ECML PKDD 2011

EUROPEAN CONFERENCE ON MACHINE LEARNING
AND PRINCIPLES AND PRACTICE
OF KNOWLEDGE DISCOVERY IN DATABASES

5-9 SEPTEMBER 2011
ATHENS-GREECE
WWW.ECMLPKDD2011.ORG

Planning to Learn and Service-Oriented Knowledge Discovery

WORKSHOP NOTES

Editors:

Jörg-Uwe Kietz

Simon Fischer

Nada Lavrač

Vid Podpečan

ECML PKDD 2011

EUROPEAN CONFERENCE ON MACHINE LEARNING
AND
PRINCIPLES AND PRACTICE OF KNOWLEDGE DISCOVERY IN DATABASES

Planning to Learn and Service-Oriented Knowledge Discovery

PlanSoKD'11

September 9, 2011

Athens, Greece

lico

Editors:
Jörg-Uwe Kietz
University of Zurich, Switzerland
Simon Fischer
Rapid-I, Germany
Nada Lavrač
Jožef Stefan Institute, Ljubljana, Slovenia
Vid Podpečan
Jožef Stefan Institute, Ljubljana, Slovenia

Preface

As the name suggests, service-oriented computing utilizes services as the basic constructs to enable composition of applications from software and other resources distributed across heterogeneous computing environments and communication networks. The service-oriented paradigm has induced a radical shift in our definition of third-generation data mining. The 1990's vision of a data mining tool suite encapsulated in a domain-specific shell gives way to a service-oriented ar- chitecture with functionality for identifying, accessing and orchestrating local and remote data/information resources and mining tools into a task-specific workflow.

Thus the major challenge facing third-generation DM systems is the integration of these distributed and heterogeneous resources and software into a coherent and effective knowledge discovery process. Semantic Web research provides the key technologies needed to ensure interoperability of these services; for instance, the availability of widely accepted task and domain ontologies ensures common semantics for the annotation, search and retrieval of the relevant data/knowledge/software resources, thus enabling the construction of shareable and reusable knowledge discovery workflows. Another important feature is advanced support to the user. Composing effective knowledge discovery processes to solve a given application problem out of the available services is still more an art than a well-understood science. Formal planning can help a user to build such processes, but an important requirement for that is the acquisition of much more control-knowledge of what should or should not be composed together. Meta-Learning has so far been mostly applied to choose single modeling tools. Learning which services should be composed together has just been started.

The Planning to Learn and Service-Oriented Knowledge Discovery Workshop (PlanSoKD 2011) was held in Athens, Greece, on September 9^{th} 2011 in conjunction with the The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2011). The PlanSoKD workshop resulted from the merge of the Planning to Learn Workshop series (at ECML/PKDD-2007, ICML/COLT/UAI 2008 and ECAI 2010) and the Service-Oriented Knowledge Discovery workshop series (ECML/PKDD-2008, 2009 and 2010). More information about this workshop and its predecessors can be found at
`http://www.ifi.uzh.ch/ddis/events/plansokd2011/`.

Acknowledgements

We would like to thank the invited speaker, all the authors who submitted papers and all the workshop participants. We are also grateful to members of the program committee members and external referees for their thorough work in reviewing submitted contributions with expertise and patience. A special thank is due to both the ECML PKDD Workshop Chairs and the members of ECML PKDD Organizing Committee who made this event possible.

The organization of this workshop is partially supported by the European Community 7th framework program ICT-2007.4.4 under grant number 231519 e-Lico: An e-Laboratory for Interdisciplinary Collaborative Research in Data Mining and Data-Intensive Science.

Zurich, August 2011

Jörg-Uwe Kietz
Simon Fischer
Nada Lavrač
Vid Podpečan

Workshop Organization

Workshop Chairs

Jörg-Uwe Kietz	University of Zurich, Switzerland
Simon Fischer	Rapid-I, Germany
Nada Lavrač	Jožef Stefan Institute, Ljubljana, Slovenia
Vid Podpečan	Jožef Stefan Institute, Ljubljana, Slovenia

Program Committee

Abraham Bernstein	University of Zurich, Switzerland
Alexandros Kalousis	University of Geneva, Switzerland
Carlos Soares	LIAAD & University of Porto, Portugal
Christophe Giraud-Carrier	Brigham Young University, USA
Hendrik Blockeel	Leuven University, Belgium
Joaquin Vanschoren	Leiden University, Netherlands
Jörg-Uwe Kietz	University of Zurich, Switzerland
Katharina Morik	University of Dortmund, Germany
Michael Berthold	Konstanz University, Germany
Nada Lavrač	Jožef Stefan Institute, Ljubljana, Slovenia
Pavel Brazdil	University of Porto, Portugal
Sašo Džeroski	Jožef Stefan Institute - Slovenia
Simon Fisher	Rapid-I GmbH, Germany
Stefan Rüping	FhG-IAIS, Germany
Filip Železný	Czech Technical University, Czech Republic

Additional Reviewers

Panče Panov	Jožef Stefan Institute, Slovenia
Floarea Serban	University of Zurich, Switzerland

Table of Contents

A meta-mining infrastructure to support KD workflow optimization

Phong Nguyen, Alexandros Kalousis, and Melanie Hilario

Artificial Intelligence Laboratory, University of Geneva

Abstract. Knowledge Discovery in Databases (KDD) is a complex process that involves many different data processing and learning operators. Today's Knowledge Discovery Support Systems (KDSS) contain several hundreds of those operators. A major challenge of third generation KDSS is to assist the user in his/her choice of different operators in order to build workflows that are not only valid but also – ideally – optimize some performance measure associated with the user goal. The ideal KDSS should be able to select those workflows that are most likely to optimize the performance measure associated with the given user goal. In this paper we present such a system. Our system is built on top of a workflow planner that can compose valid and applicable workflows given some input data and a goal description. However the planner has no way of comparing the relative merits of the different valid operators choices – a fact which leads to an explosion of the workflow search space. We present a meta-mining infrastructure which analyses previous mining experiments, i.e. applications of different workflows on different datasets, in order to extract a model that will be used by the planner during workflow construction to guide operator selection.

1 Introduction

Knowledge Discovery in Databases (KDD) is a complex process that typically involves many different data processing and learning operators (i.e. algorithm implementations). Today's Knowledge Discovery Support Systems (KDSS) contain several hundreds of those operators. For instance, the RapidMiner[1] platform, in its extended version with Weka[2] and R[3], proposes more than 500 operators. Whenever the analyst is faced with a new knowledge discovery problem he/she has to select among the available operators the ones that can be meaningfully combined in order to build a valid knowledge discovery workflow which can address his/her goal. With the advance of third generation KDSS, one of the main challenges for these systems is to *intelligently* assist the user in the design of data mining workflows. The e-LICO project[4] virtual data mining laboratory features

[1] http://www.rapid-i.com
[2] http://www.cs.waikato.ac.nz/ml/weka/
[3] http://cran.r-project.org/
[4] http://www.e-lico.eu

an Intelligent Discovery Assistant (IDA) that supports users in the construction of mining workflows which match their data analytical goals and the input data. The IDA is built upon a cooperative AI-planner [6] which constructs data mining plans following the hierarchical task networks (HTN) planning approach. Initially, the AI-planner can only identify operators whose preconditions are met at a given planning step, but is unable to determine which of these will probably attain better performance than the others. Consequently the AI-planner can produce an extremely high number of candidate plans and leave the user at a loss as to which is most appropriate for his problem. One way to address this problem is to guide operator selection[5] using some cost function related to the goal addressed by the user. For example, if the current goal is the construction of a classification model, an appropriate cost function is the expected predictive accuracy of the model that will be produced by the final workflow. At each step, the planner will then select those operators that are expected to result in models of high predictive power.

The selection of the appropriate operator (or algorithm) and parameter settings for a given inductive task has been a predominant focus of meta-learning research for the past few decades. Meta-learning is the application of machine learning techniques to improve the performance or efficiency of base-level learners. The e-LICO approach can be more aptly called meta-mining, roughly defined as process-oriented meta-learning; more precisely, it extends meta-learning to the full knowledge discovery process [4]. In the same way that meta-learning is aimed at optimizing the results of learning, meta-mining optimizes the results of data mining processes by taking into account the interdependencies and interactions between the different process operations, and in particular between learning and the different pre/post-processing steps. In this paper, we present the e-LICO architecture with the meta-mining infrastructure for the IDA. The meta-mined model ranks data mining operators according to their potential to maximize the cost function for the given input data. Meta-mining relies on a collection of past mining experiments, applications of data mining workflows on different datasets, and the produced results, which it then analyses to produce a meta-mined model that associates good workflow performance with specific dataset characteristics and workflow patterns. Thus the e-LICO IDA combines planning with meta-mining in order to provide intelligent user support in the design of data mining workflows.

In section 2, we present the differents components of the e-LICO architecture, what they do and how they are related, and in section 3, we describe in more detail the meta-mining infrastructure, its probabilistic transition model and meta-mining analysis. We finally conclude in section 4.

[5] We use the words ”operator selection” and ”operator ranking” synonymously since the first can be achieved through the second.

2 The e-LICO architecture

Figure 1 gives an overview of the e-LICO architecture. The three (blue) shaded boxes depict the system's main components: the user interface, the Data Mining Experiment Repository (DMER), and the IDA. The user interacts with the e-LICO system by selecting from two front-ends – one provided by Taverna[6], its e-science infrastructure, and the other by RapidMiner, its main DM software package. The DMER, built on the RapidAnalytics platform, stores all the resources used and produced by the system during a user session. Finally, the IDA provides the user with data mining support through the collaboration of the AI-planner (supported by DMWF, the Data Mining Worklow Ontology [6]) and the meta-miner (supported by DMOP, the Data Mining Optimization Ontology [3]).

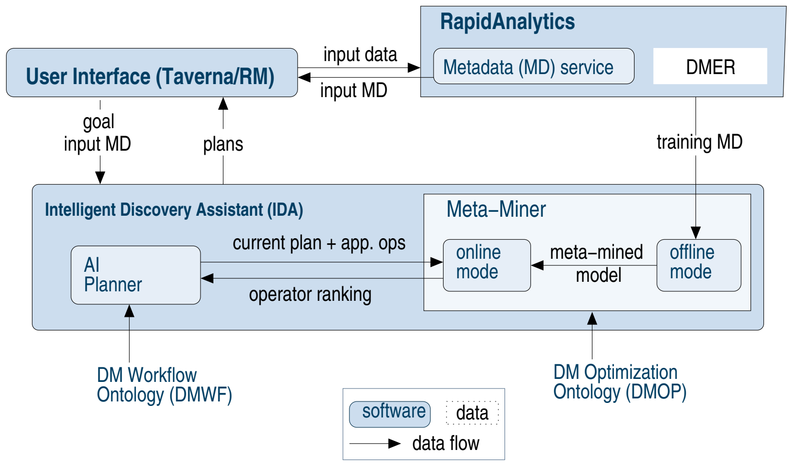


Fig. 1: The e-LICO infrastructure and its components.

The user initiates a DM experiment by selecting a data mining goal (e.g. classification) and providing an application dataset, from which RapidAnalytics Meta-data Service extracts input metadata. The MD service has been extended with the e-LICO Data Characterization Tool (DCT), which computes several types of data characteristics: 1. statistical measures (e.g. number of instances, proportion of missing values); 2. information-theoretic measures (e.g. class entropy, mutual information)[7]; 3. geometrical and topological measures (e.g. nonlinearity, volume of overlap region) [5]; and 4. model-based measures (error rates obtained by landmarkers such as 1NN or DecisionStump [7], weights learned

[6] http://www.taverna.org.uk/
[7] http://www.metal-kdd.org/

by Relief or SVM). The AI-planner uses simple statistical measures to identify applicable operators for a given task, whereas the meta-miner exploits the more advanced measures to correlate workflow performance with dataset and workflow characteristics.

