggregate condition, *i.e.* $count(condition) \geq 1$ fails.

Then we see two ways to address the issue of empty sets: firstly to consider them as a third branch in our decision trees, since they do not correspond to a success or a failure of the inequality, they are a third possibility. However, this option breaks the binary structure of the tree, which is not necessarily a problem. Nevertheless, we present another option to preserve the binary tree structure: the principle is to create a node with the $count(condition) \geq 1$ aggregate before adding a node with an aggregate with a function which may not be applicable. In the left branch, we can then add the aggregate $function(condition) \geq threshold$ without the empty set problem, since we know from the parent node that *condition* will select at least one object. This adapts the three-branch idea to preserve the binary tree structure, using two nodes instead of one. An example is shown in Fig. 1, where the aggregate *"the average charge of the carbon atoms in the molecule is greater than or equal to 0.542"* is "protected" by the existential quantifier which tests the presence of at least one carbon atom in the molecule, *i.e.* the aggregate *"the count of carbon atoms in the molecule is greater than or equal to 1"*. If the latter succeeds, then the former can be evaluated because it is meaningful to compute the average value of a non-empty set. If the existential quantifier fails, then the average is not computed because it would be meaningless.

4 Conclusion and Future Work

The method described in Sect. 3 is implemented and will be evaluated. The next step in this work is to consider other aggregate functions, on numerical attributes of the secondary objects, and to allow the change of the aggregate function in the refining process of the aggregates, by taking advantage of their ordering as in [5]. Then, another step will be to allow recursivity in the aggregates, *i.e.* create complex aggregates which have complex aggregates in their aggregate condition, to use the whole database. However, this will inevitably raise new issues, since this will add other levels of refinements. A more complex problem will be to handle many-to-many relationships, since the complex aggregates can

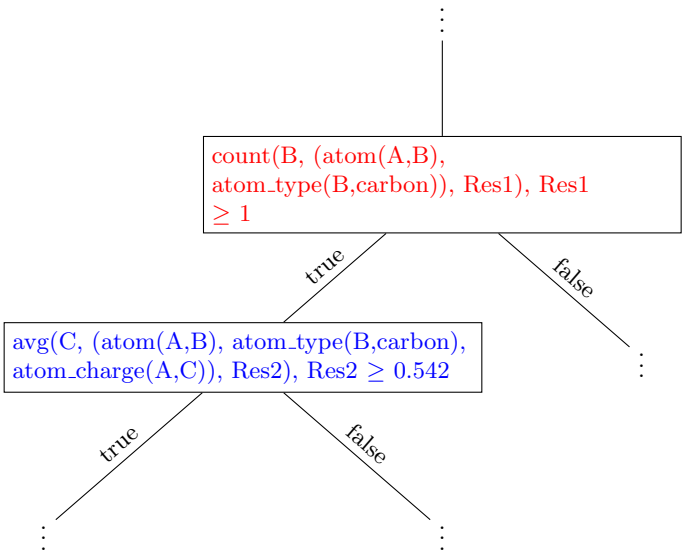


Fig. 1. Example of three-branch structure to deal with empty sets.

be formed both ways with such relationships, which can possibly lead to loops in the recursivity discussed above.

References

1. El Jelali, S., Braud, A., Lachiche, N.: Propositionalisation of continuous attributes beyond simple aggregation. In Riguzzi, F., Zelezný, F., eds.: ILP. Volume 7842 of Lecture Notes in Computer Science., Springer (2012) 32–44
2. Puissant, A., Lachiche, N., Skupinski, G., Braud, A., Perret, J., Mas, A.: Classification et évolution des tissus urbains à partir de données vectorielles. Revue Internationale de Géomatique **21**(4) (2011) 513–532
3. Blockeel, H., Raedt, L.D.: Top-down induction of first-order logical decision trees. Artif. Intell. **101**(1-2) (1998) 285–297
4. Assche, A.V., Vens, C., Blockeel, H., Dzeroski, S.: First order random forests: Learning relational classifiers with complex aggregates. Machine Learning **64**(1-3) (2006) 149–182
5. Vens, C., Ramon, J., Blockeel, H.: Refining aggregate conditions in relational learning. In Fürnkranz, J., Scheffer, T., Spiliopoulou, M., eds.: PKDD. Volume 4213 of Lecture Notes in Computer Science., Springer (2006) 383–394
6. Blockeel, H., Dehaspe, L., Ramon, J., Struyf, J., Assche, A.V., Vens, C., Fierens, D.: The ACE Data Mining System. (March 2009)
7. Vens, C.: Complex aggregates in relational learning. PhD thesis, Informatics Section, Department of Computer Science, Faculty of Engineering Science (March 2007) Blockeel, Hendrik (supervisor).

Modeling of Resilient Systems in Default Logic

Andrei Doncescu[1]
[1] LAAS-CNRS/University of Toulouse
Toulouse France

andrei.doncescu@laas.fr

Abstract. In this paper we propose a reconfiguration model based on switched flat system. The interest to have flat subsystems is to assure the property of transitivity. Transitivity is one the key points of a resilient system keeping the dependability. To reconfigure the system in the case of unexpected phenomena we use default logic.

1 Introduction

In this paper we present a reconfigurable model of resilient systems. The reconfiguration is an important method of resilient systems keeping stability. This approach could be applied to a large category of systems having a nonlinear dynamic, from biological systems to robots and aircraft. What characterizes all these systems is the high complexity. The increasing complexity makes systems more and more vulnerable for faults and chaotic behavior. The system state may either evolve continuously for some duration of time according to one set of differential equations or be abruptly reset to a new value from which evolution is governed by another set of differential equations. The commutations are typically triggered by the occurrence of some discrete event.
During the last decades the adjective Resilient has been used for labeling the systems, which are faults tolerant but ignoring the unexpected aspect of the phenomena that the systems have to face, therefore the necessity of a fault-diagnosis and fault-tolerant control. Monitoring and diagnosis of any resilient system depend on the ability to estimate the system state given the observations. Estimation for hybrid systems is particularly challenging because it requires keeping track of multiple models and the transitions between them.
The different approaches are related to the a priori representation of the knowledge. The physical models basically represented by differential equations “mime” physical structure and give a synoptic view. The engineering aspect is defined by functional models, which describe the chain of functions realized by the system. The representation of knowledge about the system leads to other type of models: informational, which are supposed to gather signals and find out the relations causality/effects.
Our viewpoint is all complex or resilient systems could be modeled by hybrid dynamical subsystems. Therefore the state may either evolve continuously for some

duration of time according to one set of differential equations or be abruptly reset to a new value from which evolution is governed by another set of differential equations, with the switches typically triggered by the occurrence of some discrete event, therefore the signal abstraction could be very useful. Two types of data exist in generic databases describing the hybrid systems: numerical and symbolic.
In the case of hybrid dynamic systems the quantity of data describing the evolution of the complex system can be very important and difficult to figure out the analytical model therefore a supervised learning model seems to be the only solution.
We point out the problem of discretization, which influences the results either by an over fitting (i.e. finding meaningless regularity in data due to a large number of possible hypotheses) or by missing important events.
The resilience is the property of a complex system to successfully recover environmental perturbations or disturbances. Contrary, of the feeling that stability is a property of resilient systems, resilience is one of the properties of stable dynamic systems.
The misunderstandings and problems that continue to occur will eventually cause fatal damage to the system must be avoid by the construction or modeling of resilient systems.
The notion of resilience has been introduced in different fields:

1. in ecology [4], referring to moving from a stability domain to another one under the influence of disturbances;
2. in business [5], referring to the capacity to reinvent a business model before circumstances force to;
3. in industrial safety [6], referring to anticipating risk changes before damage occurrence.

Our definition of resilience is:
"The capacity of a complex system to react in presence of disturbances by switching from one dynamical model to another one by keeping the global stability properties".

The main idea of flatness is to connect the different subsystems in a new configuration.

2. Flat Systems

There is a three-step process for describing equations of physics that is often helpful in clarifying the distinction between different types of ideas. The first step is to describe the kinematics of the process, i.e. the basic variables in the problem and the physically inherent restrictions of them. Next, one poses universal laws that govern all processes of the type under consideration. Finally, one postulates constitutive laws that differentiate one physical situation from another.
In the case of resilient systems we should be able to determine the state of the system and to control it from the outputs. A special type of systems named flat satisfies this request. Intuitively, a system is said to be differentially flat if a set of variables called

flat outputs can be found for which all states and actions can be determined from them without integration.

A general nonlinear system given by :

$$\dot{\underline{X}} = F(\underline{X},\underline{U}),\ \underline{X}\in\mathbf{R}^n,\ \underline{U}\in\mathbf{R}^m, \qquad \text{(A-1)}$$

where is a smooth mapping, is said explicitly flat with respect to the output vector , if is an order vector which can be expressed analytically as a function of the current state, the current input and its derivatives, while the state and the input vectors can be expressed analytically as a function of and a finite number of its derivatives. Then there exists smooth mappings G_X, G_U, and G_Z such as:

$$\begin{cases} \underline{Z} = G_Z\left(\underline{X},\underline{U},\ldots,\underline{U}^{(n_z)}\right) & \text{A-2} \\ \underline{X} = G_X\left(\underline{Z},\dot{\underline{Z}},\ldots,\underline{Z}^{(n_x)}\right) & \text{A-3} \\ \underline{U} = G_U\left(\underline{Z},\underline{Z},\ldots,\underline{Z}^{(n_x+1)}\right) & \text{A-4} \end{cases}$$

where n_z and n_x are integer numbers. Vector $\underline{Z}$ is called a flat output for the nonlinear system. There is no systematical way to determine flat outputs and eventually to prove its uniqueness, the flat outputs usually possess some physical meaning.

The explicit flatness property is of particular interest for the solution of control problems when physically meaningful flat outputs can be related with their objectives. In many situations, the control problem can be formulated as a flat output trajectory following problem. In general, for these cases, the flat output of equation (A-2) can be reduced, through state transformation, to a function of a single argument, the new system state itself:

$$\underline{Z} = G_Z\left(\underline{X}\right) \qquad \text{A-5}$$

We would like to make the dissociation between resilience and stability: it is noted that “a system can be very resilient and still fluctuate greatly, i.e., have high stability” and that “high stability seems to introduce high resilience”;

2 Modeling of Switched Systems

We have considered in this models that “Switched systems are more than the sum of their subsystems”, which is the most important property of complex and resilient systems. A switched systems is represented:

$V = U \cup Y \cup X$: Input, output, and internal (state) variables

Q: States, a set of valuations of X

$\Theta \subseteq Q$: Start states

$A = I \cup O \cup H$: Input, output, and internal actions

$D \subseteq Q \times A \times Q$: Discrete transitions

T: Trajectories for V.

3 Causality and Classical Inference

If the inference of classical logic A → B or A ⊢ B is fully described formally, with all the "good" logic properties (tautology, not contradiction, transitivity, contraposition, modus ponens, ...), a description of the properties of causality is not simple. Causality cannot be seen as a classical logic relation.
A basic example is "If it rains the grass is wet". This expression cannot be translated by the formula *Rain → lawn-wet*, which means if *it rains the grass is always wet*. Indeed, there may be exceptions to this rule (the lawn is under a shed ...). You can also change the environment (we cover the lawn).
The rules with exceptions are well known in Artificial Intelligence. They drive, in particular, to nonmonotonic logics and revision theories. On the other hand and more technical, we find here all the classic problems that arise when one wants to try to formalize and use of negation by failure in programming languages such Solar [3]. In this paper we describe a very simple and efficient form of causality necessary and probably sufficient for the application to complex and resilient systems.
To describe interactions between subsystems we use a language L of classical logic (propositional or first order logic). The proposition A (resp. ¬ A) says that A is true (false).
If the system is subject to some unexpected perturbations represented as *reability → ¬perturbation*, could be interpreted by « something » protects against perturbations. We are in a logical framework, so it is possible to represent almost everything in a natural way. But the price to pay is the complexity. If you use the entire first order language can be the combinatorial explosion of algorithms and incompleteness.
The goal of this paper is the interactions between subsystems view as a very simple form of causality. To express these interactions it is common to represent by two binary relations *connect(A,B) and failed(A,B)*. The first relation means, for example, a subsystem A stands of a subsystem B. The second relation is a failure. Conventionally, these relations are represented by $A \Rightarrow B$ *and* $A \dashv B$. Of course, this causality is basic and a lot of research papers describe this type of representation of the causality.
Depending on the context, true could be interpreted as known, certain, believed ... or, more technically in a system of automated theorem proved.
The first idea is to express these laws in classical logic by axioms:

$cause(A, B) \wedge A \Rightarrow B$
$failed(A, B) \wedge A \Rightarrow \neg B$

Therefore, to provide the causal links between our relations connect and failed in a classical language (propositional calculus or first order logic) it is necessary to describe :

1. the internal characteristics of relations and cause and block failure
2. the links between these relations and classical logic

They can also be weakly expressed more by rules of inference, close to Modus Ponens :

$cause(A, B),\ A \vdash B$
$failed(A, B),\ A \vdash \neg B$

But these two formulations are problematic when a conflict appears.
For example, a set of four formulas $F = \{A, B, cause(A, C), failed(B, C)\}$, leading to infer from F, B and ¬ B and this is inconsistent. To solve such conflicts, we can try to use some methods inspired by constraint programming, as the negation by failure.
It is also possible to use a defeasible reasoning, especially a nonmonotonic logic. The first method (negation by failure) poses many theoretical and technical problems if you leave the simple cases. These problems are often solved by adding properties to the formal system, properties that pose other problems.

3.1. Causality and default logic

To resolve conflicts seen above, the intuitive idea is to lighten the formulation of rules of causality:

(1 ') If A causes B, if A is true, and it is possible that B, then B is true.
(2 ') If A blocks B, if A is true, and it is possible that B is false then B is false.

The question then is to describe as formally as possible. This question began to arise in artificial intelligence thirty years ago, when it was formalized the natural human reasoning. In this type of reasoning, it is necessary to reason with incomplete information, uncertain and subject to revision and sometimes false information. On the other hand we have to choose between several possible conclusions contradictory. The basic example is: {The penguins are birds, birds fly, penguins do not fly}. If Tweety is a penguin we arrive at a contradiction, the system is inconsistent. This inconsistency can be ignored if we can handle the exception by replacing "Birds fly" with "Typically birds fly". The nonmonotonic logic formally describes the modes of reasoning that takes into account these phenomena.
To represent the reconfiguration of resilient systems we propose to use default logic of Reiter. In this logic, the rules (1) and (2) will be expressed intuitively.

(1) If A causes B, if A is true, and if B is not contradictory, then B is true.
(2) If A blocks B (because A failed), if A is true, and if ¬ B is not contradictory then ¬ B is true.
In default logic, these rules can be represented by normal defaults and written:

$$d1 = \frac{A : B}{B}$$

$$d2 = \frac{A : \neg B}{\neg B}$$

Therefore, the information is represented here using defaults theory $\Delta = \{W, D\}$, where W is a set of classical logic formula and is the set of defaults used to represent the uncertainty of some information.
The classical definition of extension is based on the utilization of W and a subset of defaults D. The condition to use a default starts by checking the prerequisites are satisfied and the consequence doesn't lead to contradiction. In a simple manner that means his negation is not verified. If this request is *TRUE* we add the consequence to W and the algorithm is restarted until all defaults has been used.
For example, consider $\Delta = \{W, D\}$ with $W = \{A\}$ and $D = \{d1, d2\}$.

The 2 extensions are :

E1 = { A, B} if d1 is used.

E2 = { A, B} if d2 is used.

By using default logic, the conflict is resolved, but it is not possible to rank the extensions: B is true or false ? In fact this will really depend on the context. Some times the positive interactions are preferred to negatives. Another possibility is to use probabilistic or statistical methods or to weight each extension based on the evaluation of the knowledge. From algorithmic viewpoint of the ranking of extension could be evaluated also during the calculation of the extensions even the off-line ranking is preferred.

4 Representation of Resilient Systems Reconfiguration

How it is described above the defaults are used to manage incomplete information. Its most general form, a default is an expression of the form:

$$D=(A_x(X):B_y(X) \wedge C(X))/(C(X)) \qquad \text{A-5}$$

where $A_x(X)$, $B_y(X)$ and $C(X)$ ($x = 1,2, ..., m,\ y = 1,2, ..., l$) are well-formed formulas which contain first order as free variable X or $X = (x_1, x_2, x_3, \ldots, x_n)$ as a vector of free variables. $A_x(X)$ are the prerequisites, $B_y(X)$ are the justifications and $C(X)$ is the consequent.

The default (A-5) means informally: if $A_x(X)$ are verified (at some moment t_i), if possible that $B_y(X)$ are real ($B_y(X)$ are consistent), and if possible that $C(X)$ is true (at the moment t_i+1), then we infer $C(X)$ (at the moment t_i+1).

The use of defaults increases the number of formulas derived from the knowledge base W: we get extensions that are sets of theorems derivable monotonically.

An extension of the default theory $\Delta = (D, W)$ is a set E of formulas, closed for the deduction, containing W and satisfying the following property: if d is a default of D whose prerequisites $A_x(X, t_i)$ are in E, without the negation of justifications $B_y(X)$ and of consequent $C(X, t_{i+1})$ are in E, then the consequent of d is in E.

Formally, the extensions are defined as follows:

E *is an extension of* Δ *iff* $E = \bigcup_{i=0,\infty} Ei$, *with*

$E_0 = W$

and for $i > 0$,

$$E_{i+1} = ThE_i \cup \{C(X,t_{j+1}) / \frac{(A_x(X):B_y \wedge C(X))}{C(X)} \in D, A_x(X) \in E_i(at\ t_j), \neg By \notin Ei,$$

$$\neg CX \notin Ei(at\ t_{j+1})\}$$

where $Th(Ei)$ *denotes the set of theorems obtained monotonically from* $E_i : ThE_i = \{w / E_i \vdash w\}$.

The calculation of extensions allows to study the defaults one by one and to retain those who respond to the problem and are compatible with each other. Each extension corresponds to a possible solution of the problem. To calculate an extension, we must verify that the negation of justification does not belong to E_i. We can therefore use an incremental algorithm for computing extensions.

For a default theory $\Delta = (D, W)$, with the set of defaults D and the knowledge base W, the calculation is extended according to the algorithm:

```
Input : E=θ; (set of extensions E is empty).
Output : E=U(i=0,N) Ei.
calcul_extension(E) :
{
while there is a default D=(Ax (X):By(X)∧C(X))/(C(X))
that has not yet been inspected do
    - Select the default D,
    - Verify that the prerequisites Ax(X) are true (at
    some moment tj),
    - Verify that the justifications By(X) are
    consistent with W,
    - Verify that the consequent C(X) is consistent
    with W (at the moment tj+1),
    - Add By(X) and C(X, tj+1) to W.
end while
End of the calculation for an extension.
Backtracking (Deleting the last C(X,tj+1) and By(X) added
to W).
calcul_extension(E).
}
```

In our model, to provide links between these subsystems active and non-active by failure, the intuitive idea is to weaken the formulation of 3 causation rules:

(1) *If*
system(A,ON,t_i) , connect(A,B) and connect(B,C) are true,
and if
it is possible that reliable(A,B), non_reliable(B,C) and system(B,ON,t_{i+1}),
then
system(B,ON, t_{i+1}) is true.

(2) *If*
system(A,ON,tj), connect(A,B), and connect(B,C) are true,
and if
it is possible that not_reliable(A,B), reliable(B,C) and system(B,OFF,t_{j+1}),
then
system (B,OFF, t_{j+1}) is true.

(3) *If*
system(A,OFF,t_k), connect(A,B) are true,
and if it is possible that reliable(A,B) and system (B,OFF,t_{k+1}),
then
system(B,OFF, t_{k+1}) is true.

The predicate *reliable* has the meaning of activity of two entities and the first entity trigs the second one.
Formally the possible connectivity between 3 subsystems A,B,C are described in default logic by :

(1') *If*
system(A,ON,t_i), connect(A,B) and connect(B,C) are true,
and if
connect(A,B),non_connect(B,C) and system(B,ON,ti+1) are not contradictory,
then
system(B,ON, t_{i+1}) is true

(2') *If*
system(A,ON,tj), connect(A,B) and connect(B,C) are true,
and if
non_connect(A,B), reliable(B,C) and system (B,OFF,t_{j+1}) are not contradictory,
then
system (B,OFF, t_{j+1}) is true

(3') *If*
system(A,OFF,tk) and connect(A,B) are true,
and if
reliable(A,B) and system(B,OFF,t_{k+1}) are not contradictory,
then
system(B,OFF, t_{k+}1) is true

In default logic, these rules will be represented by the set of defaults *D* and written as:

$$d1:(system(A,ON)\wedge connect(A,B)\wedge connect(B,C):reliable(A,B)\wedge non\_reliab(B,C)\wedge system(B,ON))/(system(B,ON))$$
$$d2:(system(A,up)\wedge connect(A,B)\wedge connect(B,C):non\_reliable(A,B)\wedge reliable(B,C)\wedge system(B,OFF))/(system(B,OFF))$$
$$d3:(system(A,OFF)\wedge connect(A,B):reliable(A,B)\wedge system(B,OFF))/(system(B,OFF))$$

Therefore, the conflict has been resolved.

If we consider a plant with 5 entities *A,B,C,D,E* connected between them, and A is submitted to a perturbation. We want to know what is the possible reconfigurations of B,D,C and E.
Using default theory $\Delta = (D, W)$, in that W = {*perturbation*(A, up, t_0)}, by applying the algorithm above, we have 12 exceptions.

The following is one of them:

joint(system(A,ON,t_0),non_reliable(A,B),reliable(B,D)) -> system(B,OFF,t_1)
joint(system(B,OFF,t_1),reliable(B,C)) -> system(C,OFF,t_2)
joint(system(B,OFF,t_1),reliable(B,D)) -> system(D,OFF,t_2)
joint(system(D,OFF,t_2),reliable(D,E)) -> system(E,OFF,t_3)

This result us the worst one because the configuration of the complex systems is not able assure a healthy behavior in the case of a Fault on the subsystem A even if A keeps nominal parameters and it is considered ON.

5 Conclusion

We have introduced a new-switched system model based on a hybrid approach. To switch from one dynamic to another one we use Default Logic. The most important property, which assumes the reliability, is the flatness of the subsystems.
All these representations consider the problems of uncertain and revision. For the first aspect a minimum and necessary link between two causal relationships, it was necessary to formalize by using default logic.
All this approach offers a model of simulation for resilient systems and the future work will consider the structure network as fundamental of complex systems.

References

1. H. Goto, Y. Hasegawa, and M. Tanaka, “Efficient Scheduling Focusing on the Duality of MPL Representatives,” Proc. IEEE Symp. Computational Intelligence in Scheduling (SCIS 07), IEEE Press, Dec. 2007, pp. 57-64, doi:10.1109/SCIS.2007.357670.
2. H. Nabeshima, K. Iwanuma and K. Inoue. SOLAR: a consequence finding system for advanced reasoning. Proc. Eleventh Int. Conf. Automated Reasoning with An- alytic Tableaux and Related Methods, Proc. TABLEAUX 2003, LNAI 2796, pages 257–263, Springer, 2003.
3. C.S. Holling, "Resilence and stability of ecological systems", Annual Review of Ecology and Systematics, vol. 4, 1973, pp. 1-23.
4. G. Hamel, L. Välikangas, "The quest for resilience", Harvard Business Review, Sept. 2003.
5. E. Hollnagel, D. Woods, N. Leveson (Eds.), Resilience Engineering – Concepts and Precepts, Ashgate, 2006.

Learning Multiple Description Logics Concepts

Raphael Melo[1], Kate Revoredo[1], and Aline Paes[2]

[1] Postgraduate Information Systems Program, UNIRIO
Rio de Janeiro, Brazil
[2] Department of Computer Science Institute of Computing, UFF
Rio de Janeiro, Brazil
{raphael.thiago, katerevoredo}@uniriotec.br, alinepaes@ic.uff.br

Abstract. Description logics based languages have became the standard representation scheme for ontologies. They formalize the domain knowledge using interrelated concepts, contained in terminologies. The manual definition of terminologies is an expensive and error prone task, therefore automatic learning methods are a necessity. In this paper we lay the foundations of a multiple concept learning method that uses virtual concepts to aid the learning process, yielding more compact and readable terminologies. In this paper, we define virtual concepts and how they can be implemented in the current concept learning methods. We show through experiments how the method stacks up against other multiple concept learning methods.

1 Introduction

Description logics (DLs) [1] form a family of knowledge representation languages, with different expressive power, that are typically decidable fragments of first order logic (FOL). With DL it is possible to represent domain concepts and their relations. Moreover, due to their computational power and expressiveness, they have been widely used in Semantic Web [2] for ontology representation.

The task of defining the domain knowledge through a DL is usually done manually, which is time consuming and error prone, even more because the domain experts themselves do not always agree about the definitions of concepts and their relationships [3]. Therefore, to consider applying machine learning techniques [4] for automatically learning in DLs is relevant and sometimes even required. A number of DL learning approaches have been proposed in the literature [5] [6]. In these approaches each concept is learned independently from each other, thus, none of the concepts being learned are considered in the definition of the others. If this assumption is removed, the final ontology could be clearer and closer to the way that the concepts are related in the underlying domain. In this sense, another question arises: "What is the best order to learn a set of

[1] The first author would like to thank CAPES for the financial support through a master scholarship.

related concepts?" In [7], we address this problem by defining an algorithm that discovers a taxonomy of the concepts being learned and then uses this taxonomy to define the order for learning the concepts. However, since the order is defined by a heuristic aimed at finding relations between concepts and subconcepts, it may be the case that the ordering found does not yield the best solution for the learning task. Moreover, during the learning process only previously learned concepts are considered.

In this paper, we lay the foundations of multiple concept learning using the ideas discussed in [7] as motivation. Thus, we propose a learning strategy that allows concepts not yet learned to appear in the definition of other concepts, thus making possible to learn more compact and readable terminologies.

The paper is organized as follows. In Section 2, Description Logic and concept learning are reviewed. In Section 3, we present the proposed approach to learn multiple concepts. Section 4 presents some preliminary experimental results. Section 5 concludes the paper and presents the next steps of the research.

2 DLs and concept learning in DLs

DLs knowledge bases ($\mathcal{KB}$) have two components: a *TBox* and an *ABox*. The *TBox* contains intensional knowledge in the form of a terminology. Knowledge is expressed in terms of *individuals*, *concepts*, and *roles*. Thus, the terminology consists of concepts, which denote a set of individuals and roles which denote binary relationships between individuals. In this paper, we assume the common assumption made about DL terminologies: (i) only one definition for a concept name and (ii) concept definitions are acyclic. The *ABox* is composed of assertions about the individuals of the domain. An assertion states that an individual belongs to a concept or that a pair of individuals satisfies a role. Attached to a DL's $\mathcal{KB}$ there must be a reasoning mechanism, responsible for inferring information about individuals from the $\mathcal{KB}$.

There are a number of existing approaches to automatically learn DLs concepts, most of them [5] [6] [8] are inspired by Inductive Logic Programming(ILP) [9] techniques. The goal is to induce concept descriptions from existing evidences. When learning a concept, one has the purpose of finding a generalized and correct definition of such a concept from a set of examples, as defined below:

Definition 1 (Concept Learning)
Given:

- *a knowledge base* $\mathcal{KB}$,
- *a target concept* $Target$ *such as* $Target \notin \mathcal{KB}$,
- *a set of target examples* $\mathcal{E}$*, divided into positive* ($\mathcal{E}_p$) *and negative* ($\mathcal{E}_n$) *examples, such that* $\mathcal{E} = \mathcal{E}_p \cup \mathcal{E}_n$

Find:

- *A definition of the concept* $C(Target \equiv C)$ *such that* $\mathcal{KB} \cup C \models \mathcal{E}_p$ *and* $\mathcal{KB} \cap C \not\models \mathcal{E}_n$.

Although Definition 1 requires that the learned concept covers all the positive examples and none of the negative examples, these hard criteria are usually relaxed to enable the induction.

The concept learning task can be expanded to multiple concept learning, as defined below:

Definition 2 (Multiple Concept Learning)
Given:

- *knowledge base* $\mathcal{KB}$,
- n *concept learning tasks* $\{A_{C_1}, A_{C_2}, \ldots, A_{C_n}\}$, *where* $A_{C_i} = \{\mathcal{E}_{p_i}, \mathcal{E}_{n_i}\}$

Find:

- $\mathcal{KB}' = \mathcal{KB} \cup \mathcal{T}$, *where* $\mathcal{T}$ *is a taxonomy which holds the concepts definitions found in the* n *concept learning tasks.*

In this paper we focus on multiple concept learning methods that return a $\mathcal{T}$ that have compact definitions.

In [7] we presented a pre-processing method for multiple concept learning called terminology learning. It yields compact terminologies by defining an order to execute each concept learning task. This order is definied by finding, before the learning process, all the *subsumee* and *subsumer* relationships among the concept learning tasks using the shared individuals in the example sets as evidence.

However, although the *subsumee* and *subsumer* relation is important when devising an order, it is not the only relationship among concepts that can impact the later learned definition. Thus, in this paper we follow a different approach to find out the concepts that should be used to define another concept. Instead of directly finding a taxonomy from the set of examples, we let the learning algorithm decide what is the best way of defining each concept.

3 Multiple DL Concept Learning

The concept learning task can be viewed as a search problem over the space of concepts, created using three basic elements [5]: (i) a refinement operator to build the search tree of concepts; (ii) a search algorithm to control how this search tree is traversed; (iii) a scoring function to evaluate the nodes of the tree and to point out the best current concept candidate.

The refinement operator is responsible for defining a number of rules, from which a valid candidate definition for a concept is yielded. The candidate definitions are created from combinations of *known concepts*, *roles* and *constructors*. The constructors are different according to the DL language chosen. We propose to take into account an additional type of concept when the refinement operator is generating a concept definition, henceforth called *virtual concept*.

Virtual concepts are concepts that do not have an explicit definition yet. As usual, these concepts have a set of examples related to it, divided into positives and negative examples. Once a definition for a concept is learned, it should have

considered the individuals marked as positive examples as belonging to it. On the other hand, the negative examples are the individuals that do not belong to that concept and its definition should be able to indicate that. Since each concept has these sets of examples associated to it, it is possible to take into account the set of examples instead of the concept definition when referring to a concept. In this way, we can say that, *while* a concept is not selected to be the target one, it also behaves as a virtual concept.

In order to consider virtual concepts in the concept learning task, it is necessary to make the scoring function cope with them. Usually, the scoring function is either the cover relationship or a variation of it. The cover relationship is a function that evaluates how many positive and negative examples are inferred by a candidate definition. So this can be achieved by transforming the example sets of a virtual concept into assertion inside the ABox, e.g., Albert $\in E_p$ of C, then the assertion C(Albert) will be added to the ABoxes of all concept learning tasks that could use the virtual concept C.

We argue that adding the assertions of a virtual concept can have the same result as that of the regular inference if the following assumptions holds: all the virtual concepts in the multiple concept learning task should share the same ABox and all the relevant individuals for a particular virtual concept should be covered in one of its example sets. If that is indeed the case, it is possible to use virtual concepts in the same way that other non-target instantiated concepts are used inside the concept learning task. Moreover, the addition of virtual concepts in the learning process will allow it to find concept definitions more compact then the ones found with the terminology learning method [7]. The terminology learning method only deals with one type of usage relationship between virtual concepts, the *subsumee* and *subsumer* relationship. This kind of relationship is only related to the concept and subconcept relationship, e.g. in the kinship domain we have $\mathsf{Grandfather} \sqsubseteq \mathsf{Father}$. However, it can not reliably work with relationships that differs from *subsumer* and *subsumee*, for example, the disjunction between Grandfather and Grandmother to define Grandparent. The method proposed in this paper is capable of dealing with it because we turn the learning task into the responsible component for finding relationships between concepts.

The proposed method has the local goal to find the most compact definition for a single virtual concept. The major concern that arises from this is the formation of cycles. This could be avoided by two different approaches: (i) when constructing the ABox' for a target concept, the addition of facts associated to virtual concepts that uses the current target concept in their definition can be avoided. (ii) with a post-processing procedure. The first one is likely to be more efficient in returning a solution, but does not guarantee that this is the optimal solution, while the second may have a better chance to return the optimal solution, but some concepts may be relearned several times, i.e., the learning process can become less efficient.