As mentioned in Section 1, the initially naïve planner generates a large number of candidate workflows, from which one is selected either at random or based on simple usage frequencies. All experimental metadata — descriptions of input and intermediate data; selected operators, algorithms and workflows; learned models and performance results — are recorded and stored in the DM experiment repository. These are then structured and organized in a Data Mining Experiment Database (DMEX-DB). After a sufficiently large number of experiments, the meta-miner analyses DMEX-DB meta-data to build a model that will be used for operator selection during the planning process. The meta-mining process takes place offline; the mined model is then deployed in subsequent experiments, where it is matched with the current user goal and input metadata to build a task-specific probabilistic transition model. At each planning step, the transition model ranks candidate workflows based on its estimate of a pre-selected cost function for each applicable operator, given the current partial workflow. The next section describes in detail one among many possible meta-mining approaches to workflow assessment and ranking.

3 The Meta-Miner

The AI-planner builds mining workflows following an HTN decomposition of the CRISP-DM process model [2]. Although useful for understanding the mining processes specifications, this model is not addressed for the task of operator selection. In recent work [3, 4], a Data Mining Optimization Ontology (DMOP) has been proposed which extends the Rice framework and pries open the black box of algorithms to address the meta-mining problem. This conceptual framework has been used for characterizing the interdependencies and interactions between the different process operations of a data mining experiment by extracting frequent patterns of annotated workflows following an apriori-like algorithm [1, 4]. Figure 2 shows two examples of workflow patterns: on the top, we have a very general pattern expressing that a feature selection algorithm composed of a feature weighting algorithm and a feature weights cut-off is followed by a classification algorithm. On the bottom, we have a more specific pattern expressing that a multivariate feature selection algorithm can be followed by a decision tree algorithm. Thus, these workflow patterns assess similarities between workflows and allow to addresse the task of operator selection by reflecting knowledge extracted from past experiments.

In order to understand the interaction between the meta-miner and the AI-planner, we present first a simple probabilistic transition model for operator selection based on those frequent workflow patterns. Then we present a more elaborate model which conditions these patterns on the input dataset characteristics and the specific cost function that the user would like to optimize, e.g.

accuracy in the case of classification, in order to improve the operator selection task in a manner that will result to workflows that will potentially optimize the given cost function for the specific input data.

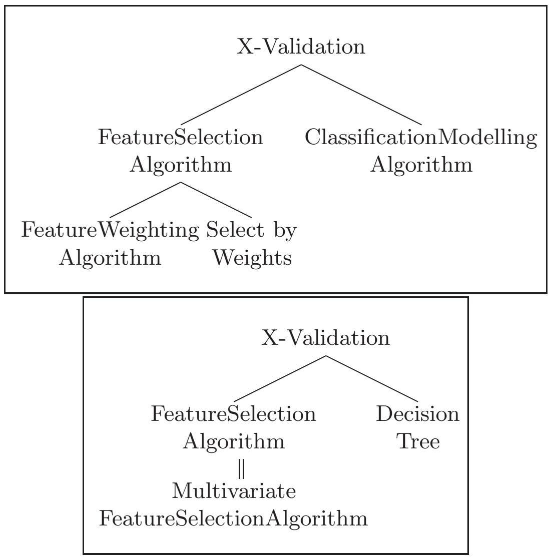

X-Validation
FeatureSelection Algorithm
ClassificationModelling Algorithm
FeatureWeighting Algorithm
Select by Weights
X-Validation
FeatureSelection Algorithm
Decision Tree
Multivariate FeatureSelectionAlgorithm

Fig. 2: Two workflow patterns
Thin edges depict workflow decomposition; double lines depict subsumption.

3.1 Basic Operator Selection

As we already mentioned in the basic operator selection scenario we follow a simple frequent pattern approach. The basic idea here is to extract frequent patterns from previously executed worflows. These patterns capture how different operators have been used together in the past to achieve a given task, however they provide no means to determine whether the specific operator combination given by them is the one that best fits the given task and the input data. In a sense the operator selection based solely on these workflow patterns is done on the basis of how often the different operators have been used in the past together. The description of the frequent pattern extraction from the data mining workflows has already been given in [4].

During the planning process the meta-miner and the AI-planner interact each time the planner has to make a choice on which is the operator that should be applied in the next step. Concretely at step i the meta-miner is given a set O of applicable operators that have been determined by the planner, based on syntactic constraints and how they contribute towards the goal, and the plan, $o_{1:i-1}$, that has been assembled so far. The meta-miner has to provide a probability for each operator $o \in O$ of being the one that should be chosen in the i_{th} step based on the workflow patterns. The planner will select that operator $\hat{o}$ that maximizes this probability, i.e.:

$$\hat{o} = \arg\max_{o \in O} p(o_i = o | o_{1:i-1}) \tag{1}$$

A frequent pattern mining approach based on ground operators is not able to discover relations between higher-level families of operators, e.g. relations such as the one given in figure 2. The approach that we took in extracting the frequent workflow patterns uses the DMOP taxonomy and is able to discover such abstract patterns which are used then to suggest what the next operator should be, as we will describe soon. This can deal with scenarios in which for example it would be better to apply an operator from some given family, say a J4.8 decision tree, but for some reason, e.g. a syntactical constraint, this operator is not applicable thus is not included in O, however there is another operator of the same family, e.g. CART, which is not used very frequently but can perform exactly the same task. Using this type of abstract workflow patterns we are able to suggest an operator from the appropriate family, even if this operator has not been used often in the past in the context of the plan that has been constructed so far.

We will now describe how to use the abstract workflows to support operator selection. Given the set of workflow patterns W extracted from the set of workflows WF stored in the DMEX-DB, we can rewrite equation 1 so that it considers the current plan and its possible solutions at the abstract level of workflow patterns as:

$$\hat{o} = \arg\max_{o \in O} (\arg\max_{w_i^o \in W} p(w_i^o | w_{i-1})) \tag{2}$$

where $w_{i-1} \in W$ is a workflow pattern that matches the current plan, and $w_i^o \in W$ is some specialization of w_{i-1}, denoted by $w_i^o \prec w_{i-1}$, with the constraint that w_i^o matches the operators $o \in O$. One can notice that all possible patterns w_i^o have the same common pattern prefix w_{i-1}. Solving equation 2 is equivalent to determining the association rule $w_{i-1} \Rightarrow w_i^o$ with the highest confidence as this is given by:

$$p(w_i^o | w_{i-1}) = \frac{s(w_i^o)}{s(w_{i-1})} \tag{3}$$

where $s(w)$ is the support function of a pattern w counting how many times a pattern appears within the different workflows:

$$s(w) = \frac{1}{|WF|} \sum_{wf \in WF} I_w(wf) \tag{4}$$

where $I_w(wf)$ returns 1 if the pattern w matches the workflow wf and zero otherwise. There can be many different w_{i-1} workflow patterns that match the plan that has been constructed so far. To select among them we rely on the trade-off of the support and confidence of the association rule $w_{i-1} \Rightarrow w_i^o$ corresponding to each one of them according to:

$$\hat{o} = \arg\max_{o \in O}(\arg\max_{w_i^o \in W}((1-\lambda)\frac{s(w_i^o)}{s(w_{i-1})} + \lambda s(w_{i-1}))) \tag{5}$$

where $\lambda \in [0,1]$ controls the trade-off between the support and the confidence. As λ approaches one we favor high support over confidence.

As already mentioned this way of selecting operators does not consider the characteristics of the input dataset nor the cost function that the user would like to optimize with the workflow under construction. In the next sections, we present a more elaborated approach to operator selection that takes into account both of them.

3.2 Conditional Operator Selection

We want to condition operator selection on the user goal g, the input metadata x extracted by the DCT as described in the section 2, and the cost function c that the user wants to optimize. More precisely given some dataset d and its metadata x, we would like to favor the use of workflow patterns that have been found to relate to good workflow performance for the goal g under the cost function c when they have been applied in the past to datasets that are similar to d, $N(d)$. In other words prefer workflow patterns that are often found in workflows that have achieved good performance when applied on $N(d)$. More precisely let:

$$\begin{aligned} N(d) \quad &= \{\ d_i|\text{ dataset } d_i \text{ is is similar to } d\} \\ WF'(x,g,c) &= \{wf|wf \in WF \text{ and } wf \text{ performs well in } N(d) \text{ for } g \text{ under } c\} \\ s(w|x,g,c) \ &= \frac{1}{|WF'(x,g,c)|}\sum_{wf \in WF'(x,g)} I_w(wf) \end{aligned} \tag{6}$$

We will see in the next section how good workflows can be defined. Note that now we define support in a different manner by considering only the good workflows, equation 6. In order to have a confidence measure that reflects the quality of the good workflow patterns, we use:

$$\text{conf}(w|x,g,c) = \frac{s(w|x,g,c)}{s(w)} \tag{7}$$

We can now use equation 7 to identify what is the operator that has the most chances achieving good performance according to the c cost function when given the d dataset described by the x metadata by:

$$p(w_i^o|w_{i-1},x,g,c) = \frac{\text{conf}(w_i^o|x,g,c)}{\text{conf}(w_{i-1,c}|x,g)} \tag{8}$$

Similar to equation 5, we will strive for a trade-off between the support of the prefix pattern and the confidence of the association rule as:

$$\hat{o} = \arg\max_{o \in O}(\arg\max_{w_i^o \in W}(\frac{\text{conf}(w_i^o|x,g)}{\text{conf}(w_{i-1}|x,g)} + \lambda\text{conf}(w_{i-1}|x,g))) \tag{9}$$

With equation 9, the meta-miner is now able to provide probability transitions to the AI-planner such that at each specialization step of the current plan the selected operator is the one that has most chances to optimize c given x.

In the next section, we present the offline meta-mining analysis which completes this section. The meta-mining analysis builds a meta-mined model that allows to compute the conditioned support.

3.3 The Meta-Mining Analysis

As mentioned in section 2, the meta-mining analysis is made offline, before the planning process starts. The goal of this component is to analyse past experiments in order to provide operator suggestions to the AI-planner through a meta-mined model. One of the key components of the meta-mining analysis is to determine the set of datasets $N(d)$ that are in some sense similar to the input dataset d. We will briefly describe how we can do so when the goal is to construct classification workflows that maximize the predictive accuracy. In order to do so we analyse a collection of datasets by applying on them a number of different classification algorithms and estimate their predictive performance. Each dataset is now described by a vector of accuracies of the classification algorithms that we have use: one-nearest-neighbor (1NN), decision tree algorithms J48 and CART, Naive Bayes (NB), logistic regression algorithm (LR), and SVMs with linear (SVM-L) and Gaussian (SVM-R) kernels. For J48 the C (pruning confidence) and M (minimum number of instances per leaf) parameters were set to 0.25 and 2 respectively; for CART the M and N (number of folds for the minimal cost-complexity pruning) parameters were set to 2 and 5 respectively. The C parameter was set to 1 for both SVM-L and SVM-R, and SVM-R's γ parameter was set to 0.1. We used the implementations of these algorithms in the RapidMiner data mining suite. These algorithms represent quite distinct learning biases. We measure the similarity of datasets using Spearman's rank correlation coefficient on the performance ranking of the different algorithms applied to them. The main idea here is that datasets will be similar for what we want to do, i.e. workflow construction for classification, if the performance order of different algorithms is similar. Then we cluster the space of datasets using agglomerative clustering with Ward's method in order to determine clusters of datasets on which we have similar profiles of algorithm performance. These clusters are used to define the neighborhood $N(d)$ of a new dataset. Since it is clear that we cannot apply the classification algorithms every time we have a new dataset, since these is computationally expensive, we need a different way to locate the cluster to which d belongs. We do so by defining a classification problem in which datasets are described in terms of the metadata description and the class is the cluster to which the dataset belongs. Additionally for each cluster we separate the workflows to "good" and "bad" workflows with respect to their relative performance on the datasets that belong in the cluster. We then score the different workflow patterns according to their conditional support as this is given in equation 6. So whenever we need to plan classification workflow for some new dataset we first need to situate the new dataset in the appropriate cluster of datasets using its metadata description and then use workflow patterns to suggest what the next operator should be according to the estimated quality of these workflow patterns in the dataset cluster.

4 Conclusion and Future Work

We have very briefly sketched the main points of an infrastructure that allows us to plan data mining workflows by exploiting patterns of good workflow performance in past experiments. The infrastructure combines traditional planning and meta-mining in order to associate workflow patterns that are expected to lead to good performance for a selected user goal under some cost function. We are currently fine-tuning the different parts of the infrastructure. Especially challenging is the construction of the appropriate classification model that will allow us to situate a new dataset in the correct cluster with respect to the expected relative performance of the different classification algorithms on it. Additionally we want to determine the performance improvement that meta-mining brings into the process of workflow planning. Finally we would like to experiment with different ways of combining workflow patterns and dataset characteristics that build on work from the area of recommender systems.

Acknowledgments This work is partially supported by the European Community 7^{th} framework program ICT-2007.4.4 under grant number 231519 "e- Lico: An e-Laboratory for Interdisciplinary Collaborative Research in Data Mining and Data-Intensive Science". The meta-mining infrastructure described in this paper is the result of ongoing collaborative work within the e-LICO project of Jörg-Uwe Kietz, Floarea Serban, Simon Fischer, Agnieszka Ławrynowicz and the authors. For ease of presentation we made some simplications with respect to the full meta-mining infrastructure.

References

1. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast discovery of association rules. In: Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.) Advances in knowledge discovery and data mining, pp. 307–328. American Association for Artificial Intelligence, Menlo Park, CA, USA (1996)
2. Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., Wirth, R.: Crisp-dm 1.0 step-by-step data mining guide. Tech. rep., The CRISP-DM consortium (August 2000), http://www.crisp-dm.org/CRISPWP-0800.pdf

3. Hilario, M., Kalousis, A., Nguyen, P., Woznica, A.: A data mining ontology for algorithm selection and meta-learning. In: Proc of the ECML/PKDD09 Workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery (2009)
4. Hilario, M., Nguyen, P., Do, H., Woznica, A., Kalousis, A.: Ontology-based meta-mining of knowledge discovery workflows. In: Jankowski, N., Duch, W., Grabczewski, K. (eds.) Meta-Learning in Computational Intelligence. Springer (2011)
5. Ho, T.K., Basu, M.: Data complexity in pattern recognition. Springer (2006)
6. Kietz, J.U., Serban, F., Bernstein, A., Fischer, S.: Towards Cooperative Planning of Data Mining Workflows. In: Proc of the ECML/PKDD09 Workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery (SoKD-09) (2009)
7. Pfahringer, B., Bensusan, H., Giraud-Carrier., C.: Meta-learning by landmarking various learning algorithms. Proc. 17th International Conference on Machine Learning pp. 743–750 (2000)

RMonto - towards KDD workflows for ontology-based data mining

Jedrzej Potoniec, Agnieszka Ławrynowicz
Institute of Computing Science,
Poznań University of Technology, Poznań, Poland,
email: {jpotoniec,alawrynowicz}@cs.put.poznan.pl

Abstract. We present a tool, implemented as an extension to the popular data mining framework RapidMiner, that supports modeling and execution of KDD workflows involving data mining methods that exploit ontologies as background knowledge. In order to support the state-of-the art in ontology related technologies, in particular Semantic Web technologies, our tool offers acquisition of data from local files as well as from SPARQL endpoints. The latter feature enables easy consumption of various distributed Semantic Web data such as Linked Open Data as input for data mining experiments. The tool is presented on the example of currently implemented clustering algorithms together with a set of operators for measuring similarity by exploitation of ontologies.

1 Introduction

The "first generation data mining systems" provided a small set of algorithms operating on attribute-valued data. Currently available "second generation systems" address scalability, functionality and flexibility issues, by providing among others an access to data warehouses, and supporting data mining schemas and query languages. It is claimed that a major challenge for emerging "third generation data mining systems" is then in the integration of distributed, and possibly heterogeneuous data and knowledge resources and tools. The systems should hence support data mining tasks that deal with distributed data, found e.g. on intranets or on the Web, and also integrate with tools for managing knowledge. An important feature of such systems is also an advanced support for the user in performing KDD tasks. To achieve its goals, third generation data mining requires modularized, and composable implementations of data mining algorithms, that would support easier construction, and execution of KDD workflows.

Following this direction of research on data mining systems we introduce a tool to support design and execution of knowledge discovery processes exploiting ontologies as background, domain knowledge, and operating on possibly distributed Semantic Web [2] data.

The tool, an extension of RapidMiner[1] [11] we named RMonto, is being designed, and developed to support data mining approaches that despite having various names, we think often have many commonalities. They may be called *Semantic Web mining* [14], in particular data mining *from* the Semantic Web, or alternatively also *ontology mining* [5] that is activities that allow to discover

[1] `http://rapid-i.com/`

hidden knowledge from ontological knowledge bases. Since ontological knowledge bases are commonly represented in description logics [1] hence we may also refer to the name *description logic learning* (or *DL-learning* in short) [9], where DL-learning is a kind of Inductive Logic Programming approach that assumes description logics as a logical language to represent data and hypotheses. A data mining approach where domain ontologies are used as background knowledge has been also recently named *semantic data mining* [12].

In the next sections, we discuss the related work, the design issues, and present the current status of the implementation of the tool.

2 Related work

With growing adoption of the Semantic Web technologies, and increasing availability of large amounts of data annotated by ontologies, ontology–based data mining approaches become an important line of research. However, there are very few tools publicly available to support such tasks.

A tool which is the most related to ours is DL-learner [10], which provides a framework for learning in description logics and OWL[2]. DL-Learner provides machine learning algorithms exploiting OWL, it supports different knowledge base formats, an OWL library, and reasoner interfaces. It uses a component-based model, with four types of components: knowledge source, reasoning service, learning problem, and learning algorithm. Current DL-Learner algorithms mostly address the concept learning task. A feature distinguishing our approach from DL-Learner is that by exploiting the framework of RapidMiner, we go further with a modularized, and compositional approach, and allow for designing a data mining workflows that ultimately may be more arbitrary, and fine-grained than what is currently available to be done with DL-Learner.

Relevant to our work is already existing RapidMiner extension: rapidminer–semweb[3] [7]. This extension provides operators for extracting an RDF graph from a repository and transforming it into a feature vector, preprocessing methods for resolving set-valued features occurring when creating feature-vectors from RDF data, and visualization of the transformation process. As such, the extension does not tackle the problem of learning directly from expresssive and semantically rich representations such as description logic knowledge bases.

A proposal for a data mining extension to SPARQL[4], a standard Semantic Web query language, named SPARQL-ML [8], may also be considered as relevant to our work. That extension introduces new keywords to the SPARQL syntax that serve for the induction of models, and their further use for prediction/classification. There have been also tools, such as g-SEGS[5], that support lightweight ontologies in the form of taxonomies in the data mining algorithm.

Despite of the availability of the abovementioned tools, we have found that none of them gathers all of the characteristcs that we find important, such as availability of algorithms supporting certain types of tasks (e.g., algorithms exploiting ontologies in a manner of DL-learning, such as clustering involving semantic similarity measures), or a possibility to model data mining workflows. RMonto was developed in order to address these shortcomings.

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

[3] `http://code.google.com/p/rapidminer-semweb/`

[4] http://www.w3.org/TR/rdf-sparql-query/

[5] `http://kt.ijs.si/anze_vavpetic/SDM/index.html`

3 A tool for ontology-based data mining

3.1 Design of a framework

While designing a tool to support data mining tasks exploiting ontologies and Semantic Web data, one needs to consider several issues such as possible distributed nature of the data, and the state-of-art in the tools supporting manipulation and reasoning with ontologies. In case of developing an extension to an existing framework such as RapidMiner, one also needs to take its characteristics into account. For example, the most typical data format handled by RapidMiner is a tabular data, while the data we deal with in our extension is relational.

In particular, we have identified the following requirements:

1. Ability to flexibly, and on-demand replace a reasoning tool. Reasoning tools differ in their capabilities, and performance, e.g. semantic stores (such as e.g., OWLim[6]) are optimized towards scalable, and fast querying and data retrieval, while OWL reasoners (such as e.g. Pellet[7] or HermiT[8]) support advanced reasoning on very expressive ontologies [3,13].
2. Support for loading data from multiple different sources, such as local files stored in different formats (e.g. RDF/XML, N3), files available on the Web, triple stores queried with SPARQL language to harvest data for data mining experiments, relational databases treated as triple-stores and queried with SQL. Subsequently, integration of the harvested data into a one knowledge base, which later can be queried with conjunctive queries, such as formulated in SPARQL to select examples.

During software development, those requirements can be addressed in the following way:

1. Plugin oriented architecture, moving dependency resolving from build time to run-time. The only thing that has to be available during software building process is a set of common abstract interfaces, used as facades to the reasoning tools and implemented by the plug-ins. In *Java*, this could be implemented for example with *Open Services Gateway initiative framework*[9] or by hand-made solution based on JAR files and their manifest files.
2. Data downloading, loading and integration can generally be addressed by adequate reasoning software parts, optionally extended with additional tools and libraries, such as JDBC for querying relational databases.

In order to support flexibility in using different semantic reasoners, and avoid dependency of our implementations on particular reasoners or APIs, we have developed our internal API, called PutOntoAPI (where “Put“ stands for Poznan University of Technology), which acts as a bridge between our implementations of data mining algorithms and reasoning/storing/etc. software that is called to invoke relevant reasoning services via APIs used by popular semantic reasoners such as Jena API, OWL API or Sesame API.

[6] `http://www.ontotext.com/owlim`
[7] `http://clarkparsia.com/pellet/`
[8] http://www.hermit-reasoner.com/
[9] `http://www.osgi.org/`

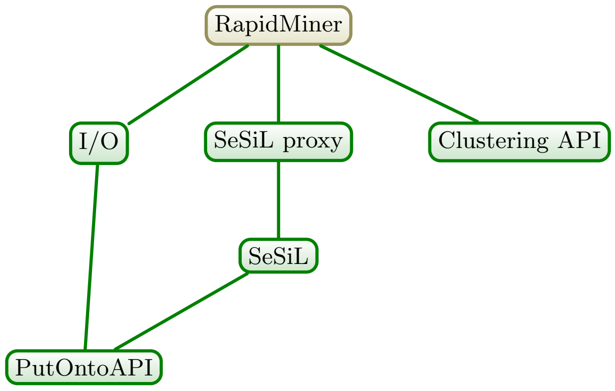


Fig. 1: Architecture of current RMonto extension.

3.2 Extending RM by New Operators

The currently implemented components of RMonto are organized with three core libraries:

PutOntoAPI which works as a facade between the higher level libraries and reasoning software with plug-in architecture;

SeSiL implementing different kernel methods for Semantic Web data;

RMonto gathering functions provided by PutOntoAPI and SeSiL into RapidMiner operators.

The overall architecture of our software is presented in Fig. 1.

During development, we created several new operators for RapidMiner. Below, they are grouped with respect to the RapidMiner operators tree sections.

Loading We have developed three operators for loading data: *Load from file*, *Load from SPARQL endpoint* and *Build knowledge base*. As one can expect, *Load from file* defines access to files accessible with normal file-system operations (i.e. stored locally or on the mounted network share). *Load from SPARQL endpoint* makes possible to download part of remote data from some SPARQL endpoint in the form of a subgraph. Those graphs are constructed with SPARQL language, namely with a CONSTRUCT query.
The reason for *Load...* operators to exist is to supply the third one with parameters of data sources. It is to avoid situation, in which the whole data loading logic is hidden in one operator, what would happen if those parameters were given as *Build knowledge base's*. Data loading is invoked inside *Build knowledge base*, as callback methods of objects received from the *Load...* operators.