4 Preliminary Experiments

To evaluate the proposed method some experiments were devised considering the concepts GRANDPARENT (GP), GRANDFATHER (GF) and GRANDMOTHER (GM) from the Family ontology.

A knowledge base of the family domain was used to run the experiments [3]. Each individual cited in the knowledge base is either a positive example or a negative example to a concept. The description learning component chosen to learn the concepts is the DL-Learner system with default settings, since it is a largely used environment to learn DLs.

The first analysis we conducted showed that our proposal, Multiple Concept Learning (MCL), is able to learn a terminology more compact and readable than the Concept Learning (CL) approach, which learns the concepts individually and independently. Table 1 shows the definition for the three concepts. Notice that GRANDPARENT is defined as a disjunction of the two other concepts.

Another evaluation concerns whether MCL is able to learn a more compact terminology when compared with the Terminology Learning (TL) task. For this comparison, we considered two concept orders for MCL:

(i) $<$ GRANDPARENT, GRANDFATHER, GRANDMOTHER $>$ and

(ii) $<$ GRANDMOTHER, GRANDFATHER, GRANDPARENT $>$.

The first one, was found by the approach proposed in [7] and then is the same one used by TL. Moreover, we avoid cycles by changing the correspondent ABox as described in Section 3. The results in Table 1 shows that different learning orders yield different results, and that the proposed method can achieve the same result as the terminology learning by reversing the order. This demonstrates that the proposed method is more versatile than the terminology learning, because it isn't bound to a learning order, while still maintaining the accuracy.

To sum up, these results point out that the method has the potential for finding compact solutions when avoiding cycles with pre-processing (MCL + Order 1 or 2), and its ability to find optimal solutions, if a post processing method to avoid cycle is used (MCL).

5 Conclusion and future remarks

In this paper we laid the foundations over which a multiple concept learning method could be built. We defined a new type of concept, the virtual concept, analogous with the definition of the concept learning task, but with a twist to make it usable inside the existing learning process of another concept. The proposed method opens the possibility of finding all the possible usage relationships among virtual concepts. Because of this, we argued that the use of virtual concepts could yield better results than the ones found with the method presented in [7] and in regular concept learning methods. We also presented directions to deal with the cycle problem that may appear with the use of this method. An

[3] `ftp://ftp.cs.utexas.edu/pub/mooney/forte`

Table 1. Resulting Concepts Definition on All Learning Experiments

Experiment	**Concept**	**Definition**	**Length**
CL	GP	EXISTS parent.EXISTS parent.TOP.	5
	GF	(male AND EXISTS parent.EXISTS parent.TOP).	7
	GM	(female AND EXISTS parent.EXISTS parent.TOP).	7
MCL	GP	(grandfather OR grandmother).	3
	GF	(grandparent AND male).	3
	GM	(grandparent AND female).	3
TL=CL+Order	GP	EXISTS parent.EXISTS parent.TOP.	5
	GF	(grandparent AND male).	3
	GM	(grandparent AND female).	3
MCL+Order 1	GP	(grandfather OR grandmother).	3
	GF	(male AND EXISTS married.grandmother).	5
	GM	(female AND EXISTS parent.EXISTS parent.TOP).	7
MCL+Order 2	GP	EXISTS parent.EXISTS parent.TOP.	5
	GF	(grandparent AND male).	3
	GM	(grandparent AND female).	3

experiment concerning the Kinship domain showed that it is possible to learn a clearer and more compact terminology without requiring a good ordering of the concepts.

In the future we would like to analyze the behavior of the method on different data sets and different DL languages, since we believe the proposed method is capable to work with all possible constructors sets.

References

1. F. Baader and W. Nutt, "Basic description logics," in *The description logic handbook* (F. Baader, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, eds.), pp. 47–100, Cambridge University Press, 2 ed., may 2010.
2. T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, vol. 5, no. 284, p. 34, 2001.
3. A. Maedche and S. Staab, "Ontology learning for the semantic web," *IEEE Intelligent Systems and Their Applications*, vol. 16, no. 2, pp. 72–79, 2001.
4. T. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.
5. J. Lehmann and P. Hitzler, "Concept learning in description logics using refinement operators," *Machine Learning*, vol. 78, no. 1-2, pp. 203–250, 2010.
6. N. Fanizzi, C. d'Amato, and F. Esposito, "Dl-foil concept learning in description logics," in *Proceedings of the 18th International Conference on Inductive Logic Programming (ILP-2008)*, vol. 5194 LNAI of *Lecture Notes in Computer Science*, pp. 107–121, Springer, 2008.
7. R. Melo, K. Revoredo, and A. Paes, "Terminology learning through taxonomy discovery," *BRACIS "to appear"*, 2013.
8. L. Iannone, I. Palmisano, and N. Fanizzi, "An algorithm based on counterfactuals for concept learning in the semantic web," *Applied Intelligence*, vol. 26, no. 2, pp. 139–159, 2007.
9. L. De Raedt, *Logical and Relational Learning*. Springer, 2008.

Background knowledge-enrichment for bottom clauses improving.

Orlando Muñoz Texzocotetla and René MacKinney-Romero

Departamento de Ingeniería Eléctrica
Universidad Autónoma Metropolitana
México D.F. 09340, México

Abstract. In this paper we present a method to enrich the hypothesis language which is used to construct the bottom clause. This approach, which is embedded in the Aleph system, is based on numerical fixed subintervals and categorical subsets. Each subinterval/subset contains values, for each attribute which is predefined, that are related with the example to saturate. The enriched language allows to reduce the number of rules of the final theories and, in some cases, to improve the accuracy.

Keywords: Artificial Intelligence, ILP, discretization, grouping, numerical attributes, categorical attributes

1 Introduction

Most of the ILP algorithms induce one clause at a time in order to find a theory T. Each clause in T is usually constructed with a single numerical (or categorical) value for each attribute[1]. This may result in inaccurate theories with a lot of rules. To overcome these drawbacks, some ILP systems use approaches which are based in the following strategies:

Discretization. Some ILP systems discretize numerical attributes in a global way, and the final intervals are used during the learning process. TILDE [3] and ECL-GSD [5] follow this strategy. In [9], a global binary discretization method is presented, but in addition to this each categorical attribute is grouped in two subsets. **Propositionalization.** These strategies consist in transforming a relational problem into a propositional problem. The main objective it is to solve a problem that was originally a relational problem, with faster and more efficient propositional algorithms than the relational systems. LINUS and DINUS [7] systems follow that procedure. **Genetic algorithms.** To deal with numerical attributes, some systems implement genetic refinement operators which search globally the best intervals. In [5], several refinement operators are presented, and a genetic algorithm is used to test the intervals. **Numerical Reasoning.** In [11], new relations (equalities, inequalities, regression models, etc.) are added a priori into the background knowledge. These relations are lazily evaluated during the bottom clause construction. Numerical reasoning is improved in [1], since

[1] We refer to the arguments of predicates as attributes.

it proposes to improve noise handling by mean of statistical-based techniques. Regarding the categorical data, we think that using grouping algorithms for categorical attributes can also improve the final theories.

In this paper we present a method to enrich the hypothesis language which is used to construct the bottom clause. This approach deals with both categorical and numerical attributes, in a local or global way. The enriched language allows to reduce the number of rules of the final theories and, in some cases, to improve the accuracy. This paper is organized as follows: section 2 describes the proposed method; section 3 shows the experimental results on several data sets; finally, section 4 presents our conclusions and future work.

2 The method

To enrich the hypothesis language our method tests different numerical subintervals (or categorical subsets) according to an evaluation function, then the best qualified are added into the background knowledge. Due to the great amount of subintervals/subsets that can be created we decided to use an evolutive approach to generate and evaluate them. This method was embedded in the basic algorithm of Aleph as follows:

0. State parameters. In addition to the mode declarations, types and determinations, users can also declare the attributes to be discretized/grouped in any of the two following ways:

i. $lit(\ldots, Attr, \ldots)$, $goal(\ldots, Class)$
Where $Attr$ is the attribute to discretize/group respect to $Class$ which can have more than two values. The predicate lit is into the background knowledge (it is possible to declare two or more attributes in lit), and $goal$ is the target predicate.

ii. $lit(\ldots, Attr, \ldots)$
In this case the corresponding classes are *positive* and *negative*.

1. Select an example e. In this step, a subset of values of $Attr$ are related to the example selected. For instance, let $e = goal(a)$ be the example selected and $lit(a, 12)$, $lit(a, 15)$, $lit(a, 20)$, $lit(b, 30)$, $lit(b, 35)$ be five facts in the background knowledge then the values $12, 15, 20$ are related to e. If $Attr$ is a numerical attribute, then these values form a *FixedSubinterval*. If $Attr$ is a categorical attribute, then we call *FixedSubset* to these subset of values. These are fixed because it will invariably be in all subintervals/subsets that will be tested.

1.5 Discretization/Grouping. Before saturation each attribute declared by the user is processed. Depending on the attribute type there are two possible cases.

i. Let Attr be a numerical attribute and $FixedSubinterval = [min, max]$ be the interval that contains all values in $Attr$ that are related to e. Then the proposed genetic algorithm returns a set of numerical intervals I_n such that: $|I_n|$ is defined by the user and $I_n = \{x \mid x = [min', max'] \wedge min' \leq min \wedge max \leq max'\}$.

Furthermore each element in I_n represents a chromosome. To evolve the chromosomes several mutation operators are defined. These can enlarge or shrink an interval, either on the left side, on the right side, or on both sides. The

fitness function for a chromosome x is given by the information gain, see Eq. (1). In this case the GA maximizes the information gain to find better subintervals.

$$IG(x) = ent(Int) - \frac{count(x)}{count(Int)} ent(x) \tag{1}$$

The corresponding entropy for some interval Int is:

$$ent(\text{Int}) = -\sum_{j=1}^{|S|} p(Int, j) log_2(p(Int, j))$$

where: $|S|$ = number of classes, Int is the numerical interval of $Attr$, $count(Int)$ = numeric values into interval Int, $p(Int, j)$, is the proportion of values in Int that belong to the jth class.

ii. Let Attr *be a categorical attribute* and *FixedSubset* be the set of all values related to e. The proposed GA searches the best $|C|$ subsets of categorical values with the following constraint: $C = \{y \mid FixedSubset \subseteq y\}$.

A chromosome is represented by a subset y. The mutation operators defined add a new categorical value in y, delete an element from the chromosome, or swap an element between the chromosome and the rest of categorical values, but maintaining the FixedSubset into each new chromosome. The crossover operator swaps an element between two chromosomes.

In this case, the fitness function is based on the distance between two values of a categorical attribute. The fitness for a chromosome y is the total sum of the distances for each value pair in y. In addition to this, if the sum of distances for two or more chromosomes is the same, then the value $\frac{1}{|y|}$ is added, because we want to minimize and to favor the subsets with more values. This fitness function, called DILCA [6], is showed in Eq. (2).

$$Fitness(y) = \sum_{c_i c_j \in y} \sqrt{\sum_{s_t \in S} (P(c_i \mid s_t) - P(c_j \mid s_t))^2} + \frac{1}{|y|} \tag{2}$$

where: $c_i, c_j \in y$ such as $i \neq j$. S is the set of classes defined, $P(c_i \mid s_t)$ is the conditional probability for c_i given s_t, and $P(c_j \mid s_t)$ is the conditional probability for c_j given s_t.

In both cases the user can give the number of chromosomes to be added.

2. Build a bottom clause that entails the example e. The enriched language is used to build a bottom clause. Consider the goal relation $no\_payment(S)$ which is true for those students who do not need to repay its loan. If literals in background knowledge are $unemployed(S)$, $enrolled(S, School, Units)$, where $Units$ is numerical and $School$ is categorical, then a bottom clause would be like this:

```
no_payment(A) :- unemployed(A),enrolled(A,B,C),
interval(C,[11.0,13.4]),interval(C,[11.7,12.5]),interval(C,[11.3,13.3]),
interval(C,[11.3,14.0]),interval(C,[11.0,13.5]),interval(C,[10.0,12.3]),
member(B,[occ,smc,uci,ucla,ucsd]),member(B,[smc,ucb,ucla,ucsd]).
```

3. Search. Aleph uses several search strategies (best first, depth first search, etc.), refinement operators, and evaluation functions (as mentioned earlier).

4. Remove covered (redundant) examples and add the best clause found to the current theory. Go to step 1.

In the next section we present the experiments performed to compare the accuracy of our method.

3 Experiments

In this section we compare our method (we call it *Extended Aleph*) with the standard Aleph, and the lazy evaluation in Aleph [11]. Our method works as the latter, namely with the bottom clause construction as our method. A 10-fold, cross-validation was performed to compare the accuracy and the simplicity (number of rules) of the final theories.

From the UCI machine learning repository [2], we used Australian Credit Approval, Pittsburgh Bridges, Japanese Credit Screening, German Credit, Iris, Moral Reasoner, Ecoli, and Student Loan datasets. We also performed tests over the well-known ILP datasets: Carcinogenesis [12], KRK illegal chess position (KRKi) [8], and mutagenesis [10] (on "regression friendly" data).

All these methods were executed in Yap Prolog version 6.0, with the global parameters: $minpos = 2$, and $noise = 5$; the remaining of the parameters are fixed by default in Aleph. All experiments were performed on a modern multicore PC machine.

Extended Aleph vs Aleph. Table 1, shows that our implementation improves the accuracy in most databases, with the exception of two datasets: *Pittsburgh Bridges and Moral Reasoner*. We can also see (figure 1) that the increase of the accuracy for the datasets *Japanese Credit Screening and Iris* is significant, 15% and 7.8% respectively. Regarding the simplicity of the final theories, extended Aleph system does not improve the simplicity with two datasets: *German Credit and KRKi*, but with the other datasets (excepting *Moral Reasoner*) our approach decreases the number of rules in the final theories. In particular, with *Iris* and *Student Loan* the reduction is almost 50% (figure 2).

Extended aleph vs Lazy Evaluation in Aleph. As in the previous case, Extended Aleph improves the accuracy except in two datasets: *Moral Reasoner and Student Loan* (table 1). Furthermore, Lazy Evaluation overcomes our method only in one single dataset: *Student Loan*, (figure 2). Although both have the same accuracy.

Dataset	**Accuracy** (%)			**Number of rules**		
	Aleph	Ext. Aleph	Lazy	Aleph	Ext. Aleph	Lazy
Australian Credit Approval (Aus)	82	**83**	80.5	27.40	**24.70**	29.9
Pittsburgh Bridges (Pitts)	89	89	88.7	13.2	**13.1**	13.4
Japanese Credit Screening (Japan)	53	**69**	65	13.2	**8.6**	11.8
German Credit (German)	70.9	**71.2**	70.8	**51.4**	52.8	53.4
Iris	85.9	**93.7**	89.1	26.9	**13.7**	22.6
Moral Reasoner (Moral)	96	96	96	1	1	1
Ecoli	91	**92**	91	37.3	**35.4**	43.5
Student Loan (Student)	93.7	**97.6**	**97.6**	46.6	25.7	**21.5**
Carcinogenesis (Carcino)	58.6	**61.3**	56.3	21.9	**18.6**	28.9
KRKi	49.4	**50.4**	49.4	**66.7**	68.1	**66.7**
Mutagenesis (Muta)	77.1	**80.8**	78.3	11.2	**9**	10.4

Table 1. Predictive accuracies and number of rules of the analyzed datasets.

In general, Extended Aleph improves or maintains the accuracy (figure 1), and decreases the number of rules for most of datasets. With regard to the running time, Extended Aleph significantly exceeds this measure with Carcinogenesis: Aleph 36.1s, Lazy 222.33s, Ext. Aleph **698.9s**; and Ecoli: Aleph 11.5s; Lazy 63.2s; Ext. Aleph **707.5s**.

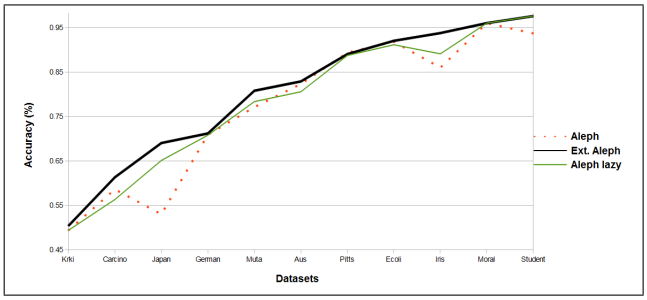


Fig. 1. Extended Aleph maintains or improves the accuracy.