ABox *SPARQL selector* and *ABox extractor* operators both provide list of URI identifying objects to be used in the learning process. Those URIs must be valid identifiers of individuals in the KB built by *Build knowledge base*. The first operator uses a SPARQL query to extract data and the second one simply extracts a list of URIs of the whole ABox.

TBox Operators *All known classes* and *Features selector* work in the similar way as those in the *ABox* section, but they build a list of classes instead. Both generate the list of classes in the internal RMonto format. We resigned

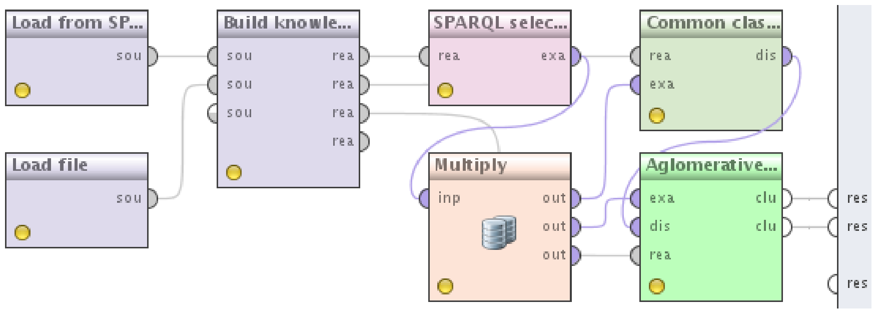


Fig. 2: A screenshot of the RapidMiner RMonto workflow.

from URI-list representation in favor of our internal format to provide more complex language, enabling usage of complex classes not defined in the KB (i.e. intersection of two defined classes). This is the way the *Features selector* operator works, providing an editor to edit list of classes by hand. *All known classes* extracts information about all named classes in the KB.

Kernels This section contains operators implementing kernel functions. They compute dissimilarity matrix, that is pairwise dissimilarities between objects. For instance, the implementation includes *Identity* and *Common classes* kernels [4] and *Epistemic kernel* [6].

Clustering Currently, there are implementations of two algorithms. The first one, *Semantic k–Medoids* is a simple *k–Medoids* algorithm implemented to be able to work with dissimilarity matrix computed by a kernel. *Agglomerative hierarchical clustering* is extended to produce semantic description of computed clusters. If the cluster has only one element, conjunction of all its related classes is adopted. When clusters are joined, their descriptions are joined with union and simplified by removing duplicates.

Fig. 2 presents a screenshot of RapidMiner with an RMonto workflow.

RMonto software as well as a short tutorial introducing a tool can be found at `http://semantic.cs.put.poznan.pl/RMonto/`

4 Conclusions and Future Work

In this paper, we present RMonto, a RapidMiner extension that supports designing and execution of KDD workflows addressing tasks of data mining exploiting ontologies as background knowledge. The tool is under development, but already supports a set of features such as data harvesting from distributed Semantic Web sources, data integration, and a set of operators suitable for similarity based data mining methods.

Ongoing and future work includes extending RMonto by new algorithm implementations, in particular the ones addressing (frequent) pattern mining.

Acknowledgements. We acknowledge the support from European Community 7th framework program ICT-2007.4.4 (grant 231519 ”e-LICO: An e-Laboratory for Interdisciplinary Collaborative Research in Data Mining and Data-Intensive Science”) and from the Polish Ministry of Science and Higher Education (grant N N516 186437).

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
2. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American 284(5), 34–43 (May 2001), http://www.scientificamerican.com/article.cfm?id=the-semantic-web
3. Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., Velkov, R.: OWLIM: A family of scalable semantic repositories. Semantic Web 2(1), 33–42 (2011)
4. Bloehdorn, S., Sure, Y.: Kernel methods for mining instance data in ontologies. In: Aberer, K., Choi, K.S., Noy, N.F., Allemang, D., Lee, K.I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudr-Mauroux, P. (eds.) The Semantic Web — Proceedings of the 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference (ISWC 2007 + ASWC 2007), November 11-15, 2007, Busan, Korea. Lecture Notes in Computer Science, vol. 4825, pp. 58–71. Springer, Berlin–Heidelberg, Germany (2007)
5. d'Amato, C., Fanizzi, N., Esposito, F.: Inductive learning for the Semantic Web: What does it buy? Semantic Web 1(1-2), 53–59 (2010)
6. Fanizzi, N., D'Amato, C., Esposito, F.: Learning with kernels in description logics. In: Proceedings of the 18th international conference on Inductive Logic Programming. pp. 210–225. ILP '08, Springer-Verlag, Berlin, Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-85928-4_18
7. Khan, M.A., Grimnes, G.A., Dengel, A.: Two pre-processing operators for improved learning from Semantic Web data http://www.cs.jyu.fi/ai/papers/IJIIT-2005.pdf
8. Kiefer, C., Bernstein, A., Locher, A.: Adding data mining support to SPARQL via statistical relational learning methods. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC. Lecture Notes in Computer Science, vol. 5021, pp. 478–492. Springer (2008)
9. Kietz, J.U., Morik, K.: A polynomial approach to the constructive induction of structural knowledge. Machine Learning 14(1), 193–217 (1994)
10. Lehmann, J.: DL-Learner: Learning concepts in description logics. Journal of Machine Learning Research 10, 2639–2642 (2009)
11. Mierswa, I., Scholz, M., Klinkenberg, R., Wurst, M., Euler, T.: Yale: Rapid prototyping for complex data mining tasks. In: In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 935–940. ACM Press (2006)
12. Novak, P.K., Vavpetič, A., Trajkovski, I., Lavrač, N.: Towards semantic data mining with g-SEGS. In: Proceedings of the 11th International Multiconference Information Society IS 2009 (2009)
13. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J. Web Sem. 5(2), 51–53 (2007)
14. Stumme, G., Hotho, A., Berendt, B.: Semantic Web Mining: State of the art and future directions. J. Web Sem. 4(2), 124–143 (2006)

Semantic Data Mining System g-SEGS

Anže Vavpetič[1], Igor Trajkovski[2],
Petra Kralj Novak[1], Nada Lavrač[1,3]

[1] Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia
[2] Faculty of Electrical Engineering and Information Technologies, Ss. Cyril and Methodius University, Skopje, Macedonia
[3] University of Nova Gorica, Nova Gorica, Slovenia
{anze.vavpetic,petra.kralj.novak,nada.lavrac}@ijs.si

Abstract. With the expanding of the Semantic Web and the availability of numerous ontologies which provide domain background knowledge and semantic descriptors to the data, the amount of semantic data is rapidly growing. The data mining community is faced with a paradigm shift: instead of mining the abundance of empirical data supported by the background knowledge, the new challenge is to mine the abundance of knowledge encoded in domain ontologies, constrained by the heuristics computed from the empirical data collection. We address this challenge by an approach, named semantic data mining, where domain ontologies define the hypothesis search space, and the data is used as means of constraining and guiding the process of hypothesis search and evaluation. The use of prototype semantic data mining system g-SEGS is demonstrated in a simple semantic data mining scenario and in two real-life functional genomics scenarios of mining biological ontologies with the support of experimental microarray data.

Keywords. semantic data mining, ontologies, background knowledge, relational data mining

1 Introduction

The knowledge discovery process can significantly benefit from the domain (background) knowledge, as successfully exploited in relational data mining and Inductive Logic Programming (ILP). Additional means of providing more information to the learner is by providing semantic descriptors to the data.

Usually, there is abundant empirical data, while the background knowledge is scarce. However, with the expanding of the Semantic Web and the availability of numerous ontologies which provide domain background knowledge and semantic descriptors to the data, the amount of *semantic data* (e.g., ontologies and annotated data collections) is rapidly growing[1]. The data mining community is now faced with a paradigm shift: instead of mining the abundance of empirical data

[1] See the Linked Data site http://linkeddata.org/

supported by the background knowledge, the new challenge is to mine the abundance of knowledge encoded in domain ontologies, constrained by the heuristics computed from the empirical data collection. This paper uses the term *semantic data mining* to denote this new data mining challenge and approaches in which semantic data are mined.

We present g-SEGS, a prototype semantic data mining system implemented in the novel service-oriented data mining environment Orange4WS [16] which supports knowledge discovery workflow construction from distributed data mining services. System g-SEGS is a successor of SEGS, a system for Searching of Enriched Gene Sets [20] designed specifically for functional genomics tasks. While SEGS is a special purpose system for analyzing microarray data with biological ontologies as background knowledge, g-SEGS is a general purpose semantic data mining system. It takes as input (1) domain ontologies in the OWL format, used to construct a version space of hypotheses (patterns) to be mined, and (2) an empirical data collection, annotated by domain ontology terms, used to constrain and guide the top-down search of hierarchically structured space of hypotheses, as well as for hypotheses quality evaluation.

After presenting the related work in Section 2, the paper presents the g-SEGS system, its implementation in Orange4WS, and its applications. Sections 3 first introduces the semantic data mining task and presents the proposed semantic data mining methodology, together with the g-SEGS algorithm implementation. Section 4 presents an illustrative example of using g-SEGS in a simple handcrafted semantic data mining scenario, followed by the presentation of selected results of using g-SEGS in real-life functional genomics use cases in Section 5. In Section 6 we conclude and propose directions for further work.

2 Related work

The idea of using hierarchies as background knowledge to generalize terms in inductive rule learning has been proposed already in [13]. More recent usage of ontologies in data mining includes [7, 2, 19, 3, 12] as well as domain specific systems which use ontologies as background knowledge for data mining [9, 20].

In [7], the use of taxonomies (where the leaves of the taxonomy correspond to attributes of the input data) on paleontological data is studied. The problem addressed was to predict the age of a fossil site on the basis of the taxa that have been found in it – the challenge was to consider taxa at a suitable level of aggregation. Motivated by this application, the authors studied the problem of selecting an antichain from a taxonomy that improves the prediction accuracy. In [2], background knowledge is in the standard inheritance network notation and the KBRL[2] algorithm performs a general-to-specific heuristic search for a set of conjunctive rules that satisfy user-defined rule evaluation criteria. In [19], ontology-enhanced association mining is discussed and four stages of the (4ft-Miner-based) KDD process are identified that are likely to benefit from

[2] KBRL is based on the RL learning program of [5]

ontology application: data understanding, task design, result interpretation and result dissemination over the semantic web. The work of [3] first focuses on preprocessing steps of business and data understanding in order to build an ontology driven information system (ODIS), and then the knowledge base is used for the post-processing step of model interpretation. Liu et al. [12] propose a learning-based semantic search algorithm to suggest appropriate Semantic Web terms and ontologies for the given data.

An ontology driven approach to knowledge discovery in biomedicine is described in [9], where efforts to bridge knowledge discovery in biomedicine and ontology learning for successful data mining in large databases are presented. A domain specific system that uses ontologies and other hierarchies as background knowledge for data mining is SEGS [20]. The SEGS system finds groups of differentially expressed genes, called *enriched gene sets*[3]. Compared to earlier work [18, 10], the novelty of SEGS is that it does not only test existing gene sets (existing ontology terms) for differential expression but it generates also new gene set descriptions that represent novel biological hypotheses.

The main differences of system g-SEGS compared to the related approaches is that these either (1) use non-standard ontology formats [7, 20], (2) are domain specific [7, 20], (3) are not implemented as web services [2, 7], or (4) perform non-symbolic classification [7].

3 Semantic Data Mining with g-SEGS

In this paper we use the term *semantic data mining* to denote a data mining task in which semantic data are mined. This section first introduces this task, followed by the methodology of semantic data mining as implemented in g-SEGS.

3.1 Semantic data mining

A *semantic data mining task*, illustrated in Figure 1, is defined as follows.

Given: a set of domain ontologies and an empirical data collection, annotated by domain ontology terms,

Find: a hypothesis (a predictive model or a set of descriptive patterns) by mining the abundance of information in ontologies, constrained by the information in the empirical data collection.