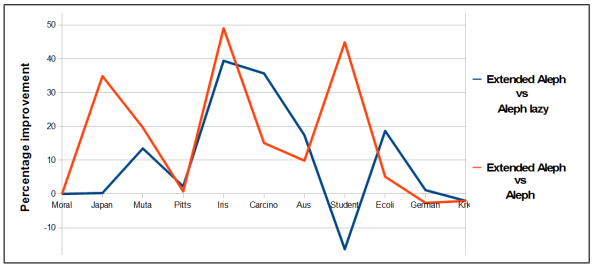


Fig. 2. Percentage reduction of the number of rules in final theories.

4 Conclusions and Future Work

With the obtained results, we can draw some conclusions: this method does not guarantee to improve the accuracy and simplicity of the theories in all cases. The improvement depends on the selected attributes, namely if an attribute is relevant in the theory construction then its processing will help to improve the final theory. ILP problems whose final theories do not need constants to be induce on them can not benefit from this method. In this case, variables are useful to better explain that concept. The size of the search space can be affected with the proposed method. On one hand a discretized/grouped attribute which is not relevant can generate a large amount of unnecessary candidate rules. On

the other hand a discretized/grouped attribute which is relevant can decrease significantly the number of candidate rules. Users do not know in advance which attributes will be most useful, therefore the intuition is an important element to select each attribute. Our method allows to experiment easily with different attributes. Finally, unlike other genetic approaches like SMART+ [4], our method can deal with both categorical and numerical attributes.

Some considerations for further work are as follows. Since not all cases were successful it is necessary to test other fitness functions and more datasets. These tests will help us to identify the factors which affect both accuracy and simplicity of the final theories, as well as the search space size. Another path of research is to look into multivariable predicates, if there are correlations between two or more attributes. Thus, we want to investigate if these correlations are relevant to improve the performance in ILP (accuracy and simplicity) and what kind of problems can be treated with these predicates. Finally, we want to implement in Aleph system these ideas and compare performance with other ILP systems that handle data types.

References

1. A. Alves, R. Camacho, and E. Oliveira, *Improving numerical reasoning capabilities of inductive logic programming systems*, in IBERAMIA, 2004, pp. 195–204.
2. K. Bache and M. Lichman, *UCI machine learning repository*, 2013.
3. H. Blockeel and L. D. Raedt, *Lookahead and discretization in ilp*, in In Proceedings of the 7th International Workshop on Inductive Logic Programming, Springer-Verlag, 1997, pp. 77–85.
4. M. Botta and A. Giordana, *SMART+: A multi-strategy learning tool*, in IJCAI, 1993, pp. 937–945.
5. F. Divina and E. Marchiori, *Handling continuous attributes in an evolutionary inductive learner.*, IEEE Trans. Evolutionary Computation, 9 (2005), pp. 31–43.
6. D. Ienco, R. G. Pensa, and R. Meo, *Context-based distance learning for categorical data clustering*, in IDA, 2009, pp. 83–94.
7. N. Lavrac, S. Dzeroski, and M. Grobelnik, *Learning nonrecursive definitions of relations with linus*, in Proceedings of the European Working Session on Machine Learning, EWSL '91, London, UK, UK, 1991, Springer-Verlag, pp. 265–281.
8. S. Muggleton, M. Bain, J. Hayes-michie, and D. Michie, *An experimental comparison of human and machine learning formalisms*, in In Proceedings of the Sixth International Workshop on Machine Learning, Morgan Kaufmann, 1989, pp. 113–118.
9. O. Muñoz and R. MacKinney-Romero, *Multivalue in ILP*, in 20th International Conference on Inductive Logic Programming, 2011.
10. Srinivasan, Muggleton, A. Srinivasan, and S. H. Muggleton, *Mutagenesis: Ilp experiments in a non-determinate biological domain*, in Proceedings of the 4th International Workshop on Inductive Logic Programming, volume 237 of GMD-Studien, 1994, pp. 217–232.
11. A. Srinivasan and R. Camacho, *Experiments in numerical reasoning with inductive logic programming*, Journal of Logic Programming.
12. A. Srinivasan, R. D. King, S. Muggleton, and M. J. E. Sternberg, *Carcinogenesis predictions using ilp*, in ILP, 1997, pp. 273–287.

Comparison between Explicit Learning and Implicit Modeling of Relational Features in Structured Output Spaces

Ajay Nagesh[123], Naveen Nair[123], and Ganesh Ramakrishnan[21]

[1] IITB-Monash Research Academy, Old CSE Building, IIT Bombay
[2] Department of Computer Science and Engineering, IIT Bombay
[3] Faculty of Information Technology, Monash University
{ajaynagesh,naveennair,ganesh}@cse.iitb.ac.in

Abstract. Building relational models for the structured output classification problem of sequence labeling has been recently explored in a few research works. The models built in such a manner are interpretable and capture much more information about the domain (than models built directly from basic attributes), resulting in accurate predictions. On the other hand, discovering optimal relational features is a hard task, since the space of relational features is exponentially large. An exhaustive search in this exponentially large feature space is infeasible. Therefore, often the feature space is explored using heuristics. Recently, we proposed a Hierarchical Kernels-based feature learning approach (StructHKL) for sequence labeling [18], that optimally learns emission features in the form of conjunctions of basic inputs at a sequence position. However, StructHKL cannot be trivially applied to learn complex relational features derived from relative sequence positions. In this paper, we seek to learn optimal relational sequence labeling models by leveraging a relational kernel that computes the similarity between instances in an implicit space of relational features. To this end, we employ relational subsequence kernels at each sequence position (over a time window of observations around the pivot position) for the classification model. While this method of modeling does not result in interpretability, relational subsequence kernels do efficiently capture relational sequential information on the inputs. We present experimental comparison between approaches for explicit learning and implicit modeling of relational features and explain the trade-offs therein.

Keywords: Subsequence Kernels, StructSVM, Sequence Labeling

1 Introduction

Structured output classification has gathered significant interest in the machine learning community during the last decade [24, 6, 21, 15].The goal of such works is to classify complex output structures such as sequences, trees, lattices or graphs, in which the class label at each node/position of the structure has to be inferred

based on observed evidence data. The possible space of structured outputs tends to be exponential and thus structured output classification is a challenging research area. We, in our research work, focus on a specific structured output classification problem, popularly known as sequence labeling. As in any classification setting, the sequence labeling domain is also characterized by complex relationships among entities and uncertainties in their relationships. Efficient models can be constructed by exploiting these relationships. However, discovering relationships that enhance the discriminative power of classifiers is a hard task, since the relationship space is often too large. Therefore, most of the research in sequence labeling and other structured output space classification, either ignore the complex relationships or use heuristics to learn the relationships. In this work, we focus on exploiting complex relationships in both the input as well as the output space in an efficient way to improve sequence labeling models. We begin with a brief introduction to the task of sequence labeling.

The objective in sequence labeling is to assign a state (class label) to every instance of a sequence of observations. Typical sequence labeling algorithms learn probabilistic information about the neighboring states along with the probabilistic information about the observations. Hidden Markov Models (HMM) [19], Conditional Random Fields (CRF) [9] and StructSVM [24] are three models used popularly for sequence labeling problems. The training objective can be posed as learning feature weights that make the score F ($F : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$), of the true output sequence Y greater than any other possible output sequence, given an input sequence X. The score is defined as:

$$F(X, Y; \mathbf{f}) = \langle \mathbf{f}, \boldsymbol{\psi}(X, Y) \rangle \tag{1}$$

where $\boldsymbol{\psi}$ is the feature vector (describing observations and transitions), and $\mathbf{f}$ is the weight vector. Inference is performed by the decision function $\mathcal{F} : \mathcal{X} \rightarrow \mathcal{Y}$ defined by

$$\mathcal{F}(X; \mathbf{f}) = \arg\max_{Y \in \mathcal{Y}} \; F(X, Y; \mathbf{f}) \tag{2}$$

Recent works have shown that learning the relational structure between input features improves the efficiency of sequence labeling models [13, 17, 12]. However, the space of relational features is exponential in the number of basic observations, making the discovery of useful features a difficult task. For instance, the simple case of learning features that are conjunctions of basic observations at any single sequence position results in a feature space that is exponential. The problem is further exacerbated if we consider complex relational features built from observations at different relative positions. An exhaustive search in this exponentially large feature space is infeasible. Therefore, most systems that learn relational features follow a greedy search strategy based on heuristics to select useful features. These approaches start with an initial (possibly empty) set of features and iteratively search (using some ordering of the feature space) for refinements that improve the heuristic score.

In our previous work [18], we propose and develop a Hierarchical Kernels based approach for optimally learning features which are conjunctions of basic

features at a particular sequence position (*simple conjuncts* or $\mathcal{SC}$s) for each label. The approach is referred to as Hierarchical Kernel Learning for Structured Output Spaces (StructHKL) [4]. Although it optimally learns the most discriminative $\mathcal{SC}$s, its applicability in learning complex relational features that are derived from observations at different relative positions in a sequence, is non-trivial and challenging. To address this issue, our follow-up work [16], determines simple feature classes that can be composed to yield complex ones, with the goal of formulating efficient yet effective relational feature learning procedures. We identify feature classes called absolute features ($\mathcal{AF}$) and composite features ($\mathcal{CF}$) in increasing order of their complexity respectively [5]. It is posited that optimal relational features can be learned by enumerating $\mathcal{AF}$s and discovering their useful compositions ($\mathcal{CF}$) using StructHKL. However, the space of $\mathcal{AF}$s is prohibitively large and it is not feasible to enumerate all of them in a domain. To circumvent this issue, we propose to selectively enumerate $\mathcal{AF}$s based on some relevance criteria such as the support of $\mathcal{AF}$s in the training set.

An $\mathcal{AF}$ is formed by combining one or more predicates which share variables. The partial ordering of $\mathcal{AF}$s does not comply with the requirement of StructHKL that the descendant kernels should be summable in polynomial time. This limits the possibility of leveraging StructHKL to optimally learn features in the space of $\mathcal{AF}$s (and its super-space of $\mathcal{CF}$s). For this reason, in the current piece of work, we leverage a relational kernel that computes the similarity between instances in an implicit feature space of $\mathcal{CF}$s. To this end, we employ the relational subsequence kernel [2] at each sequence/pivot position (over a time window of observations around it) for the classification model. We would like to learn composite features which capture relational information about basic observations at positions relative to the pivot position for every sequence step. This sequence information would provide a rich feature space for the algorithm to learn a more expressive model. However, explicitly enumerating such a feature space is not feasible due to the high dimensionality of the feature space. Relational subsequence kernels implicitly capture the effectiveness of this rich feature space. We also show that the feature space of $\mathcal{CF}$s (*explicit* features) are captured by the *relational subsequence kernels* (*implicit* features). While this way of modeling does not result in interpretability, relational subsequence kernels do efficiently capture the relational sequential information on the inputs.

We evaluate the performance of our approaches on publicly available activity recognition datasets. Our experiments show improvements over other standard and state-of-the-art sequence labeling techniques. The paper is organized as follows.

Section 2 discusses background work. We discuss our approach in Section 3. Experimental setup and results are discussed in Section 4 and we conclude the paper in Section 5.

[4] StructHKL is derived from StructSVM in which we use sparsity inducing hierarchical regulariser for observation features.

[5] For the definitions and examples of $\mathcal{AF}$, $\mathcal{CF}$ and other feature classes, please refer to [16].

2 Background

Approaches to learning relationships for sequence labeling could be based on basic input features at a single sequence step or input features at multiple sequence steps and/or relationships among output variables. Some of these approaches are discussed below.

McCallum [14] as well as Nair *et. al* [17] propose feature induction methods that iteratively construct feature conjunctions that increase an objective. These approaches start with an initial set of features (conjunctions or atomic) and at each step, consider a set of candidate features that are refinements of the current set of features. Features whose inclusion will lead to maximum increase in the objective are selected. Weights for the new features are trained. The steps are iterated until convergence. While McCallum trains a CRF model and uses conditional log-likelihood as the objective for the greedy induction, Nair *et. al* train an HMM and use prediction accuracy on a held out dataset (part of the training data) as the objective. This effectively solves the problem of incorrect assumption, that individual observations are independent, while not dealing with exponential observation space. Although these greedy feature induction approaches have been shown to improve performance, they cannot guarantee an optimal solution. An exhaustive search to find the optimal solution is expensive due to the exponential size of the search space.

Kersting *et. al.* [8] discusses the Logical Hidden Markov Model which is a relational representation of HMM. However, this work does not investigate learning the input structure. Thon *et. al* ([22], [23]) elaborate on relational markov processes which are concerned with efficient parameter learning and inference. They assume that a structure has been provided upfront. Similarly, a relational bayesian network learning is discussed in [20] with the goal of learning the parameters given the structure of the bayes-net.

Hierarchical Kernel Learning for Structured Output Spaces (StructHKL) [18], optimally and efficiently learns discriminative features for multi-class structured output classification problems such as sequence labeling. StructHKL builds on the Support Vector Machines for Structured Output Spaces (StructSVM) model [24] for sequence prediction problems, wherein, all possible $\mathcal{SC}$s form the input features while the transition features are constructed from all possible transitions between state labels. A ρ-norm hierarchical regularizer is employed to select a sparse set of $\mathcal{SC}$s. Since there is a need to preserve all possible transitions, a conventional 2-norm regularizer is employed for state transition features. The exponentially large observation feature space is searched using an active set algorithm and the exponentially large set of constraints is handled using a cutting plane algorithm.

In our follow-up work [16], we learn complex relational features derived from relative sequence positions. We propose to enumerate $\mathcal{AF}$s and leverage StructHKL to learn their compositions, which are $\mathcal{CF}$s. However, it is noted that the space of $\mathcal{AF}$s is prohibitively large and therefore it is not feasible to enumerate all $\mathcal{AF}$s in a domain. As a solution we selectively enumerate $\mathcal{AF}$s based on some relevance criteria such as support of the $\mathcal{AF}$ in the training set. A feature is

considered to be *strongly relevant* if it helps the classification model to discern classes optimally. On the other hand, a feature is *weakly relevant* if it covers atleast a threshold percentage of examples. As discovering strongly relevant $\mathcal{AF}$s is a hard task, the focus is on discovering weakly relevant $\mathcal{AF}$s using Inductive Logic Programming tools. Pattern mining approaches are employed to discover a relevant set of $\mathcal{AF}$s. Specifically, a relational pattern miner called Warmr [3] is used. Warmr uses a modified version of Apriori algorithm [1] to find frequent patterns ($\mathcal{AF}$s) which have minimum support, as specified by the user. Once a set of relevant $\mathcal{AF}$s are enumerated, StructHKL is used to learn useful compositions of $\mathcal{AF}$s and their parameters to get the final model. This can be viewed as projecting the space of complex relational features such as $\mathcal{CF}$s into the space of $\mathcal{SC}$s and leveraging StructHKL.

TildeCRF [5] has an objective similar to our approach, where the relational structure and parameters of a CRF for sequence labeling are learned. TildeCRF uses relational regression trees and gradient tree boosting for learning the structure and parameters. Unlike in TildeCRF, in this work, we derive convex formulations for learning relational models.

In this paper, we provide operative definitions of the feature classes such as $\mathcal{AF}$ and $\mathcal{CF}$. For a more detailed exposition of the feature classes and the relationships between them, the reader is pointed to our previous work [16].

3 Implicit Modeling of Features for Sequence Labeling

In Section 1, we have stated our objective as exploiting complex relationships among input variables in sequence labeling problems to improve the efficiency of classification. We now formalize our intuitions and present our proposed approach in detail.

We have presented the training and inference objectives of sequence labeling problems in equations (1) and (2), where the features and feature weights are represented by $\boldsymbol{\psi}$ and $\mathbf{f}$, respectively. Elements of $\boldsymbol{\psi}$ correspond to the emission (basic input/observation) features and the transition features. We represent the emission and transition parts of the vector $\boldsymbol{\psi}$ as $\boldsymbol{\psi_E}$ and $\boldsymbol{\psi_T}$, respectively. We assume that both $\boldsymbol{\psi_E}$ and $\boldsymbol{\psi_T}$ are vectors of dimension equal to the dimension of $\boldsymbol{\psi}$ with zero values for all elements not in their context. That is, $\boldsymbol{\psi_E}$ has dimension of $\boldsymbol{\psi}$, but has zero values corresponding to the transition elements. In the dual space, we represent the kernels corresponding to transition and emission as κ_T and κ_T respectively. Our proposed approach is to leverage (implicitly or explicitly) discriminative observation features ($\boldsymbol{\psi_E}$) that capture complex relationships among input variables in an implicit manner.