Successful approaches to solving the semantic data mining task may result in a paradigm shift in which the abundance of domain ontologies will be mined, and the empirical training data will be used mainly to constrain the hypothesis search space by the heuristics computed from the training data collection[4].

[3] A gene set is enriched if the genes that are members of this gene set are statistically significantly differentially expressed compared to the rest of the genes.

[4] A similar challenge is faced in pattern mining research where the original problem of mining the abundance of data was recently transformed into a problem of mining the abundance of induced patterns, constrained by the heuristics computed from the training data.

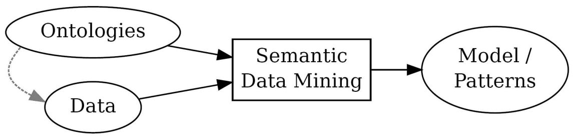


Fig. 1. Schema of a semantic data mining process, with ontologies and annotated data as inputs.

Motivated by the successful applications of SEGS [20, 14], we have decided to generalize SEGS to become domain independent, and developed a new system named g-SEGS (generalized SEGS).The methodology, implemented in the g-SEGS system, assumes that the hypothesis language are logical rules, where rule conditions are conjunctions of ontology terms. While statistical significance of rules could be measured on the fly in the process of rule construction, we have decided to construct all the rules satisfying the support constraint, and to eliminate insignificant rules in rule postprocessing, using a heuristic known from subgroup discovery. As shown in Section 4, semantic data mining results in more general and semantically more meaningful rules, if compared to standard rule learning. From the four main components of SEGS, only the SEGS hypothesis language and the generation and pruning procedure are used unchanged in the new semantic data mining system g-SEGS.

The proposed *semantic data mining methodology*, implemented in g-SEGS, is described in terms of its four main components: the hypothesis language, the input (domain ontologies and training data), the hypothesis generation procedure and the hypothesis (pattern) evaluation and filtering procedure.

3.2 Hypothesis language

The hypothesis language are descriptive patterns in the form of rules $Class \leftarrow Conditions$, where $Conditions$ is a logical conjunction of ontology terms. For example, a rule whose antecedent is a conjunction of three terms, has the form $Class \leftarrow X \wedge Y \wedge Z$, where X stands for all $x \in X$, Y stands for all $y \in Y$, and Z stands for all $z \in Z$, and where e.g., $X \in Ont1, Y \in Ont2$, and $Z \in Ont3$.

3.3 Input

g-SEGS requires two types of inputs: the ontological background knowledge and the training data.

Background knowledge consists of domain ontologies, typically in the OWL format.[5] Ontologies are used to construct the hypothesis search space.

[5] In addition to OWL ontologies, we allow for other formats of annotated hierarchically structured data sources, such as the ENTREZ and KEGG hierarchies, which were used in one of the two real-life functional genomics use cases in Section 5.

Training data are class-labeled vectors of attribute values, annotated by the terms in domain ontologies. The data are used to constrain the hypothesis search, and for rule quality evaluation in rule postprocessing.

3.4 Rule construction

Rule construction results in a set of rules satisfying the minimal support criterion. As a rule antecedent is a conjunction of ontology terms, all possible conjunctions of ontology terms can be generated and evaluated for small ontologies. In case of large ontologies, however, the search space needs to be pruned. To do so, we use the subsumption property of a relation which forms the hierarchical backbone of the ontology (e.g. `is-a`). Suppose that rule $C \leftarrow X' \wedge Y' \wedge Z'$ has been constructed by the specialization of rule $C \leftarrow X \wedge Y \wedge Z$, where $X' \preceq X, Y' \preceq Y, Z' \preceq Z$ ($\preceq$ denotes *more or equally specific* relation). If rule $C \leftarrow X' \wedge Y' \wedge Z'$ covers m objects where $m < N$ (m is lower than the support threshold N which determines the minimal number of objects to be covered by each rule), it is pruned and none of its specialization will be constructed. This results in a significant reduction of the hypothesis search space.

In a simplified case, where three ontologies $Ont1$, $Ont2$ and $Ont3$ are given, hypothesis generation consists of creating the conjunctions of individual ontology terms, one from each ontology. Hypothesis construction is performed in a top-down manner, starting from the most general terms in each of the three ontologies, and specializing the rule antecedent as long as the stopping criterion is satisfied (ensuring sufficient coverage of data instances)[6]. If one conjunct does not satisfy the constraint, then its descendents will also not satisfy it, because they cover a subset of instances covered by the conjunction. Therefore, we first construct conjuncts from the top nodes of $Ont1$, $Ont2$ and $Ont3$, and if the conjunction fails to satisfy the given constraint, g-SEGS will not refine the last added term. Note that the efficiency of the algorithm comes from the usage of the hierarchical structure of ontologies.

In addition to `is-a` or `instance-of` subsumption relations there may be other links (relations) among ontology terms, e.g, the `interacts` relation. Consider a simple rule `class(A)` $\leftarrow$ `is-a(A,B)`, and suppose that ontology term `B` is linked with term `C` through `interacts(B,C)`. In this case, the rule's antecedent can be refined to form a conjunction `is-a(A,B)` $\wedge$ `interacts(B,C)`. This illustrates a situation which is common to ILP, as one can also make statements about `B` or `C`, not only about term `A` which appears in the rule head `class(A)`. Hence a simple top-down refinement approach to rule construction is insufficient, as will be shown in an example of Section 5.

[6] If the ontology is simply a hierarchy (a tree), with the root of the graph being the most general term, this means that substantial pruning of the search space can be achieved in rule construction.

3.5 Rule filtering and evaluation

As the number of generated rules can be large, uninteresting and overlapping rules have to be filtered. In g-SEGS, rule filtering is performed using the *wWRAcc* (Weighted Relative Accuray heuristic with example weigths) heuristic [11], which uses example weights to provide the means for considering different parts of the example space when selecting the best rules in rules postprocessing. In the *wWRAcc* heuristic defined below, N' denotes the sum of weights of all examples, $n'(C)$ is the sum of weigths of examples of concept C, $n'(Cnd)$ is the sum of weights of all covered examples, and $n'(Cnd \wedge C)$ is the sum of weights of all correctly covered examples of concept C.

$$wWRAcc(C \leftarrow Cnd) = \frac{n'(Cnd)}{N'} \cdot \left(\frac{n'(Cnd \wedge C)}{n'(Cnd)} - \frac{n'(C)}{N'} \right)$$

Rule filtering, using the weighted covering approach, proceeds as follows. It starts with a set of generated rules, a set of examples with weights equal to 1 and parameter k, which denotes how many times an example can be covered before being removed form the example set. In each iteration, we select the rule with the highest *wWRAcc* value, add it to the final rule set, and remove it from the set of generated rules. Then counter m is increased to $m + 1$ and weigths of examples covered by this rule decreased to $\frac{1}{m+1}$, where example weight $\frac{1}{m}$ means that the example has already been covered by $m < k$ rules. These steps are repeated until the algorithm runs out of examples or rules or if no rule has a score above 0. Once the learning process is finished and the rules have been generated and filtered, they are evaluated and sorted using the Fisher's exact test or the original *WRAcc* (Weighted Relative Accuray) measure known from CN2-SD subgroup discovery, which trades-off the generality of a rule and its precision. The *WRAcc* heuristic is defined as

$$WRAcc(C \leftarrow Cnd) = \frac{n(Cnd)}{N} \cdot \left(\frac{n(Cnd \wedge C)}{n(Cnd)} - \frac{n(C)}{N} \right)$$

where N is the number of all examples, $n(C)$ is the number of examples of concept C, $n(Cnd)$ is the number of all covered examples, and $n(Cnd \wedge C)$ is the number of all correctly covered examples of concept C.

3.6 g-SEGS implementation

The g-SEGS system takes as input the ontologies in the OWL format and data in the Orange [15] format, uses the hierarchical structure of the `is-a` relation of ontologies for efficient search and pruning of the rule search space, generates rules by forming conjunctions of terms from different ontologies, and uses the *wWRAcc* (Weighted Relative Accuray heuristic with example weigths) for rule pruning by iteratively selecting the rules and Fischer exact test or *WRAcc* (Weighted Relative Accuray) to sort/rank the selected rules.

g-SEGS is implemented in the Orange4WS [16] environment which upgrades the freely available Orange [15] data mining environment with several additional

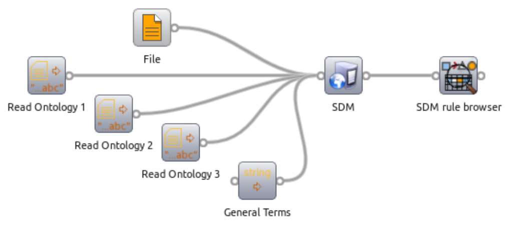


Fig. 2. An Orange4WS workflow with g-SEGS.

features: simple creation of new visual programming units (widgets) from distributed web services, composition of workflows from both local and distributed data processing/mining algorithms and data sources, and implementation of a toolkit for creating new web services. By using these tools, we were able to give g-SEGS a user-friendly interface and the ability to be executed remotely as a web service. By mapping the g-SEGS input to the SEGS input we were able to fully reuse the already implemented SEGS system. We defined the g-SEGS web service using WSDL (Web Service Definition Language). Using the web service definition and the set of tools provided by Orange4WS, we created a web service for our system. Finally, also using Orange4WS, we imported the web service into the Orange visual programming environment, thus allowing g-SEGS to be used in various workflows together with other Orange widgets.

A screenshot of an Orange4WS workflow with g-SEGS is shown in Figure 2. The workflow is composed of one widget for loading the dataset (`File`), three widgets for loading the three ontologies (`Read Ontology`), and one widget for specifying top-level ontology terms that are too general to appear in the final rules (`General terms`). These five widgets act as the input to the g-SEGS widget, which generates rules, displayed in the g-SEGS `Rule set browser` widget.

4 An illustrative example

As a proof-of-concept semantic data mining example, consider a bank which has the following data about its customers: place of living, employment, bank services used, which includes the account type, possible credits and insurance policies and so on. The bank also categorized the clients as 'big spenders' or not and wants to find patterns describing big spenders. Table 1 presents the training data.

The application of standard classification rule learning algorithm CN2 (we chose the Orange [15] implementation of CN2) to these data generates the rules presented in the top part of Table 2, and the middle part of this table presents the results obtained by using the CN2-SD subgroup discovery algorithm [11].

While CN2 generates a set of dependent and very specific classification rules, CN2-SD produces rules representing individual subgroup descriptions which are better suited for the comparison with the results obtained with g-SEGS. Note that both sets of rules are rather specific, due to the specificity of the attribute-value data representation. Standard data mining does not provide automated

Table 1. Table of bank customers described by different attributes and class 'big spender'.

id	occupation	location	account	loan	deposit	inv_fund	insur.	big_spender
1	Doctor	Milan	Classic	No	No	TechShare	Family	YES
2	Doctor	Krakow	Gold	Car	ShortTerm	No	No	YES
3	Military	Munich	Gold	No	No	No	Regular	YES
4	Doctor	Catanzaro	Classic	Car	LongTerm	TechShare	Senior	YES
5	Energy	Poznan	Gold	Apart.	LongTerm	No	No	YES
...	...	...	...	...	...	...	...	...
25	Transport	Cosenza	Classic	Car	ShortTerm	No	Family	NO
26	Police	Tarnow	Gold	Apart.	No	No	No	NO
27	Nurse	Radom	Classic	No	No	No	Senior	NO
28	Education	Catanzaro	Classic	Apart.	No	No	No	NO
29	Transport	Warsaw	Gold	Car	ShortTerm	TechShare	Regular	NO
30	Police	Cosenza	Classic	Car	No	No	No	NO

means for rule generalization; if more general rules were desired, the data should have been manually preprocessed and attribute-values generalized to obtain more general rules and therefore more valuable results.

Table 2. Rules generated by CN2, CN2-SD and g-SEGS from the data in Table 1. Coverage, confidence and WRAcc were computed in postprocessing.