In the previous sections, we have identified $\mathcal{CF}$s as the class of features that explicitly capture complex relationships among input variables at relative sequence positions. We have also defined $\mathcal{CF}$s as compositions of $\mathcal{AF}$s and that, since the partial ordering of $\mathcal{AF}$s does not comply with the requirements of StructHKL, it is not feasible to leverage StructHKL for learning features in the space of $\mathcal{AF}$s (and its super-space of $\mathcal{CF}$s). For this reason, in the sequence

labeling model, we leverage a relational kernel that computes the similarity between instances in an implicit feature space of $\mathcal{CF}$s. To this end, we employ the relational subsequence kernel [2] at each sequence position (over a time window of observations around the pivot position) for the classification model. We now briefly discuss about relational subsequence kernels in the following paragraph.

Subsequence kernels have been used to extract relations between entities in natural language text [2], where the relations are between protein names in biomedical texts. The features are (possibly non-contiguous) sequences of word and word classes anchored by the protein names at their ends. They extend the string kernels [11] for this task.

We have defined $\mathcal{CF}$s as explicit features that capture the subset of features at the current position as well as its relative positions. To implicitly capture this feature space, we employ a relational subsequence kernel at each position of the input sequence, with the current position as the pivot position. Suppose we consider an input $\mathbf{x}_i^p$ at position p for example i. Let the previous k positions relative to p have inputs $\mathbf{x}_i^{p-1}, \ldots \mathbf{x}_i^{p-k}$ and next l positions relative to p have inputs $\mathbf{x}_i^{p+1}, \ldots \mathbf{x}_i^{p+l}$. Let there be N basic features at a time-step t denoted by $x^{1^t} \ldots x^{N^t}$.[6] Essentially our sequence for the particular time-step pivoted at p, denoted by Q^p, is as follows:

$$Q^p = \{x^{1^{p-k}}, \ldots x^{N^{p-k}}\}, \ldots, \{x^{1^{p-1}}, \ldots x^{N^{p-1}}\},$$
$$\{x^{1^p}, \ldots x^{N^p}\}, \{x^{1^{p+1}}, \ldots x^{N^{p+1}}\} \ldots \{x^{1^{p+l}}, \ldots x^{N^{p+l}}\}$$

Given two sequences Q^p and Q^q, we define the relational subsequence kernel $SSK(Q^p, Q^q)$ as elaborated in [2]. This kernel will implicitly enumerate all possible common subsequences between Q^p and Q^q. We now show that the feature space of $\mathcal{CF}$s are captured by our relational subsequence kernel.

Claim: Relational subsequence kernels implicitly enumerate all the features in the feature space defined by Composite Features ($\mathcal{CF}$) given a constant context window.

Proof. By their definition the relational subsequence kernel $SSK(Q^i, Q^j)$ will implicitly enumerate all possible common subsequences between Q^i and Q^j. $\mathcal{CF}$s are conjunctions of features in the present time-step with features present in time-steps before and after the current time-step, which can be represented by $\mathcal{AF}$s. Since we are considering all the sub-sequences in the given context (time) window in the relational kernel, we implicitly enumerate space of $\mathcal{CF}$s.

We now define the kernel for StructSVM framework below, which represents the kernel resulting from the difference in values for the original and the candidate sequences. This stands for the inner product, $\langle {\boldsymbol{\psi_i}}^{\delta}(Y), {\boldsymbol{\psi_i}}^{\delta}(Y^{'})\rangle$ with $\psi_i^{\delta}(Y)$ defined as: $\psi_i^{\delta}(Y) = \psi(X_i, Y_i) - \psi(X_i, Y)$. The kernel, which is a combination of transition (κ_T) and emission (κ_E) kernels, is defined as follows:

[6] Ignoring the example number i for simplicity

$$\kappa\big((X_i, Y_i, Y), (X_j, Y_j, Y^{'})\big) = \kappa_T(Y_i, Y, Y_j, Y^{'}) + \kappa_E\big((X_i, Y_i, Y), (X_j, Y_j, Y^{'})\big) \tag{3}$$

where

$$\kappa_T\big(Y_i, Y, Y_j, Y^{'}\big) = \kappa_T(Y_i, Y_j) + \kappa_T(Y, Y^{'}) - \kappa_T(Y_i, Y^{'}) - \kappa_T(Y_j, Y), \tag{4}$$

$$\begin{aligned}\kappa_T(Y_i, Y_j) &= \sum_{p=1}^{l_i-1}\sum_{q=1}^{l_j-1} \Lambda(y_i^p, y_j^q)\Lambda(y_i^{p+1}, y_j^{q+1}) \\ &= \sum_{p=2}^{l_i}\sum_{q=2}^{l_j} \Lambda(y_i^{p-1}, y_j^{q-1})\Lambda(y_i^p, y_j^q),\end{aligned} \tag{5}$$

$\Lambda(y_i^p, y_j^q) = 1$ if $y_i^p = y_j^q$; 0 otherwise. and

$$\kappa_E\big((X_i, Y_i, Y), (X_j, Y_j, Y^{'})\big) = \sum_{p=1}^{l_i}\sum_{q=1}^{l_j} \kappa_E(x_i^p, x_j^q)\Big(\Lambda(y_i^p, y_j^q) + \Lambda(y^p, y^{'q}) - \Lambda(y_i^p, y^{'q}) - \Lambda(y^p, y_j^q)\Big) \tag{6}$$

In our setting of subsequence kernels for StructSVM, the kernel $\kappa_E(x_i^p, x_j^q)$ is the relational subSequence kernel, where we may be considering some window time steps before and after p and q, with p and q as pivots.

The dual of the primal SVM formulation as defined by Tsochantaridis et. al. [24] for structured output spaces with the new kernel can be written as,

$$\begin{aligned}&\max_{\alpha} \sum_i \sum_{Y \in S_i} \alpha_{iY} - \frac{1}{2}\sum_i \sum_{Y \in S_i} \sum_j \sum_{Y' \in S_j} \alpha_{iY}\alpha_{jY'}\Big(\kappa_T^{\delta}(Y_i, Y, Y_j, Y^{'}) + \kappa_E^{\delta}\big((X_i, Y_i, Y), (X_j, Y_j, Y^{'})\big)\Big) \\ &\quad s.t.\ \ \forall i, \forall Y \in S_i, \quad \alpha_{iY} \geq 0 \\ &\qquad \forall i, \quad m\sum_{Y \in S_i} \frac{\alpha_{iY}}{\Delta(Y_i, Y)} \leq C.\end{aligned} \tag{7}$$

where α is the Lagrange dual variable, Δ is the loss function, S_i and S_j are the active constraint sets for example i and j respectively.

Now the margin violation cost function for a candidate output sequence Y for example i (for the cutting plane algorithm) can be written as,

$$\begin{aligned}H(Y) &= \Big(1 - \langle \boldsymbol{\psi}_i^{\delta}(Y), \mathbf{f}\rangle\Big)\Delta(Y_i, Y) \\ . \quad &= \Big(1 - \sum_j \sum_{y^{'} \in S_j} \alpha_{jY'} \langle \boldsymbol{\psi}_i^{\delta}(Y), \boldsymbol{\psi}_j^{\delta}(Y^{'})\rangle\Big)\Delta(Y_i, Y) \\ . \quad &= \Big(1 - \sum_j \sum_{y^{'} \in S_j} \alpha_{jY'} \kappa\big((X_i, Y_i, Y), (X_j, Y_j, Y^{'})\big)\Big)\Delta(Y_i, Y)\end{aligned} \tag{8}$$

where S_j is the active constraint set for example j.

The dual objective and the margin violation cost function can be plugged into the cutting plane algorithm to solve the objective. While this way of modeling does not result in interpretability, relational subsequence kernels do efficiently capture the relational sequential information on the inputs.

As in typical sequence labeling systems, we perform inference using a dynamic programming approach called the Viterbi algorithm [4].

The next section discusses our experiments and results.

4 Experiments

Our entire implementation is in Java. Our experiments are carried out on two publicly available activity recognition datasets. The first is the data provided by [7]. The dataset is extracted from a household fitted with 14 binary sensors. Eight activities have been annotated for 4 weeks. Activities are daily house hold activities like *sleeping*, $usingToilet$, $preparingDinner$, $preparingBreakfast$, $leavingOut$, *etc.* A data instance is recorded for a time interval of 60 seconds and there are 40006 such data instances. Since the authors of the dataset are from the University of Amsterdam, we will refer to the dataset as the UA data. The second data is the relational activity recognition data provided by [10] of Katholieke University, Leuven. We refer to the data as KU data. The data has been collected from a kitchen environment with 25 sensors/RFID attached to objects. There are 19 activities annotated. The data has been divided into 20 sequences. In this data, we perform our experiments in a leave one out cross-validation setup and report average of the accuracies returned from each fold.

In UA data, We use 25% of data for training and the rest for testing and report all accuracies by average across the four folds (the dataset is split into different sequences and each sequence is treated as an example). We report both micro-average and macro-average prediction accuracies. The micro-average accuracy is referred to as time-slice accuracy by [7], and is the average of per-class accuracies, weighted by the number of instances of the class. Macro-average accuracy, referred to as class accuracy by [7], is simply the average of the per-class accuracies. Micro-averaged accuracy is typically used as the performance evaluation measure. However, in data that is biased towards some classes, too worse macro-average is an indicator of a bad prediction model.

As we discussed previously, we leverage a relational kernel that computes the similarity between instances in an implicit feature space of $\mathcal{CF}$s. To this end, we employ the relational subsequence kernel [2] at each sequence position (over a time window of observations around the pivot position) for the classification model. We refer to this approach as Relational Subsequence Kernels for StructSVM approach (SubseqSVM).

We have compared our approach against TildeCRF [5], StructSVM [24] and enum$\mathcal{AF}$ [16]. While we treat StructSVM as a baseline for our experiments, TildeCRF is a state-of-the-art approach for learning relational features for sequence labeling, and operates in the same feature space that we are interested

in. In our experiments with StructSVM, individual basic features are assumed to be conditionally independent given the label.

The comparison of results on the UA dataset is outlined in Table 1. Results show that enum$\mathcal{AF}$ and our approach for learning complex features for sequence labeling *viz.* SubseqSVM performed better than the baseline approach (StructSVM) and the state-of-the-art approach (TildeCRF). Although enum$\mathcal{AF}$ optimally finds $\mathcal{CF}$s as conjunctions of (selectively enumerated) $\mathcal{AF}$s, the step for selectively enumerating $\mathcal{AF}$s is based on heuristics. In contrast, SubseqSVM works on a convex formulation and learns an optimal model. This explains the better performance of SubseqSVM.

The comparison of results on the KU dataset is outlined in Table 2. As a single sequence step in this data has only one input feature, the feature space is not rich enough to evaluate the efficiency of our approach. The baseline reported the best performance. While the performance of SubseqSVM approach is slightly inferior to the baseline and the state-of-the-art, enum$\mathcal{AF}$ performed poorly on this dataset.

In the case of the UA dataset, both enum$\mathcal{AF}$ and SubseqSVM took 24 hours approximately to train the model. In comparison, TildeCRF and StructSVM took 0.5 hours and 20 hours, respectively. On the KU data, enum$\mathcal{AF}$ took around 24 hours and SubseqSVM took approximately 1.5 hours to train the model. In comparison, TildeCRF and StructSVM took 10 minutes and 15 hours, respectively. We now present an analysis of the progression of results on UA data, using different categories of features we have experimented with.

	Micro avg.	Macro avg.
`tildeCRF`	56.22(±12.08)	35.36 (±6.55)
`StructSVM`	58.02 (±11.87)	35.00 (±05.24)
`enum`$\mathcal{AF}$	60.36 (±6.99)	30.39 (±4.31)
`SubseqSVM`	65.25(±4.81)	29.34 (±2.78)

Table 1: Micro average accuracy and macro average accuracy of classification in percentage using various approaches on UA data.

	Micro avg.	Macro avg.
`tildeCRF`	66.04 (±13.50)	84.01 (±8.76)
`StructSVM`	66.35 (±17.16)	66.64 (±16.04)
`enum`$\mathcal{AF}$	33.24 (±15.72)	23.02 (±11.13)
`SubseqSVM`	64.66 (±8.42)	63.08 (±7.05)

Table 2: Micro average accuracy and macro average accuracy of classification in percentage using various approaches on KU data.

The progression on experiments on UA data based on feature categories is shown in Table 3. The baseline for sequence labeling can be one among the approaches that assume conditional independence among individual features, given the label. HMM, CRF, and StructSVM falls into this category. These approaches consider input features at a sequence step and assumes conditional independence among them given the label. Since StructSVM is the state-of-the-art in this category, we use StructSVM results for comparison. The next level of features is the set of simple conjuncts $\mathcal{SC}$, which are conjunctions of input features at a single sequence step. $\mathcal{SC}$s capture relationships among co-occurring features. We present the StructHKL results for this. Next is the category of $\mathcal{CF}$s,

which are capable of capturing input relationships across time steps in sequence labeling. We present the results of SubseqSVM in this category.

Feature	Approach	Micro avg.	Macro avg.
Basic	StructSVM	58.02 (±11.87)	35.00 (±05.24)
$\mathcal{SC}$	StructHKL	63.96 (±05.74)	32.01 (±03.04)
$\mathcal{CF}$	SubseqSVM	65.25(±4.81)	29.34 (±2.78)

Table 3: Progression on sequence labeling experiments on the UA dataset based on feature categories.

5 Conclusion

Recent works have shown the importance of learning the input structure, in the form of relational features, for sequence labeling problems [13, 17, 12]. Most of the existing feature learning approaches employ greedy search techniques to discover relational features. In this work, we discussed approaches that looked into learning optimal relational features for sequence labeling. We identify that the relational feature space is exponentially large and therefore, learning explicit features of arbitrary complexity in our most general feature subspace, is a hard task. To this end, we presented an approach that learns relational sequence labeling models (capturing the richness of relational features implicitly) by leveraging relational subsequence kernels in the dual objective of the StructSVM framework. From our discussions and empirical analysis, we conclude that it is desirable to use powerful kernels that capture the relational features implicitly, although the resulting model may not be interpretable.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proc. of 20th Intl. Conf. on VLDB. pp. 487–499 (1994)
2. Bunescu, R., Mooney, R.J.: Subsequence kernels for relation extraction. In: Submitted to the Ninth Conference on Natural Language Learning (CoNLL-2005). Ann Arbor, MI (July 2006)
3. Dehaspe, L., Toivonen, H.: Discovery of frequent datalog patterns. Data Min. Knowl. Discov. 3(1), 7–36 (Mar 1999)
4. Forney, G.J.: The viterbi algorithm. Proceedings of IEEE 61(3), 268–278 (1973)
5. Gutmann, B., Kersting, K.: Tildecrf: conditional random fields for logical sequences. In: Proceedings of the 17th European conference on Machine Learning. pp. 174–185. ECML'06, Springer-Verlag, Berlin, Heidelberg (2006)
6. Joachims, T., Finley, T., Yu, C.N.J.: Cutting-plane training of structural svms. Mach. Learn. 77(1), 27–59 (Oct 2009)
7. van Kasteren, T., Noulas, A., Englebienne, G., Kröse, B.: Accurate activity recognition in a home setting. In: Proceedings of the 10th international conference on Ubiquitous computing. pp. 1–9. UbiComp '08, ACM, New York, NY, USA (2008)