CN2 rules for class big_spender='YES'	**Coverage**	**Confid.**	**WRAcc**
occupation='Doctor'	20.00%	83.33%	0.067
loan='No' ∧ account='Gold'	10.00%	100.00%	0.050
occupation='Health-care'	6.67%	100.00%	0.033
occupation='Education' ∧ account='Gold'	6.67%	100.00%	0.033

CN2-SD rules for class big_spender='YES'	**Coverage**	**Confid.**	**WRAcc**
account='Gold' ∧ investment_fund='No'	33.33%	80.00%	0.100
account='Gold'	46.67%	64.29%	0.067
occupation='Doctor'	20.00%	83.33%	0.067
occupation='Health-care'	6.67%	100.00%	0.033
investment_fund='TechnologyShare' ∧ account='Classic'	13.33%	75.00%	0.033

g-SEGS rules for class big_spender='YES'	**Coverage**	**Confid.**	**WRAcc**
occupation(Public) ∧ bankingService(Gold)	26.67%	87.50%	0.100
bankingService(Gold)	46.67%	64.29%	0.067
occupation(Doctor)	20.00%	83.33%	0.067
occupation(Public) ∧ bankingService(Deposit)	26.67%	75.00%	0.067
occupation(Health)	23.33%	71.43%	0.050
occupation(Doctor) ∧ bankingService(Deposit)	16.67%	80.00%	0.050
location(Bavaria)	16.67%	80.00%	0.050
location(Germany) ∧ occupation(Service) ∧ bankingService(investmentFund)	16.67%	80.00%	0.050

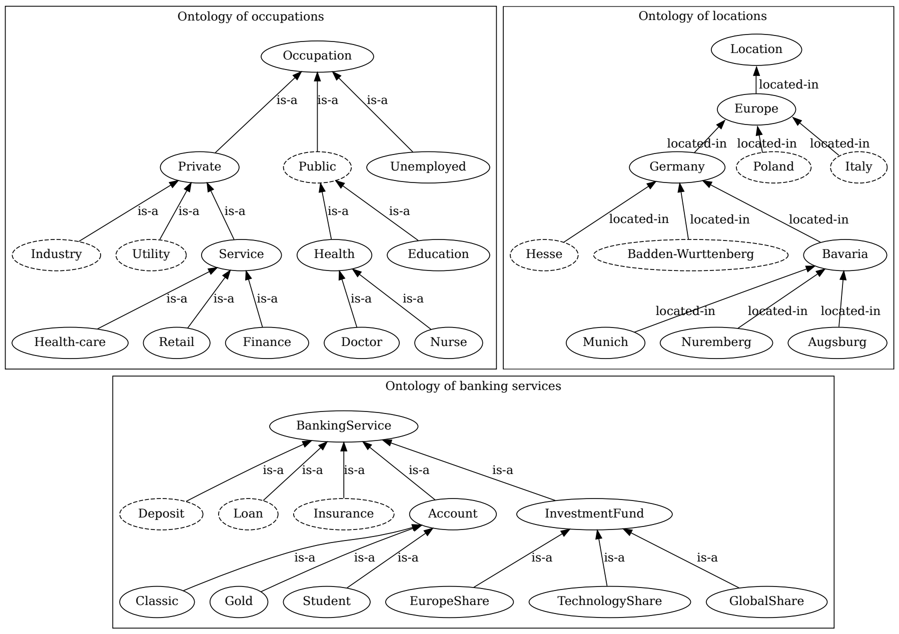


Fig. 3. Ontologies for data in Table 1. Note that these are not the full ontologies, but only the parts needed to interpret the rules presented in this paper. Concepts with omitted subconcepts are drawn with a dashed line.

In semantic data mining using g-SEGS, in addition to the data in Table 1, three ontologies shown in Figure 3 are used as input to introduce semantics into the discovery process. The result of applying g-SEGS to these ontologies and the given training data is presented in the bottom part of Table 2.[7]

The result illustrates the following characteristics of semantic data mining by g-SEGS: (a) Conditions of g-SEGS rules are conjunctions of literals, having ontology terms as arguments of predicates bearing the ontology name (and therefore logically defined semantic meaning), while the conditions of CN2 and CN2-SD rules are conjunctions of attribute-value pairs, (b) g-SEGS rules are more general compared to rules constructed by CN2, CN2-SD or other non-semantic data mining algorithms, and (c) automated and therefore repeatable data preprocessing of data and rules can be performed, less prone to human preprocessing errors.

[7] The same data and background knowledge could also be used for describing credit holders or clients that have closed their account in a bank.

5 Functional genomics use cases

We tested g-SEGS on biological microarray datasets: *acute lymphoblastic leukemia* (ALL) [4] and *human scenescence mesenchymal stem cells* (hMSC) [21]. Both publicly available datasets[8] encode gene expression data for two classes and the challenge is to produce descriptions of differentially expressed genes involved in the process of each domain.

To show the form of rules produced, recall the results of the SEGS system, the ancestor of g-SEGS, on the same dataset from a clinical trial in acute lymphoblastic leukemia (ALL). The ALL dataset was chosen as it is typical for medical research and has a reference role for such evaluations as it has been a model dataset for other microarray data analysis tools as well. The analysis of differences in gene expression between two lymphocyte subtypes (lymphocyte B and lymphocite T) was performed as follows. Genes were first ranked according to their expression value, and differentially expressed genes were selected by gene filtering according to $logFC$ cut-off value $|0.3|$. Three semantic knowledge sources were used as background knowledge to SEGS and g-SEGS: GO, KEGG and Entrez. As, except for GO, these hierarchies are not available in the OWL format, in SEGS a dedicated algorithm for merging these three sources was used to form the joint input database format, which can be chosen as a parameter in g-SEGS, in addition to the default OWL format. Discovered rules, describing subgroups of differentially expressed genes, are formed as conjuctions of terms, e.g., `receptor-binding(G)` $\wedge$ `T-cell-activation(G)`. Similar to previous research, the results show that one of the main differences between differentially expressed and non-differentially expressed gene groups is the expression of major histocompatibility complex (HLA) related genes.

In the experiments with g-SEGS, we used the same preprocessing steps. Genes were first ranked using the ReliefF [17] algorithm and then filtered using the logarithm of expression fold change (logFC). All genes with $|logFC| < 0.3$ were removed from the set, resulting in 9,001 genes in the ALL domain and 20,326 genes in the hMSC domain. The top 300 genes were used as the positive class both in SEGS and g-SEGS, while g-SEGS treats all other examples as negative.

Table 3 present the quality measurements of g-SEGS on the two domains. The discovered rule sets were evaluated using the descriptive measures of rule interestingness as proposed in [11]: $AvgCov$ - the average rule coverage, $AvgSup$ - the overall support, $AvgSig$ - the average significance, $AvgWRAcc$ - the average unusualness and AUC - the area under the convex hull (method 1). Additionally we also measured the execution time t.

The results show that g-SEGS produces more significant rules in the ALL dataset, and that the other results are comparable. Of course we need to take into account that it is not necessary that these measures well reflect the quality of the rule set, i.e. whether they provide novel and interesting knowledge for

[8] http://segmine.ijs.si

	g-SEGS					
Domain	*AvgCov*	*AvgSup*	*AvgSig*	*AvgWRAcc*	*AUC*	t[min]
ALL	0.024	0.770	15.214	0.0012	0.573	11.25
hMSC	0.023	0.700	10.427	0.0005	0.563	10.75

Table 3. Experimental results.

the domain expert. Such an analysis with a domain expert is planned for future work.

6 Conclusions

This paper discusses *semantic data mining* as an adequate approach to face a potential paradigm shift in data mining, addressing the new challenge of mining the knowledge in ontologies, constrained by the empirical evidence in the collected data. In our approach, domain ontologies define the hypothesis search space, and the data is used as means of guiding and constraining the hypothesis search and evaluation.

Prototype semantic data mining system g-SEGS is used to illustrate the approach in a simple semantic data mining scenario and in two real-life functional genomics scenarios. The g-SEGS system takes ontologies in the OWL format and data in a standard attribute-value format as its input, and takes advantage of the hierarchical relationships in ontologies for efficient search and pruning of the hypothesis search sapce. A user friendlly interface is also one of the key features of the g-SEGS system.

There are many possible fields of application of semantic data mining. It can be directly applied to domains where data are characterized by sparsity and taxonomies are available, like market basket analysis, to give an example. We have demonstrated the usefulness of semantic data mining in two real-life functional genomics scenarios where biological ontologies are mined with the support of experimental microarray data. The prototype semantic data mining system g-SEGS shows major advantages compared to non-semantic systems, as more general rules and automated data preprocessing are performed. There are also advantages compared to ILP and other related approaches since our system uses a standardized encoding of knowledge.

A systematic comparison of g-SEGS to the state of the art relational data mining systems is planned in our further work. The first results of comparing g-SEGS to the state of the art ILP system Aleph indicate that using the ontologies in their native format substantially simplifies the system's use in real life scenarios, by reducing the encoding time and ensuring the system's reusability.

Acknowledgments

The research presented in this paper was supported by the Slovenian Ministry of Higher Education, Science and Technology (grant no. P-103) and the EU-FP7 projects e-LICO and BISON.

References

[1] C.C. Aggarwal and H. Wang, editors. Managing and Mining Graph Data. Springer US, 2010.

[2] J.M. Aronis, F.J. Provost, and B.G. Buchanan. Exploiting background knowledge in automated discovery. In *Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 355–358, 1996.

[3] L. Brisson and M. Collard. How to semantically enhance a data mining process? In *Proc. of the 10th International Conference on Enterprise Information Systems, ICEIS 2008*, pages 103–116, 2008.

[4] S. Chiaretti, X. Li, R. Gentleman, A. Vitale, M. Vignetti, F. Mandelli, J. Ritz and R. Foa. Gene expression profile of adult T-cell acute lymphocytic leukemia identifies distinct subsets of patients with different response to therapy and survival. *Blood*, 103, 2771–2778, 2004.

[5] S.H. Clearwater and F.J. Provost. Rl4: A tool for knowledge-based induction. In *Proc. of the 2nd International IEEE Conference on Tools for Artificial Intelligence*, pages 24–30, November 1990.

[6] L. De Raedt. *Logical and Relational Learning*. Springer Berlin Heidelberg, 2008.

[7] G.C. Garriga, A. Ukkonen, and H. Mannila. Feature selection in taxonomies with applications to paleontology. In *Proc. of the 11th International Conference on Discovery Science*, DS '08, pages 112–123. Springer-Verlag, 2008.

[8] T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, M. Caligiuri, C. Bloomfield, and E. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoringt. *Science*, 286:531–537, 1999.

[9] P. Gottgtroy, N. Kasabov, and S. MacDonell. An ontology driven approach for knowledge discovery in biomedicine. In *Proc. of the VIII Pacific Rim International Conferences on Artificial Intelligence (PRICAI)*, 2004.

[10] S.Y. Kim and D.J. Volsky. Page: Parametric analysis of gene set enrichment. *BMC Bioinformatics*, 6(144), 2005.

[11] N. Lavrač, B. Kavšek, P.A. Flach, and L. Todorovski. Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, 5:153–188, 2004.

[12] H. Liu. Towards semantic data mining. In *Proc. of the 9th International Semantic Web Conference (ISWC2010)*, November 2010.

[13] R.S. Michalski. A theory and methodology of inductive learning. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An artificial intelligence approach*, pages 83–129. Palo Alto: Tioga Publishing Company, 1983.

[14] I. Mozetič, N. Lavrač, V. Podpečan, P. Kralj Novak, et al. Bisociative knowledge discovery for microarray data analysis. in *Proc. of the First Intl. Conf. on Computational Creativity*, 190–199, Springer, 2010.

[15] J. Demšar, B. Zupan, and G. Leban. Orange: From experimental machine learning to interactive data mining, white paper (www.ailab.si/orange). Faculty of Computer and Information Science, University of Ljubljana, 2004.