8. Kersting, K., Raedt, L.D., Raiko, T.: Logical hidden markov models. Journal of Artificial Intelligence Research 25, 2006 (2006)
9. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data (2001), iCML
10. Landwehr, N., Gutmann, B., Thon, I., Raedt, L.D., Philipose, M.: Relational transformation-based tagging for activity recognition. Progress on Multi-Relational Data Mining 89(1), 111–129 (2009)
11. Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.: Text classification using string kernels. J. Mach. Learn. Res. 2, 419–444 (Mar 2002)
12. Mauro, N.D., Basile, T.M.A., Ferilli, S., Esposito, F.: Feature construction for relational sequence learning (2010)
13. McCallum, A., Li, W.: Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In: Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4. pp. 188–191. CONLL '03, Association for Computational Linguistics, Stroudsburg, PA, USA (2003)
14. McCallum, A.K.: Efficiently inducing features of conditional random fields (2003), proceedings of the Nineteenth Conference Annual Conference on Uncertainty in Artificial Intelligence
15. Miao, X., Rao, R.P.: Fast structured prediction using large margin sigmoid belief networks. Int. J. Comput. Vision 99(3), 302–318 (Sep 2012)
16. Nair, N., Nagesh, A., Ramakrishnan, G.: Probing the space of optimal markov logic networks for sequence labeling. In: Proceedings of the 22nd international conference on Inductive logic programming. Springer-Verlag, Berlin, Heidelberg (2012)
17. Nair, N., Ramakrishnan, G., Krishnaswamy, S.: Enhancing activity recognition in smart homes using feature induction. In: Proceedings of the 13th international conference on Data warehousing and knowledge discovery. pp. 406–418. DaWaK'11, Springer-Verlag, Berlin, Heidelberg (2011)
18. Nair, N., Saha, A., Ramakrishnan, G., Krishnaswamy, S.: Rule ensemble learning using hierarchical kernels in structured output spaces. In: AAAI (2012)
19. Rabiner, L.R.: Readings in speech recognition. chap. A tutorial on hidden Markov models and selected applications in speech recognition, pp. 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1990)
20. Schulte, O., Khosravi, H., Kirkpatrick, A., Man, T., Gao, T., Zhu, Y.: Modelling relational statistics with bayes nets. In: proceedings of 22nd International Conference on Inductive Logic Programming (ILP-2012). Springer (2012)
21. Taskar, B., Lacoste-Julien, S., Jordan, M.I.: Structured prediction, dual extragradient and bregman projections. J. Mach. Learn. Res. 7, 1627–1653 (Dec 2006)
22. Thon, I.: Don't fear optimality: sampling for probabilistic-logic sequence models. In: Proceedings of the 19th international conference on Inductive logic programming. pp. 226–233. ILP'09, Springer-Verlag, Berlin, Heidelberg (2010)
23. Thon, I., Landwehr, N., Raedt, L.: Stochastic relational processes: Efficient inference and applications. Mach. Learn. 82(2), 239–272 (Feb 2011)
24. Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y.: Support vector machine learning for interdependent and structured output spaces. In: Proceedings of the twenty-first international conference on Machine learning. pp. 104–. ICML '04, ACM, New York, NY, USA (2004)

Uplift Modeling with ROC: An SRL Case Study

Houssam Nassif, Finn Kuusisto, Elizabeth S. Burnside, and Jude Shavlik

University of Wisconsin, Madison, USA

Abstract. Uplift modeling is a classification method that determines the incremental impact of an action on a given population. Uplift modeling aims at maximizing the area under the uplift curve, which is the difference between the subject and control sets' area under the lift curve. Lift and uplift curves are seldom used outside of the marketing domain, whereas the related ROC curve is frequently used in multiple areas. Achieving a good uplift using an ROC-based model instead of lift may be more intuitive in several areas, and may help uplift modeling reach a wider audience.

We alter SAYL, an uplift-modeling statistical relational learner, to use ROC instead of lift. We test our approach on a screening mammography dataset. SAYL-ROC outperforms SAYL on our data, though not significantly, suggesting that ROC can be used for uplift modeling. On the other hand, SAYL-ROC returns larger models, reducing interpretability.

1 Introduction

Uplift modeling is a modeling and classification method initially used in marketing to determine the incremental impact of an advertising campaign on a given population [8]. Seminal work includes Radcliffe and Surry's true response modeling [8], Lo's true lift model [4], and Hansotia and Rukstales' incremental value modeling [3]. In some applications, especially medical decision support systems, gaining insight into the underlying classification logic can be as important as system performance. Reviewing the classification logic in medical problems can be an important method to discover disease patterns that may not be known or easily otherwise gleaned from the data. Such insight can be achieved using rule-learners. Decision trees [9, 10], inductive logic programming (ILP) [7], and statistical relational learning (SRL) [6] methods have been proposed.

Uplift modeling aims at maximizing uplift, which is the difference in a model or intervention M's lift scores over the subject and control sets:

$$Uplift_M = Lift_M(subject) - Lift_M(control). \quad (1)$$

Given a fraction ρ such that $0 \leq \rho \leq 1$, a model M's lift is defined as the number of positive examples amongst the model's ρ-highest ranking examples. Uplift thus captures the additional number of positive examples obtained due to the intervention. Quality of an uplift model is often evaluated by computing an uplift curve [9], generated by ranging ρ from 0 to 1 and plotting $Uplift_M$. The higher the uplift curve, the more profitable a marketing model/intervention is. The area under the uplift curve (AUU) is often used as a metric to optimize.

Let P be the number of positive examples and N the number of negative examples in a given dataset D. Lift represents the number of true positives detected by model m amongst the top-ranked fraction ρ. Varying $\rho \in [0,1]$ produces a lift curve. The area under the lift curve (AUL) for a given model and data becomes:

$$AUL = \int Lift(D,\rho)d\rho \approx \frac{1}{2}\sum_{k=1}^{P+N}(\rho_{k+1}-\rho_k)(Lift(D,\rho_{k+1})+Lift(D,\rho_k)) \quad (2)$$

Let s be the subject set, and c the controls. For a given ρ, we can rewrite equation 1 as:

$$Uplift_M(\rho) = Lift_M(s,\rho) - Lift_M(c,\rho). \quad (3)$$

Since uplift is a function of a single value for ρ, the area under the uplift curve (AUU) is the difference between the areas under the lift curves (AUL) for the subjects and the controls, $\Delta(AUL)$:

$$AUU = AUL_s - AUL_c = \Delta(AUL). \quad (4)$$

Lift and uplift curves are seldom used outside of the marketing domain, whereas the related ROC curve is frequently used in the machine learning and biomedical informatics communities. Especially in the biomedical domain, using ROC may be more intuitive, and may help uplift modeling reach a wider audience. This work investigates the use of the area under the ROC curve (AUR) as an alternate scoring method, while still resulting in a good model uplift. We alter SAYL [6], the state-of-the-art relational uplift modeling algorithm, to select rules that optimize $\Delta(AUR)$ instead of $\Delta(AUL)$. We test our approach on a screening mammography dataset.

2 Lift and ROC Area Under the Curve

There is a strong connection between AUL and AUR. Let $\pi = \frac{P}{P+N}$ be the prior probability for the positive class or skew, then:

$$AUL = P * (\frac{\pi}{2} + (1-\pi)\ AUR)\ [11, p.549]. \quad (5)$$

Uplift modeling aims at optimizing uplift, the difference in lift over two sets. It constructs a new classifier such that:

$$\Delta(AUL^*) > \Delta(AUL) \quad (6)$$

As discussed in [6], by expanding and simplifying we get:

$$AUL_s^* - AUL_c^* > AUL_s - AUL_c$$
$$P_s(\frac{\pi_s}{2} + (1-\pi_s)AUR_s^*) - P_c(\frac{\pi_c}{2} - (1-\pi_c)AUR_c^*) >$$
$$P_s(\frac{\pi_s}{2} + (1-\pi_s)AUR_s) - P_c(\frac{\pi_c}{2} - (1-\pi_c)AUR_c)$$
$$P_s(1-\pi_s)AUR_s^* - P_c(1-\pi_c)AUR_c^* > P_s(1-\pi_s)AUR_s - P_c(1-\pi_c)AUR_c$$
$$P_s(1-\pi_s)(AUR_s^* - AUR_s) > P_s(1-\pi_s)(AUR_c^* - AUR_c)$$

and finally

$$\frac{AUR_s^* - AUR_s}{AUR_c^* - AUR_c} > \frac{P_c}{P_s}\frac{1-\pi_c}{1-\pi_s}. \tag{7}$$

In a balanced dataset, we have $\pi_c = \pi_s = \frac{1}{2}$ and $P_c = P_s$, so we have that $\frac{P_c}{P_s}\frac{1-\pi_c}{1-\pi_s} = 1$. If the subject and control sets have the same numbers and skew, we can conclude that $\Delta(AUL^*) > \Delta(AUL)$ implies $\Delta(AUR^*) > \Delta(AUR)$. If the two sets are skewed or their numbers differ, we cannot guarantee that $\Delta(AUL^*) > \Delta(AUL)$ implies $\Delta(AUR^*) > \Delta(AUR)$, as we can increase uplift with rules that have similar accuracy but cover more cases in the positive set. In general, the two metrics are related, with uplift being more sensitive to variations in coverage when the two groups have different size.

3 SAYL-ROC

SAYL [6] is a Statistical Relational Learner based on SAYU [1] that integrates uplift modeling with the search for relational rules. Similar to SAYU, every valid rule generated is used to construct a Bayesian network (alongside with current theory rules) via propositionalization, but instead of constructing a single classifier, SAYL constructs two TAN [2] classifiers; one Bayes net for each of the subject and control groups. Both classifiers use the same set of attributes, but are trained only on examples from their respective groups. SAYL uses the TAN generated probabilities to construct the lift and uplift curves. If a rule improves AUU by threshold θ, the rule is added to the attribute set. Otherwise, SAYL continues the search.

Algorithm 1 SAYL

```
Rs ← {}; M_0^s, M_0^c ← InitClassifiers(Rs)
while DoSearch() do
    e_s^+ ← RandomSeed();
    ⊥_{e_s^+} ← saturate(e);
    while c ← reduce(⊥_{e_s^+}) do
        M^s, M^c ← LearnClassifiers(Rs ∪ {c});
        if Better(M^s, M^c, M_0^s, M_0^c) then
            Rs ← Rs ∪ {c}; M_0^s, M_0^c ← M^s, M^c;
            break
        end if
    end while
end while
```

The SAYL algorithm is shown as Algorithm 1. SAYL maintains a current set of clauses, Rs, and current reference classifiers for the subjects M^s and controls M^c. SAYL requires separate training and tuning sets, accepting a rule only when it improves the score on both sets. This requirement is extended with the threshold of improvement θ, and a minimal rule coverage requirement *minpos*. Finally, SAYL has two search modes, greedy and exploration. Refer to [6] for details.

SAYL guides the rule search by using the AUU score. It computes AUU by computing AUL for each of the groups using the two classifiers, and returning the difference $\Delta(AUL)$ (Equation 4). We implement SAYL-ROC, a SAYL variant that computes AUR instead for each of the groups using the two classifiers, and returns $\Delta(AUR)$ as a rule score to guide the search. SAYL thus optimizes for $\Delta(AUL)$, while SAYL-ROC optimizes for $\Delta(AUR)$.

4 Experimental Results

We test SAYL-ROC on a breast cancer mammography dataset, fully described in [5]. Our subject and control sets are respectively older and younger patients with confirmed breast cancer. Positive instances have in situ cancer, and negative instances have invasive cancer. The aim is to maximize the in situ cases' uplift.

The older cohort has 132 in situ and 401 invasive cases, while the younger one has 110 in situ and 264 invasive. The skews are $P_s = 132$, $\pi_s = \frac{132}{132+401}$ (older), and $P_c = 110$, $\pi_c = \frac{110}{110+264}$ (younger). Thus equation 7 becomes:

$$\frac{AUR^*_s - AUR_s}{AUR^*_c - AUR_c} > 0.86. \tag{8}$$

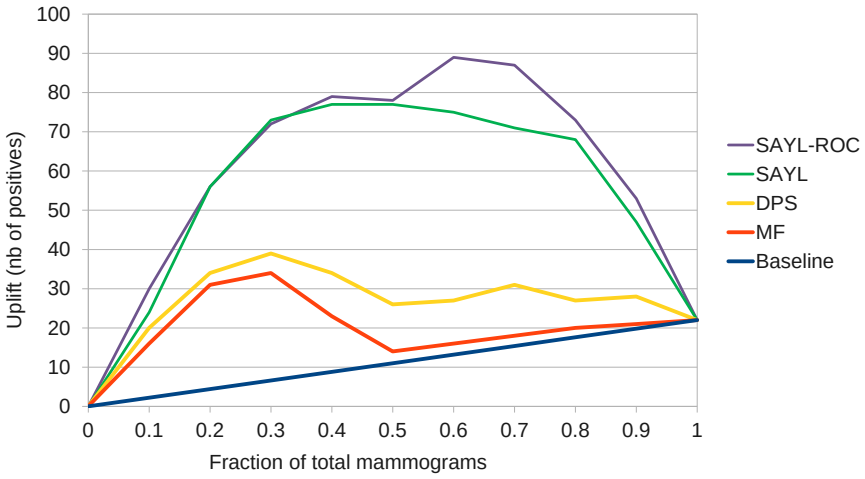
We use 10-fold cross-validation, making sure all records pertaining to the same patient are in the same fold. We run both SAYL and SAYL-ROC with a time limit of one hour per fold. For each cross-validated run, we use 4 training, 5 tuning and 1 testing folds. For each fold, we used the best combination of parameters according to a 9-fold internal cross-validation using 4 training, 4 tuning and 1 testing folds. We try both search modes, vary *minpos* between 7 and 13 (respectively 5% and 10% of older in situ examples), and set θ to 1%, 5% and 10%. We evaluate the final SAYL and SAYL-ROC models using their final uplift curves, concatenated from the results of each testing set.

Table 1. 10-fold cross-validated SAYL-ROC and SAYL performance. Rule number averaged over the 10 folds of theories. For comparison, we include results of Differential Prediction Search (DPS) and Model Filtering (MF) methods [7]. We compute the p-value comparing each method to SAYL, * indicating significance.

Algorithm	AUU	AUL_s	AUL_c	Rules Avg	p-value
SAYL-ROC	62.99	95.64	32.65	24.7	0.4316
SAYL	58.10	97.24	39.15	9.3	-
DPS	27.83	101.01	73.17	37.1	0.0020 *
MF	20.90	100.89	80.99	19.9	0.0020 *
Baseline	11.00	66.00	55.00	-	0.0020 *

Table 1 compares SAYL-ROC, SAYL, and the ILP-based methods Differential Prediction Search (DPS) and Model Filtering (MF) [7], both of which had *minpos* = 13 (10% of older in situ). A baseline random classifier achieves an AUU of 11. We use the paired Mann-Whitney test at the 95% confidence level to compare two sets of experiments. We plot the uplift curves in Figure 1.

Fig. 1. Uplift curves for SAYL-ROC, SAYL, ILP methods Differential Prediction Search (DPS) and Model Filtering (MF), both with *minpos* = 13 [7], and baseline random classifier. Uplift curves start at 0 and end at 22, the difference between older (132) and younger (110) total in situ cases. The higher the curve, the better the uplift.



5 Discussion and Future Work

SAYL and SAYL-ROC significantly outperform previous methods (Table 1, Figure 1), but there is no significant difference between the two. Even though SAYL-ROC is optimizing for $\Delta(AUR)$ during its training phase, it returns a slightly better testing $\Delta(AUL)$ than SAYL, which optimizes for $\Delta(AUL)$.

This result suggests that, on a moderately subject/control skewed data, AUR can indeed be used for uplift modeling. ROC is more frequently used than lift, and may be more intuitive in many domains. Nevertheless, more experiments are needed to establish ROC-based uplift performance. We plan on measuring $\Delta(AUL)$ vs. $\Delta(AUR)$ for various Equation 7 skews.

SAYL-ROC produces as many rules as ILP-based methods, more than twice that of SAYL. The ILP theory is a collection of independent rules that each individually increases uplift [7]. It is thus easy to interpret the final model. SAYL and SAYL-ROC theory rules are conditioned on each other as nodes in a Bayesian network, decreasing rule interpretability especially in larger graphs. Individual rules may not increase uplift, but the final network does. At an average of 9.3 rules, a SAYL model is interpretable, whereas at 24.7, SAYL-ROC sacrifices interpretability.

We note that Equation 7 depends on both the positive number and skew. Even if the subject and control positive skews were equal, say $P_c = 100$, $N_c = 200$, $P_s = 10$ and $N_s = 20$, we will have $\frac{1-\pi_c}{1-\pi_s} = 1$ but $\frac{P_c}{P_s} = 10$, maintaining a subject/control Equation 7 skew.

This work uses the definition of lift as the *number* of positives amongst the ρ-highest ranking examples. An alternative lift definition is the *fraction* of positives amongst the ρ-highest ranking examples. Equation 7 then becomes:

$$\frac{AUR_s^* - AUR_s}{AUR_c^* - AUR_c} > \frac{1 - \pi_c}{1 - \pi_s}, \tag{9}$$

eliminating the dependence on the number of positive instances. We plan on investigating how $\Delta(AUL)$ and $\Delta(AUR)$ empirically relate under this definition.

In conclusion, SAYL-ROC exhibits a similar performance to SAYL on our data, suggesting that ROC can be used for uplift modeling. SAYL-ROC returns larger models, reducing interpretability. More experiments are needed to test ROC-based uplift over different subject/control skews.

Acknowledgments We thank NIH grant R01-CA165229, the Carbone Cancer Center, and NCI grant P30CA014520 for support.

References

1. Davis, J., Burnside, E.S., de Castro Dutra, I., Page, D., Santos Costa, V.: An integrated approach to learning Bayesian Networks of rules. In: Proceedings of the 16th European Conference on Machine Learning. pp. 84–95. Porto, Portugal (2005)
2. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine Learning 29, 131–163 (1997)
3. Hansotia, B., Rukstales, B.: Incremental value modeling. Journal of Interactive Marketing 16(3), 35–46 (2002)
4. Lo, V.S.: The true lift model - a novel data mining approach to response modeling in database marketing. SIGKDD Explorations 4(2), 78–86 (2002)
5. Nassif, H., Page, D., Ayvaci, M., Shavlik, J., Burnside, E.S.: Uncovering age-specific invasive and DCIS breast cancer rules using Inductive Logic Programming. In: ACM International Health Informatics Symposium (IHI). pp. 76–82. Arlington, VA (2010)
6. Nassif, H., Kuusisto, F., Burnside, E.S., Page, D., Shavlik, J., Santos Costa, V.: Score as you lift (SAYL): A statistical relational learning approach to uplift modeling. In: European Conference on Machine Learning (ECML-PKDD). pp. 595–611. Prague (2013)
7. Nassif, H., Santos Costa, V., Burnside, E.S., Page, D.: Relational differential prediction. In: European Conference on Machine Learning (ECML-PKDD). pp. 617–632. Bristol, UK (2012)
8. Radcliffe, N.J., Surry, P.D.: Differential response analysis: Modeling true response by isolating the effect of a single action. In: Credit Scoring and Credit Control VI. Edinburgh, Scotland (1999)
9. Radcliffe, N.J., Surry, P.D.: Real-world uplift modelling with significance-based uplift trees. White Paper TR-2011-1, Stochastic Solutions (2011)
10. Rzepakowski, P., Jaroszewicz, S.: Decision trees for uplift modeling with single and multiple treatments. Knowledge and Information Systems 32, 303–327 (2012)
11. Tufféry, S.: Data Mining and Statistics for Decision Making. John Wiley & Sons, 2nd edn. (2011)

Learning the Parameters of Probabilistic Description Logics

Fabrizio Riguzzi[1], Elena Bellodi[2], Riccardo Zese[2], and Evelina Lamma[2]

[1] Dipartimento di Matematica e Informatica – University of Ferrara
Via Saragat 1, I-44122, Ferrara, Italy
[2] Dipartimento di Ingegneria – University of Ferrara
Via Saragat 1, I-44122, Ferrara, Italy
{fabrizio.riguzzi,elena.bellodi,evelina.lamma,riccardo.zese}@unife.it

Abstract. Uncertain information is ubiquitous in the Semantic Web, due to methods used for collecting data and to the inherently distributed nature of the data sources. It is thus very important to develop probabilistic Description Logics (DLs) so that the uncertainty is directly represented and managed at the language level. The DISPONTE semantics for probabilistic DLs applies the distribution semantics of probabilistic logic programming to DLs. In DISPONTE, axioms are labeled with numeric parameters representing their probability. These are often difficult to specify or to tune for a human. On the other hand, data is usually available that can be leveraged for setting the parameters. In this paper, we present EDGE that learns the parameters of DLs following the DISPONTE semantics. EDGE is an EM algorithm in which the required expectations are computed directly on the binary decision diagrams that are built for inference. Experiments on two datasets show that EDGE achieves higher areas under the Precision Recall and ROC curves than an association rule learner in a comparable or smaller time.

1 Introduction

Due to the ubiquity of uncertain information, many authors [9, 17, 8] have recently studied approaches to add uncertainty to the Semantic Web. Since Description Logics (DLs) are at the basis of the Semantic Web, in [12] we proposed the DISPONTE ("DIstribution Semantics for Probabilistic ONTologiEs", Spanish for "get ready") semantics. DISPONTE applies the distribution semantics of probabilistic logic programming [15] to DLs.

In [14] we presented an algorithm, called EDGE for "Em over bDds for description loGics paramEter learning", for learning the parameters of probabilistic DLs that follow the DISPONTE semantics. EDGE starts from examples of instances and non-instances of concepts and builds a set of Binary Decision Diagrams (BDDs) that represent their explanations. The parameters are then tuned using an EM algorithm [6] in which the required expectations are computed directly on the BDDs. In [14] the parameters learned by EDGE were compared with those given by the confidence of Association Rules (ARs for short in

the following) on a dataset extracted from `educational.data.gov.uk`. EDGE achieved significantly higher areas under the Precision Recall and the Receiver Operating Characteristics curves (AUCPR and AUCROC).

In this paper we extend the experiments presented in [14] by also considering a dataset extracted from DBPedia and by recording the time required by EDGE and by the computation of ARs' confidence. EDGE achieves again higher AUCPR and AUCROC. Moreover, the time taken by EDGE is comparable to the one required for computing ARs' confidence.

The paper is organized as follows. Section 2 introduces DLs and the DISPONTE semantics while Section 3 introduces EDGE. Section 4 discusses related works and Section 5 shows the results of experiments. Section 6 concludes the paper.

2 Description Logics and the DISPONTE semantics

DLs are particularly useful for representing ontologies and have been adopted as the basis of the Semantic Web. They are usually represented using a syntax based on concepts and roles. A concept corresponds to a set of individuals of the domain while a role corresponds to a set of couples of individuals of the domain. In the following we consider and describe $\mathcal{ALC}$ [16].

We use $\mathbf{A}$, $\mathbf{R}$ and $\mathbf{I}$ to indicate *atomic concepts*, *atomic roles* and *individuals*, respectively. A *role* is an atomic role $R \in \mathbf{R}$. *Concepts* are defined as follows. Each $A \in \mathbf{A}$, $\bot$ and $\top$ are concepts. If C, C_1 and C_2 are concepts and $R \in \mathbf{R}$, then $(C_1 \sqcap C_2)$, $(C_1 \sqcup C_2)$ and $\neg C$ are concepts, as well as $\exists R.C$ and $\forall R.C$.

Let C and D be concepts, R be a role and a and b be individuals, a *TBox* $\mathcal{T}$ is a finite set of *concept inclusion axioms* $C \sqsubseteq D$, while an *ABox* $\mathcal{A}$ is a finite set of *concept membership axioms* $a : C$ and *role membership axioms* $(a, b) : R$. A *knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$.

A KB is usually assigned a semantics using interpretations of the form $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty *domain* and $\cdot^{\mathcal{I}}$ is the *interpretation function* that assigns an element in $\Delta^{\mathcal{I}}$ to each individual a, a subset of $\Delta^{\mathcal{I}}$ to each concept C and a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to each role R. The mapping $\cdot^{\mathcal{I}}$ is extended to all concepts (where $R^{\mathcal{I}}(x) = \{y | (x, y) \in R^{\mathcal{I}}\}$ and $\#X$ denotes the cardinality of the set X) as:

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}} \qquad \bot^{\mathcal{I}} = \emptyset$$
$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \qquad (C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$$
$$(C_1 \sqcup C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}} \qquad (\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | R^{\mathcal{I}}(x) \subseteq C^{\mathcal{I}}\}$$
$$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | R^{\mathcal{I}}(x) \cap C^{\mathcal{I}} \neq \emptyset\}$$

A query over a KB base is usually an axiom for which we want to test the entailment from the KB. The entailment test may be reduced to checking the unsatisfiability of a concept in the KB, i.e., the emptiness of the concept.

DISPONTE applies the distribution semantics to probabilistic ontologies [15]. In DISPONTE [2, 10–13] a *probabilistic knowledge base* $\mathcal{K}$ is a set of certain and probabilistic axioms. *Certain axioms* take the form of regular DL axioms.

Probabilistic axioms take the form $p :: E$, where p is a real number in $[0, 1]$ and E is a DL axiom. The idea of DISPONTE is to associate independent Boolean random variables with the axioms. Thus, a single random variable is associated with axiom E and p represents its probability of being true.

A DISPONTE KB defines a distribution over regular DL KB called *worlds*. Each world is obtained by including every certain axiom. For each probabilistic axiom, we decide whether or not to include it in the world. By multiplying the probability of the choices made to obtain a world we can assign a probability to it. The probability of a query is then the sum of the probabilities of the worlds where the query holds true.

3 EDGE

EDGE [14] is based on the algorithm EMBLEM [4, 3] developed for learning the parameters for probabilistic logic programs under the distribution semantics. EDGE adapts EMBLEM to the case of probabilistic DLs under the DISPONTE semantics. EDGE takes as input a DL KB and a number of positive and negative examples that represent the queries in the form of concept assertions, i.e., of the form $a : C$ for an individual a and a class C. Positive examples represent information that we regard as true and for which we would like to get high probability while negative examples represent information that we regard as false and for which we would like to get low probability.

EDGE first computes, for each example, the BDD encoding its explanations using the reasoner BUNDLE [13]. For a positive example of the form $a : C$, EDGE looks for the explanations of $a : C$ and encodes them in a BDD. For a negative example of the form $a : C$, EDGE looks for the explanations of $a : C$, encodes them in a BDD and negates it with the NOT BDD operator. Then EDGE enters the EM cycle, in which the steps of Expectation and Maximization are repeated until the log-likelihood (LL) of the examples reaches a local maximum or until the maximum number of iterations is reached. The EM algorithm is guaranteed to find a local maximum, which however may not be the global maximum. The LL of the examples is guaranteed to increase at each iteration.

Function Expectation takes as input a BDD for each example Q, and computes $P(X_i = x|Q)$ for all the variables X_i in the BDD. Finally, it returns the LL of the data that is used in the stopping criterion: EDGE stops when the difference between the LL of the current iteration and that of the previous one drops below a threshold ϵ or when this difference is below a fraction δ of the previous LL. Function Maximization computes the parameters' values for the next EM iteration by relative frequency. For more details see [4].

The phase of explanations research for each example has high complexity in the worst case, since the explanations may grow exponentially in number; however, BUNDLE is able to handle domains of significant size. The EM phase has a linear cost in the number of nodes since the E-step requires two traversals of the diagram.

4 Related Work

CR$\mathcal{ALC}$ [9] is an extension of $\mathcal{ALC}$ that adopts an interpretation-based semantics for allowing statistical axioms of the form $P(C|D) = \alpha$ (for each element x in $\mathcal{D}$ that belongs to D, the probability that belongs also to C is α) and of the form $P(R) = \beta$ (for each couple of elements x and y in $\mathcal{D}$, the probability that x is linked to y by the role R is β). On the other hand, CR$\mathcal{ALC}$ does not allow to express a degree of belief in axioms. A CR$\mathcal{ALC}$ KB $\mathcal{K}$ can be represented as a directed acyclic graph $G(\mathcal{K})$ in which a node represents a concept or a role and the edges represent the relations between them. The algorithm of [9] learns parameters and structure of CR$\mathcal{ALC}$ knowledge bases. It starts from positive and negative examples for a single concept and learns the best probabilistic definition for the concept chosen using an EM algorithm. Differently for us, the expected counts are computed by resorting to inference in the graph, while we exploit the BDD structures.

GoldMiner [17, 8] is an algorithm that exploits ARs for building ontologies. GoldMiner extracts information about individuals, named classes and roles using SPARQL queries. From these data, it builds two *transaction tables*: one that stores the classes to which each individual belongs and one that stores the roles to which each couple of individuals belongs. Finally, the APRIORI algorithm [1] is applied to each table in order to find ARs. Implications of the form $A \Rightarrow B$ can be converted to subclass axioms of the form $A \sqsubseteq B$. Moreover, the confidence associated with ARs can be interpreted as the probability of the axiom $p :: A \sqsubseteq B$. So GoldMiner can be used to obtain a probabilistic knowledge base.

5 Experiments

EDGE has been compared with ARs over two real world datasets from the Linked Open Data cloud: `educational.data.gov.uk` and an extract of DBPedia. In our experiment, we wanted to simulate the situation in which an expert provides the structure of the ontology together with information on a set of individuals. The ontologies were obtained with GoldMiner: we extracted 10,000 individuals for `educational.data.gov.uk` and 7,200 for DBPedia and we learned ARs from the resulting transaction tables. The ARs were then converted into subclass axioms.

In order to generate a set of examples for EDGE, for each extracted individual a we sampled three named classes: A and B were sampled from the named classes to which a explicitly belonged, while C was sampled from the named classes to which a did not explicitly belong but that exhibited at least one explanation for the query $a : C$. Then, we randomly split individuals into two equally sized sets: the membership assertions regarding the individuals from the first set constituted the training set while the ones in the second set constituted the testing set. The axiom $a : A$ is added to the KB, while $a : B$ is considered as a positive example and $a : C$ as a negative example. The training set contained only the membership assertions for the first set of individuals, while for the testing phase

we removed the membership assertions of the training set from the KB and added the assertions of the second set.

We compared the parameters learned by EDGE with ARs' confidence. For each AR corresponding to the subclass axiom $A \sqsubseteq B$, we computed the confidence by running two SPARQL queries over the training KBs, one for finding all the individuals that belong to $A \sqcap B$ and one for those that belong to A. The confidence is then given by the ratio of the number of individuals in $A \sqcap B$ over those in A. We created 330 different SPARQL queries for `educational.data.gov.uk` and 2,243 for DBPedia.

Next we ran EDGE over the KBs where all the subclass axioms were assigned an initial random probability. We then computed the probability of the examples in the testing set according to the theory learned by EDGE and to the theory composed of the ARs with the confidence as probability. We drew the Precision-Recall and the Receiver Operating Characteristics curves and computed the Area Under the Curve (AUCPR and AUCROC) following the methods of [5, 7]. Table 1 shows the AUCPR, the AUCROC and the execution times (in seconds). Note that the elapsed time for EDGE depends on the number of executed queries and the number of different explanations involved in each query, while the elapsed time for ARs depends on the number of classes in the KB. EDGE achieves much higher areas in a time that is of the same or lower order of magnitude with respect to ARs.

Areas Under Curves		EDGE	ARs
educational.data.gov.uk	AUCPR	0.9702	0.8804
	AUCROC	0.9796	0.9158
	Time (s)	65,170	10,490
DBPedia	AUCPR	0.9784	0.5916
	AUCROC	0.9902	0.4346
	Time (s)	50,800	578,420

Table 1. Resulting AUCPR, AUCROC and execution times.

6 Conclusions

EDGE applies an EM algorithm for learning the parameters of probabilistic knowledge bases under the DISPONTE semantics. It exploits the BDDs that are built during inference to efficiently compute the expectations for hidden variables. EDGE is available for download from `http://sites.unife.it/ml/edge`. The experiments over two real world datasets show that EDGE achieves larger areas both under the PR and the ROC curve with respect to an algorithm based on ARs in a comparable or smaller time, thus demonstrating that EDGE is a viable alternative to ARs.
We plan to extend EDGE for learning the structure together with the parameters.

7 Acknowledgements

Elena Bellodi is partially supported by Gruppo Nazionale per il Calcolo Scientifico, Istituto Nazionale di Alta Matematica "F. Severi".