[16] V. Podpečcan, M. Juršič, M. Žakova, and N. Lavrač. Towards a service-oriented knowledge discovery platform. In *Proc. of the ECML/PKDD Workshop on Third-generation data mining: Towards service-oriented knowledge discovery*, 25-36, 2009.

[17] M. Robnik-Šikonja and I. Kononenko. Theoretical and Empirical Analysis of ReliefF and RReliefF. *Machine Learning*, 53 (1-2), 23–69, 2003.

[18] P. Subramanian, P. Tamayo, V.K. Mootha, S. Mukherjee, B.L. Ebert, and M.A. Gillette. Gene set enrichment analysis: A knowledge based approach for interpreting genome-wide expression profiles. *Proc. of the National Academy of Science, USA*, 102(43):15545–15550, 2005.

[19] V. Svátek, J. Rauch, and M. Ralbovský. Ontology-enhanced association mining. In *Semantics, Web and Mining, Joint International Workshops, EWMF 2005 and KDO 2005*, pages 163–179, 2005.

[20] I. Trajkovski, N. Lavrač, and J. Tolar. SEGS: Search for enriched gene sets in microarray data. *Journal of Biomedical Informatics*, 41(4):588–601, 2008.

[21] W. Wagner, P. Horn, M. Castoldi, A. Diehlmann, S. Bork, R. Saffrich, V. Benes, J. Blake, S. Pfister, V. Eckstein and A.D. Ho. *PLoS ONE*, 3 (5), e2213, 2008.

A Browser-based Platform for Service-Oriented Knowledge Discovery

Janez Kranjc, Vid Podpečan, and Nada Lavrač

Jožef Stefan Institute,
Jamova cesta 39, 1000 Ljubljana, Slovenia
`{janez.kranjc,vid.podpecan,nada.lavrac}@ijs.si`
`http://kt.ijs.si`

Abstract. The paper proposes a novel platform for knowledge discovery which is based on modern web technologies, and is implemented as a web application. It is based on the principles of service-oriented knowledge discovery, and features interactive scientific workflows. In contrast with existing comparable platforms, ours is suitable for any knowledge discovery task, offers advanced workflow construction including meta-workflows, can use any existing web service as a workflow processing component, and runs in all major web browsers and platforms, including mobile devices.

Keywords: knowledge discovery, service-oriented architecture, web application, web services, scientific workflows

1 Introduction

This paper presents a platform for designign and running data mining workflows which was designed to overcome recognized deficiencies of existing solutions while retaining all of their important features. As such, the proposed platform benefits from service-oriented technologies, a visual programming paradigm, as well as platform *and* software independent technologies.

First, the visual programming paradigm simplifies the construction of complex knowledge discovery scenarios by providing basic building blocks (widgets) which can be connected and executed, enables repeatability of experiments by saving constructed workflows and parameters, provides an intuitive structuring of complex parts of workflows by introducing the notion of meta-workflow (workflow of workflows), and makes the platform suitable also for non-experts due to the representation of complex procedures as sequences of simple processing steps. Along with the presented platform, notable application that employ the visual programming paradigm include Weka [1], Orange [2], KNIME [3] and RapidMiner [4]. The most important common feature is the implementation of a *workflow canvas* where workflows can be constructed using drag, drop and connect operations on the available components.

Secondly, service-oriented architecture featuring web services as processing components enables parallelization, remote execution, and high availability by default, provides access to large public (and proprietary) databases, enables easy integration of third party components (as services) and loose coupling, and supports not only distributed processing but also distributed development. Platforms such as Weka4WS [5], Orange4WS [6], Web Extension for RapidMiner and Taverna [7] allow the integration of web services as workflow components. With the exception of Orange4WS and Web Extension for RapidMiner, these applications are mostly specialized for domains like systems biology, chemistry, medical imaging, ecology and geology. None of these applications are browser based and still require specific hardware and software.

Finally, as the platform and software independence can be achieved by using web technologies only, our platform relies on standards such as HTML, CSS, Ajax and Javascript, and widely supported and accepted software solutions such as Apache and PHP.

To summarize, the presented platform offers a complete service-oriented workflow environment, suitable for any knowledge discovery task. The platform is truly independent as it is implemented in the form of a web application which is accessible from any modern web browser. The functionalities and the design of the platform are presented in this paper along with an illustrative use case. A detalied description of two additional practical use cases may be found on the authors' website[1].

2 Platform design and functionalities

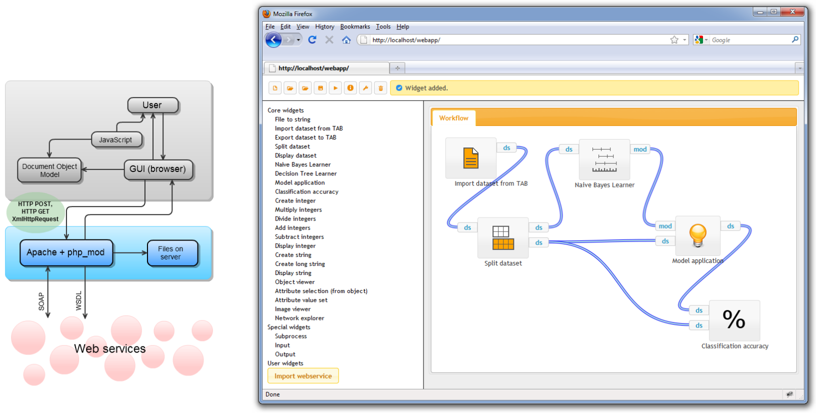
The presented platform consists of three layers as shown in Figure 1(a). The uppermost layer presents the parts of the platform which run at the client side. The middle layer is located on the server where the platform is hosted. The bottom layer consists of the remote resources which provide web services.

The graphical user interface was implemented in HTML and JavaScript. Two JavaScript libraries were used to implement the interactivity of the GUI. The jQuery UI library[2] provided dragging and dropping functionalities used for manipulating the canvas. The jQuery library[3] was used to handle mouse and keyboard events, and asynchronous invocation of server side scripts. The GUI consists of three main parts: the toolbar, the widget repository, and the canvas. A sample screenshot of the user interface is shown in Figure 1(b).

The toolbar is used to start, execute, save and load workflows or parts of workflows. The primary function of the toolbar is to start, execute, save, and load workflows, and to separate parts of the workflow.

The widget repository provides a clickable list of available widgets. By clicking on a widget, its instance appears on the canvas hosting the currently active workflow. The repository can be expanded by the user by saving parts of a workflow as a new widget or importing new widgets from web services.

[1] http://kt.ijs.si/janez_kranjc/usecases.html
[2] http://jqueryui.com/
[3] http://jquery.com/



(a) (b)

Fig. 1. The three layered design of the platform (a) and a screenshot of the environment in the Mozilla Firefox browser (b).

The workflow canvas is used for connecting widgets into a workflow. Widgets can be connected by clicking on their inputs and outputs. When both an input and an output are selected, an event is triggered which checks for cycles in the workflow graph using the depth first search algorithm. If no cycles are detected, a connection is drawn and the corresponding widgets become connected.

The execution of the entire workflow is realized by a special JavaScript function, which iteratively searches for widgets whose predecessors have finished their execution, and executes them.

To execute a widget, the platform passes all inputs of a widget to the corresponding server side script using asynchronous HTTP POST requests. When the results are available (or when an error occurs), a call-back function is called which stores the results of the execution of the widget into the output variables in the underlying document object model. The script may either return the data in a serialized form or issue a special command which instructs the user interface to open a pop-up window for displaying the results (data visualization widgets utilize this functionality). The execution of multiple independent widgets simultaneously is assured by the asynchronous nature of POST requests.

Workflow components of the presented platform may be implemented as remote web services provided by a third party, or as PHP scripts located on the server hosting the platform.

Since web services are completely defined by their WSDL descriptions, the functionality to import web services was implemented in PHP by parsing the corresponding WSDL document. For every operation provided by a web service the PHP script returns an HTML description of the corresponding widget. In the user interface, this procedure is accessible through a button whose event handler queries the user for the location of a WSDL file which is then imported and parsed, and a list of available widgets is added to the repository.

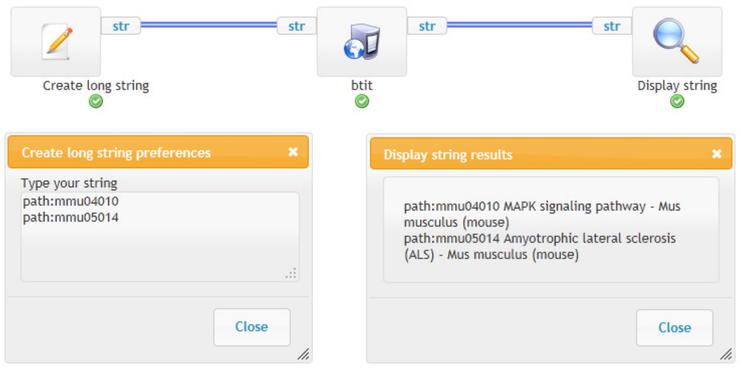
3 An illustrative use case

In order to demonstrate some of the abilities of the service oriented browser based platform a simple use case is presented. The workflow in this use case is based on a simple workflow available on the *myExperiment* website[4].

The workflow utilizes a publicly avaiable web service for querying the KEGG database (Kyoto Encyclopedia of Genes and Genomes) to retrieve definitions of given database entries.

First, the KEGG API (a SOAP web service)[5] was imported in our workflow nvironment by providing the location of the WSDL document. The import operation resulted in a collection of widgets in the repository representing all KEGG web service operations. Then, an instance of the *btit* widget representing *btit(string:str)* operation of the KEGG API was created by clicking on its name in the repository.

The *btit()* operation expects database entries as an input which we provided by using a local widget for composing strings. The results of the operation (the definitions of database entries) were displayed in a local widget for displaying text. The workflow and examples of its input and output data are shown in Figure 2.



path:mmu04010
path:mmu05014

path:mmu04010 MAPK signaling pathway - Mus musculus (mouse)
path:mmu05014 Amyotrophic lateral sclerosis (ALS) - Mus musculus (mouse)

Fig. 2. A simple workflow for querying the KEGG database using the *btit()* operation. The input and output are handled by local string operations widgets. A sample query and the result are also shown.

[4] http://www.myexperiment.org/workflows/1099.html
[5] http://soap.genome.jp/KEGG.wsdl

4 Conclusions

The paper presents a browser-based platform for service oriented knowledge discovery which relies on modern web standards and widely supported and accepted software solutions. Coupled with the extreme versatility and power of web services, the platform presents a new generation tool, ready to be used by researchers and students in any scenario or form of knowledge discovery, including mining of web and data streams thus surpassing all currently available knowledge discovery software tools. Moreover, the proposed environment is able to run in all modern web browsers, including those available on mobile devices, which presents great opportunities for its deployment and widespread use.

In future work we will explore adding means of mining data streams as well as semi-automatic workflow construction based on planning algorithms, modern knowledge discovery ontologies, and systems for semantic annotation of web services. Moreover, additional built-in workflow components as well as web services, specialized for data mining, text mining, and systems biology are also under development. Finally, we plan to provide a public installation of the environment, a workflow repository, a community web site, and release the sources under an open-source license.

Acknowledgments. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) within the context of the project FIRST, Large scale information extraction and integration infrastructure for supporting financial decision making, under grant agreement n. 257928.

References

1. Witten, I. H. and Frank, E. (2005) *Data Mining: Practical machine learning tools and techinques, 2nd edition.* Morgan Kaufmann.
2. Demšar, J., Zupan, B., and Leban, G. (2004). *Orange: From experimental machine learning to interactive data mining.* White Paper.
3. Berthold, M. R., Cebron, N., Dill, F., Gabriel, T. R., Ktter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K., and Wiswedel, B. (2007) KNIME: The Konstanz information miner. *Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007).* Springer.
4. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., and Euler, T. (2006) Yale: Rapid prototyping for complex data mining tasks. In Ungar, L., Craven, M., Gunopulos, D., and Eliassi-Rad, T. (eds.), *KDD '06: Proceeding of the 12th ACMS SIGKDD international conference on Knowledge discovery and data mining*, New York, NY, USA, August, pp. 935-940. ACM.
5. Talia, D., Trunfio, P., and Verta, O. (2005) Weka4WS: A WSRF-enabled Weka toolkit for distirbuted data mining on grids. *Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pp. 309-320.