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB 1994. pp. 487–499. Morgan Kaufmann (1994)
2. Bellodi, E., Lamma, E., Riguzzi, F., Albani, S.: A distribution semantics for probabilistic ontologies. In: URSW 2011. CEUR Workshop Proceedings, vol. 778. Sun SITE Central Europe (2011)
3. Bellodi, E., Riguzzi, F.: Experimentation of an expectation maximization algorithm for probabilistic logic programs. Intelligenza Artificiale 8(1), 3–18 (2012)
4. Bellodi, E., Riguzzi, F.: Expectation Maximization over binary decision diagrams for probabilistic logic programs. Intelligent Data Analysis 17(2), 343–363 (2013)
5. Davis, J., Goadrich, M.: The relationship between precision-recall and ROC curves. In: ICML 2006. pp. 233–240. ACM (2006)
6. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. Royal Statistical Society. Series B 39(1), 1–38 (1977)
7. Fawcett, T.: An introduction to ROC analysis. Pattern Recognition Letters 27(8), 861–874 (2006)
8. Fleischhacker, D., Völker, J.: Inductive learning of disjointness axioms. In: OTM Conferences 2011. LNCS, vol. 7045, pp. 680–697. Springer (2011)
9. Luna, J.E.O., Revoredo, K., Cozman, F.G.: Learning probabilistic description logics: A framework and algorithms. In: MICAI 2011. LNCS, vol. 7094, pp. 28–39. Springer (2011)
10. Riguzzi, F., Bellodi, E., Lamma, E.: Probabilistic Datalog+/- under the distribution semantics. In: DL 2012. CEUR Workshop Proceedings, vol. 846. Sun SITE Central Europe (2012)
11. Riguzzi, F., Bellodi, E., Lamma, E.: Probabilistic ontologies in Datalog+/-. In: CILC 2012. CEUR Workshop Proceedings, vol. 857, pp. 221–235. Sun SITE Central Europe (2012)
12. Riguzzi, F., Bellodi, E., Lamma, E., Zese, R.: Epistemic and statistical probabilistic ontologies. In: URSW 2012. CEUR Workshop Proceedings, vol. 900, pp. 3–14. Sun SITE Central Europe (2012)
13. Riguzzi, F., Bellodi, E., Lamma, E., Zese, R.: BUNDLE: A reasoner for probabilistic ontologies. In: Faber, W., Lembo, D. (eds.) RR 2013. LNCS, vol. 7994, pp. 183–197. Springer (2013)
14. Riguzzi, F., Bellodi, E., Lamma, E., Zese, R.: Parameter learning for probabilistic ontologies. In: Faber, W., Lembo, D. (eds.) RR 2013. LNCS, vol. 7994, pp. 265–270. Springer (2013)
15. Sato, T.: A statistical learning method for logic programs with distribution semantics. In: ICLP 1995. pp. 715–729 (1995)
16. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. Artificial Intelligence 48(1), 1–26 (1991)
17. Völker, J., Niepert, M.: Statistical schema induction. In: ESWC 2011. LNCS, vol. 6643, pp. 124–138. Springer (2011)

Predicting Top-k Trends on Twitter using Graphlets and Time Features

Gustav Šourek, Ondřej Kuželka, and Filip Železný

Faculty of Electrical Engineering, Czech Technical University in Prague
Technicka 2, 16627 Prague, Czech Republic
{souregus,kuzelon2,zelezny}@fel.cvut.cz,

Abstract. We introduce a novel method for predicting trending keywords on Twitter. This new method exploits topology of the studied parts of the social network. It is based on a combination of graphlet spectra and so-called time features. We show experimentally that using graphlets and time features is beneficial for the accuracy of prediction.

1 Introduction

In this paper, we present a method which exploits information about graph structure of sub-networks of the social network Twitter for making better predictions about which topics will get among the top k. We show experimentally that with information about the graph structure we are able to obtain better predictive accuracies than with a model trained on the same data which does not use information about the graph structure. Importantly, the presented method does not need access to the entire graph structure as it works only with certain sets of derived attributes, which makes it potentially possible to combine the method with sampling strategies and also to make it able to work in differentially private settings.

As the network structure has been proved to play an important role in spreading social trends [1], we want to exploit the effect of social network topology beyond the scope of previous works (e.g. beyond merely measuring nodes' degrees, centrality etc.). Inspired by creating network signatures from graphlet degree distributions [2] in biological networks, we use similar representation to reflect a trend presence within our network. For that we create graphlet features - small connected subgraphs, representing various local relative topology options, and measure their presence in the network by means of subgraph matching. The network trend signatures, calculated from the frequencies of respective features occurrences, are then used as feature vectors for standard machine learning algorithms.

2 Problem Setting

Twitter is an online social networking service that enables its users to send and read text-based messages of up to 140 characters known as *tweets*. At the same

time it enables users to connect to others through the *follows* relationship. The users that a particular user is following through this relation are referred as his friends. Users on the other side who are following the particular user are referred simply as his followers. Tweets posted by a particular user are stored and displayed as a chronological sequence in user's timeline. Each such a tweet being posted is also broadcasted to the users followers. Tweets are, by default, public, which means that anyone can list them out through Twitter's search engine or other Twitter API facilities and join the related conversation. Moreover Twitter users can engage in a direct conversation between each other. As for the information content, users can group posts together by type with the use of *hashtags* - words or phrases prefixed with a "#" sign, referring a tweet to the specified topic. Hashtag signed tweets have special treatment in Twitter's engine and can be easily searched out.

Now, we define the prediction problem that we will be dealing with in this paper, namely the problem of predicting the top-k trends. Unlike original Twitter engine, we consider trending topics clearly by measuring the frequency of occurrence of corresponding hashtag in the network. If the relative frequency of hashtag in a particular timeframe is among the top-k, we declare it a trend. One can imagine a web page which gives its users a list of k hashtags which are predicted to get among the hottest topics in his subnetwork in the near future.

Learning is performed on data as a time series, where each hashtag occurrence information goes into a prepared *time-fold* according to its time of creation. The task is to predict which hashtags will be trending in the future target time-folds (determined to day intervals). Unlike in the case of the basic supervised learning task, there is an additional constraint on the output of the classifier. On every single day, the classifier must mark as trending exactly k hashtags. To satisfy this constraint, the classifier takes the probability distribution of classification given by the learned Random Forest [3] model and creates respective ranking on the instances that are subject to the current prediction. That means that only the top k instances classified with highest confidence as trending will be considered positive. This k-set will then be compared with the true top-k list for the target day. The natural measure of quality of such a prediction, denoted as top-k% metric here, is the percentage of correctly predicted topics in the target list.

3 Simple and Baseline Models

In order to assess the contribution of graphlets and time features to accuracy of prediction, we created two models which we call *simple model* and *baseline model* and which serve as a baseline in this paper. In these approaches we wittingly ignore the social graph structure and take the problem as a time series prediction, which is the case of most methods found in literature. The simple model method represents the common sense approach to the statistics measured. It builds on basic average occurrence of the hashtags, calculated over all time-folds in the training part of the time-window, and treats them as if they were to

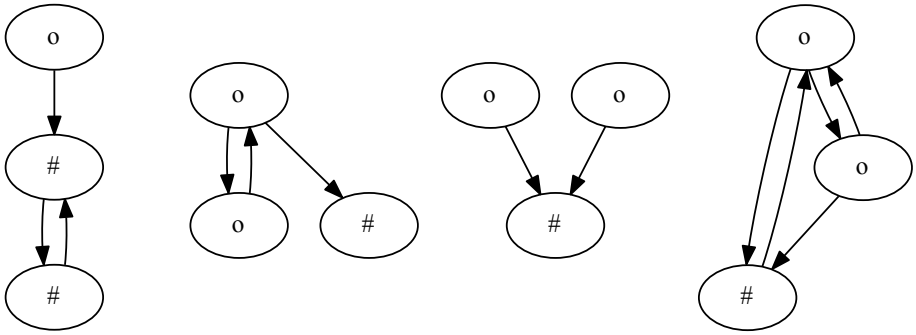


Fig. 1. Selected examples of features of size = 3, with some of the highest information gains with respect to the trends spread.

continue constantly with that occurrence in the target part. The baseline learner represents a classical time-series prediction method, with the use of a model to predict future values based on previously observed values in data with natural temporal ordering. To make a clear comparison of the models used, we turn the time series forecasting task into a standard classification problem, using a sliding window technique [4]. Since our main focus lies with the features extracted, it always uses the same machine learning algorithm as the graph learner.

4 A Model Based on Graphlets and Time Features

The term *graphlet*, as used in this paper, refers to a small directed graph (with up to 3 nodes) which contains at least one node labelled by "#". Examples of graphlets are shown in Figure 1. For every day D and every hashtag H, the *snapshot* of the sub-network is a directed labelled graph constructed as follows. There is one node for every user. There is a directed edge between two users U_i and U_j if and only if the user U_i *follows* the user U_j. A node is labelled by "#" if and only if the user corresponding to this node used the hashtag H on the day D in at least one tweet. Given a list of graphlets L_g and a snapshot S of the sub-network we can construct a so-called *frequency-feature vector* as follows. For every graphlet $g_i \in L_g$, we count the number of homomorphisms of g_i to the snapshot graph S (respecting the labels "#") and store the number in the i-th element of the frequency-feature vector. Clearly, frequency-feature vectors are not very suitable for prediction of top-k trends because their values are sensitive to the overall activity of the users on the given day. Therefore we need so-called *rank-feature vectors* which can be constructed from the frequency-feature vectors. Given a set of frequency-feature vectors for all hashtags of interest on a given day, the i-th element of a rank-feature vector V_{rank} for a hashtag H is the rank (i.e. order) of the respective i-th element of the frequency-feature vector V_{freq} corresponding to the same H among all i-th elements of the other frequency-feature vectors corresponding to the same hashtag H. Given a time

window consisting of several days, one can easily create a graphlet representation by concatenating the rank-feature vectors corresponding to these days.

The rationale behind the graphlet model is that graphlets can capture how natural user to user connections in Twitters network affect the topics discussed. They consist of nodes and edges representing occurrence of hashtag on users time-line in context of his neighbors, e.g. his followers and friends. A somewhat similar representation was used in the work of Nataša Pržulj for computing a network structure similarity measure using so called graphlet degree distribution [2], where graphlets were small connected non-isomorphic induced subgraphs of a large network. Our relational approach differs in that, with our features, we generate the subgraphs separately, in advance of further matching in the whole network, while we do not restrict them to be induced. Even more importantly, as far as we know, our approach is the first to use graphlets to model dynamical processes in complex networks.

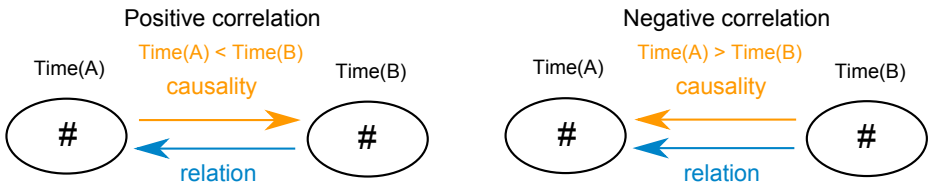


Fig. 2. The two cases of correspondence between the relation and causality in hashtag spreading.

Besides graphlets we use also so-called *time features*. The main purpose of time features is to add a measure of some time properties of the underlying networks relations. The motivation for this comes from a natural intuition of trends spread in social networks. In a directed network like Twitter, if the information is being spread through the network, the fashion of the spreading should correspond to the network structure. By that we mean that the directed relations between the users should actually represent the causality links in the trend spread dynamics. If it is not the case, the information is probably not coming from the network and is being spread by other channels. Now how to measure this networks causality correspondence? For a snapshot corresponding to a hashtag H, we label all the directed edges $e = (U_i, U_j)$ which connect two nodes U_i and U_j both of which are labelled by "#", by the time between the instant when U_i posted the tweet with the hashtag H and the instant when U_j posted a tweet with the same hashtag. Since the difference is measured against the direction of the underlying relation, it can in general be negative as depicted in Figure 2. The time features can then be computed as averages and standard deviations of these time differences by which the edges are labelled.

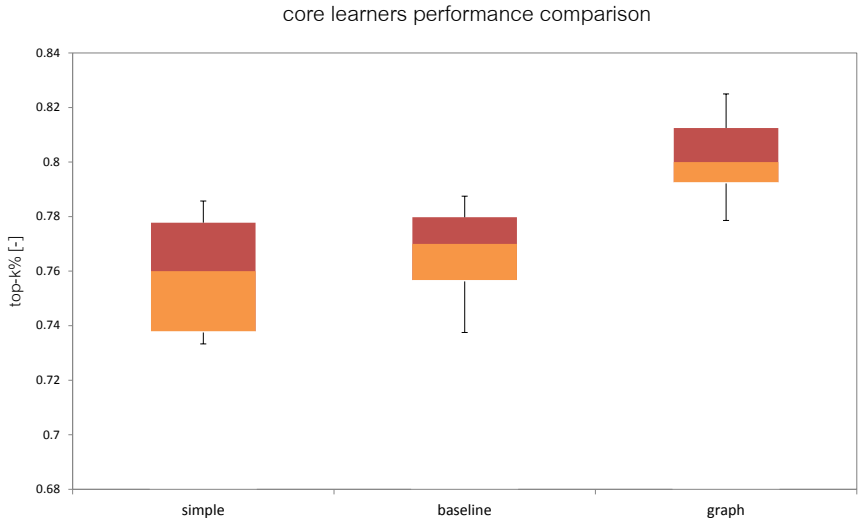


Fig. 3. Final comparison of core learners on the top-k% measure.

5 Experimental Evaluation

For the testing of our graph-based approach we needed to download a suitable dataset of tweets from Twitter. We implemented a simple crawler using the Twitter API. The sampling strategy, i.e. the choice of the set of users that should be downloaded was driven by a simple heuristic. Starting from a random seed, the heuristic orders the nodes (corresponding to users) whose neighbours should be added to the database in a greedy way so that the crawled sub-network would be as compact as possible – it picks preferentially those nodes which share most edges with nodes already stored in the database. The network subsets that were eventually crawled consist of approx. 8 thousand users with 3 million internal connections, and millions of public tweet (hashtag containing) records from December 2012, March and April 2013.

We measured the performance of our novel method and the simple and baseline methods on the largest dataset from March 2013, with windows consisting of 4 days of training and 1 day for prediction. We set k to 20 and performed multiple runs of the classifier with varying seed. We tested our features with two classifiers, namely SVM and Random Forest, both giving similar results, yet the later proved more suitable for tuning and time complexity reasons. The choice of parameters was tuned as to avoid overfitting of the classifier, i.e. by extending the training timescope to at least 4 days, and to have a clear threshold to cut between trending and non-trending hashtags, i.e. higher k such as 20 helped to avoid the situation of constant classification confidence for all the trending hashtags.

The improvement on the top-k% metric achieved by our method, as displayed in Figure 3, might in reality correspond to early detection of a couple more upcoming trends that wouldn't be otherwise discovered without taking the structure of the network into account.

We also performed other experiments with the novel method which we do not report in detail here due to lack of space but which can be found in [5]. For example we evaluated the approach on different metrics, trend definitions and parameters. We examined the influence of various timescope settings, e.g. the size of train and test parts of the sliding window and various time-fold granularities. We tested the resilience of the approaches to change of data content and network structure by an interchange of training sets from multiple datasets, proving reasonable sensitivity of the graphlet approach both to the change of the content (negligible) and to the change of the structure (slightly bigger sensitivity). We also tested whether we could not obtain better results with other relations, but the results came in the favour of the original *follows* relation over the *retweets* and *replies*. We also assessed the usefulness of time features. It turned out that time features contribute to the performance in the order of several percent. Nevertheless, time features on their own performed worse than graphlets.

6 Conclusions

In this paper, we presented an approach for prediction of trends spread within a local Twitter subnetwork, utilizing topology structure information, based on representation, inspired by methods from the area of biological networks. The results prove the value of knowledge on the network structure and the contribution of the approach itself. One of the appealing properties of the method is that it exploits information about structure of the network but at the same time it does not need the entire structure nor it does need to construct models of individual users.

Acknowledgements

This work was supported by the Czech Science Foundation grant no. P202/12/2032 and the Czech Technical University internal grant SGS11/155/OHK3/3T/13.

References

1. NA, C., JH, F.: Social network sensors for early detection of contagious outbreaks (2010) PLoS ONE 5(9): e12948.
2. Tijana Milenkovic, N.P.: Uncovering biological network function via graphlet degree signatures (2006) Oxford university press.
3. Breiman, L.: Random forests (2001) Statistics Department, University of California, Berkeley, CA 94720.
4. Dietterich, T.G.: Machine learning for sequential data: A review (2010)
5. Šourek, G.: Twitters local trends spread analysis (2013) `http://cyber.felk.cvut.cz/research/theses/detail.phtml?id=339`.