6. Podpečan, V., Juršič, M., Žáková, M., Lavrač, N. (2009) Towards a service-oriented knowledge discovery platform. In Podpečan, V., Lavrač, N., Kok, J. N., Bruin, J. (eds.). *Third generation data mining: towards service-oriented knowledge discovery, SoKD'09* : September 7, 2009, Bled, Slovenia. [S. l.: s. n.], 2009, pp. 25-38
7. DeRoure, D., Goble, C., and Stevens, R. (2008) The design and realisation of the myExperiment virtual research environment for social sharing of workflows. *Future Generation Computer Systems*, 25, 561-567.

Web Services for Stream Mining: A Stream-Based Active Learning Use Case

Martin Saveski and Miha Grčar

Jožef Stefan Institute, Ljubljana, Slovenia
{Martin.Saveski, Miha.Grcar}@ijs.si

Abstract. The nature of data on the Web is becoming more and more stream-oriented and in this context, the idea of mining Web-generated data streams is becoming a hot topic. Web services, on the other hand, have become an inevitable tool for the future development of the Web. While Web services have been very successful in providing distributed computing environments, they have not been exploited for building and executing stream mining workflows. In this paper, we discuss a service-based environment suitable for stream mining and present a service-oriented stream mining workflow for sentiment classification through active learning. In the context of this use case, we present the general idea of active learning as well as an empirical evaluation of several active learning methods on a stream of opinionated Twitter posts.

Keywords: stream data mining, Web services, active learning, sentiment analysis, Twitter.

1 Introduction and Motivation

Handling vast Web-generated streams is a relatively new challenge emerging mainly from the self-publishing activities of Web users (e.g., blogging, twitting, and participating in discussion forums and social networks). Furthermore, news streams (e.g., Dow Jones, Business Wire, Bloomberg, Reuters) are growing in number and rate, which makes it impossible for the users to systematically follow the topics of their interest. The number, heterogeneity, and rate of streams, "produced" by the Web, are expected to grow substantially in the upcoming decades. In accordance with the "Web of Things"[1] vision, many devices (such as wireless sensors and even household appliances) are expected to be connected to the Web, producing data streams for the purpose of surveillance (incl. alerting and visualization) and data analysis (such as stock price prediction and monitoring, environmental and traffic monitoring, and vital signs monitoring). From this perspective, we can talk about a "Web of Streams".

In the European project FIRST (Large-scale information extraction and integration infrastructure for supporting financial decision making), the goal is to provide an infrastructure for analyzing vast streams of user-generated Web content and news feeds from the domain of financial markets. The integration infrastructure devised in

[1] Web of Things, http://en.wikipedia.org/wiki/Web_of_Things

the project will be based on SOA[2] principles. Service-oriented architectures are more and more often the technological choice for software integration and exposure of public interfaces (e.g., Yahoo!, Google, Twitter, and eBay APIs). In a standardized manner, SOA allows software interoperability across multiple, separate systems and domains. Furthermore, running on the provider's servers, it hides the proprietary implementations from the user, ensures the sufficient hardware resources, and allows the (indirect) use of proprietary data. However, SOAs suffer from several drawbacks in stream-based real-time scenarios, which have to be accounted for as further discussed in Section 2.

In this paper, we present a use case of service-oriented stream-based active learning for the purpose of sentiment analysis. The aim of sentiment classification is to decide, given a fragment of text or full text, whether the sentiment attributed to the object discussed in the text is positive or negative. The approaches to sentiment classification are in general either rule-based [1] or based on machine learning [2, 3]. We use a machine-learning approach by training a classifier on a labeled dataset of messages (i.e., tweets) posted on Twitter. One of the main problems in this setting is the lack of labeled data, especially in the domain of financial markets. Creating labeled datasets manually is an expensive and time-consuming process as it requires the active participation of domain experts. To reduce these costs, we employ active learning, a machine-learning technique designed to actively query the domain expert for new labels by putting forward data instances that, upon being labeled, contribute most to the model being built. In effect, after a certain relatively low amount of carefully chosen instances were labeled, the model performs better than if the same amount of instances were selected randomly. Our design of putting an active learning "loop" into a stream-based SOA is discussed in Sections 3 and 4.

2 Web Services for Stream Mining

Service-oriented architectures suffer from several drawbacks in specific scenarios. In their traditional form, they employ a request-response protocol, such as REST[3] or SOAP[4]. If the requests and responses are large and frequent, the connection between the client and the server most likely represents a bottleneck. In data mining workflows, inputs to elementary services (e.g., a labeled dataset required by an algorithm for building a classification model) and their outputs (e.g., the parameters of a classification model) are often very large (thousands or millions of data instances and model parameters). This situation is illustrated in Figure 1. In stream mining workflows, the inputs and outputs are usually smaller because only the newly arrived data instances and changes to the models can be passed around. However, the frequency tends to be much higher and is in fact posed by the input stream speed. In both of these two scenarios, a lot of (unnecessary) traffic happens between the client and the server and can result in a bottleneck.

[2] Service-Oriented Architecture, http://en.wikipedia.org/wiki/Service-oriented_architecture
[3] Representational State Transfer, http://en.wikipedia.org/wiki/Restful
[4] Simple Object Access Protocol, http://en.wikipedia.org/wiki/Soap

Two relatively simple techniques, pipelining and parallelization, can be employed to increase the throughput of a stream mining workflow. Pipelining refers to the parallel execution of elementary services in the "horizontal" direction (i.e., one after another). Even though the services are executed one after another, the pipeline maximizes the throughput by processing several data units at the same time. Alternatively or even in addition, parallelization in the "vertical" direction can be implemented so that two or more elementary services, either processing the same data unit or performing load balancing, are executed simultaneously. In the SOA paradigm, these techniques need to be implemented on the client side which creates a lot of (unnecessary) engineering overhead. Furthermore, as the server (or the service provider) is unaware of the workflow which is defined on the client side, it is difficult to optimize the workflow execution and workflow topology on the server side in order to increase the quality of service (QOS).

Last but not least, because streams are infinite (i.e., continuously flowing into the system) and the client is acting as a broker, the client is required to have constant access to the services. This is highly unrealistic, especially if the client is not a server machine (which is usually the case) and is eventually shut down or loses internet connection.

The solution to the discussed problems is to employ a publish-subscribe mechanism (such as ZeroMQ[5] and Java Messaging Service [14, 15, 17]) that allows elementary services to communicate with each other directly without the client acting as a broker. In this setting, the client first builds (instantiates) the workflow, which can be distributed across several servers, by "subscribing" inputs of services to outputs of services preceding them in the workflow. The client then uploads the data (and other required parameters) to the first service in the workflow. When the request is successfully processed by the first service, the results are passed directly to the next service (or services) in the workflow without the client's intervention. The client can at any time query the services about the status of its request and eventually download the results stored on the server by the last service in the workflow. The "heavy" client-server network traffic is thus limited to the upload of the data and parameters at the beginning and the download of the results when the workflow finishes processing the request.

When dealing with streams, there are two alternatives to how a data stream enters a workflow. It can either be provided by the client (constantly sending updates to the workflow) or it can be obtained from the Web by one of the services. Our use case luckily falls into the second category: the data is obtained from the Twitter API[6]. This means that after the client instantiates and configures the workflow, it can disconnect from the server without shutting down the workflow. The data-mining models built by such stream-based workflow are constantly being updated (to be up-to-date with the stream) and can at any time be queried by the client. This is illustrated in Figure 2. Apart from this clear advantage of the publish-subscribe mechanism, pipelining and parallelization come naturally with the way the inter-service communication works. Furthermore, a server is aware of the workflow fragments built by the users with the services hosted by the server. It is also aware of the other servers to which the outputs

[5] ZeroMQ, http://www.zeromq.org/

[6] Twitter API, http://dev.twitter.com/doc

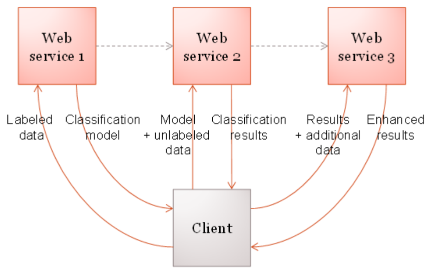


Fig. 1. The main shortcoming of Web services in data mining workflows: a lot of data is passed between the client and the server(s).

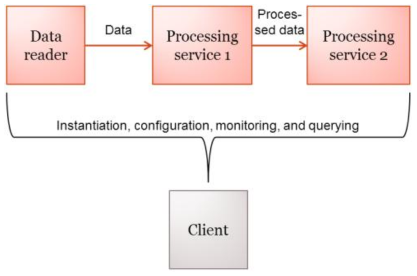


Fig. 2. The publish-subscribe paradigm in the stream-based setting: severely reduced traffic between the client and the server(s). Clearly advantageous in stream-based service-oriented environments.

are sent and from which the inputs are received in the case of a distributed environment. This allows the service owners to analyze the workflows and optimize either the elementary services that represent bottlenecks or the entire topology if, for example, the bottleneck results from the communication between the servers.

In the following sections, we present a use case for such a service-oriented publish-subscribe environment. We first present the individual services and then the workflow for stream-based active learning for building sentiment classification models.

3 Stream-Based Active Learning Workflow

In this section, we present a particular use case based on the service-oriented principles discussed in the previous sections. Specifically, we present a service-oriented workflow which dynamically builds and applies a model for sentiment classification of financial Twitter posts. To deal with the lack of manually labeled data and the dynamic nature of the data stream, we include active learning as one of the main components in the workflow. The concepts of active learning and sentiment analysis are further discussed in Section 3. In addition, we present the workflow components: Twitter API, language detector, near-duplicate detector, and active learner in more detail. Each component is implemented as a separate Web service and directly communicates with the subsequent service without the client acting as a broker. Thus, for instance, the output of the language detector component is provided as an input to the near-duplicate detector. In addition, each component provides an interface through which the client can receive a status report or configure the service.

Twitter API. Since we are interested in analyzing financial Twitter posts (tweets), the main data resource in our workflow is the Twitter API. By the informal Twitter conventions, all tweets discussing stocks contain “$” as a prefix to the stock symbol which is discussed. For instance, the “$GOOG” tag indicates that the tweet discusses the Google stocks or the “$AAPL” tag refers to the Apple stocks. This convention makes it easy to retrieve financial tweets. Twitter provides three types of APIs: REST, Streaming, and Search API. To collect as much tweets as possible, we combine the Streaming and Search API. Through the Search API, we constantly poll for a predefined set of stock symbols and through the Streaming API, we consume the Spritzer tweet stream, approximately 1% of all public tweets, and we filter out the non-financial tweets.

Language Detection. We have observed that many of the collected tweets are not in English. To ensure a better performance of the text processing components in our workflow, we have constrained the workflow to process only English tweets. Although the Twitter API provides the information about the language of a particular Twitter user, this information is often incorrect (e.g., non-English speakers often "tweet" in English). Therefore, we have developed a custom n-grams-based language detection model [4]. N-grams are *n* characters long sequences created by slicing up the text tokens. Using several text corpora in the languages we want to be able to detect, we developed a *profile* – histogram of n-gram frequencies – for each language. Thus, to detect the language of a tweet, we count the n-gram occurrences and we find the profile which makes the best match. Tweets that do not match the English language profile are discarded.

Near-duplicate Detector. We also noticed that tweets with very similar content occur many times in the stream. We observe that this is mainly caused by re-tweets and spam. Twitter provides a feature with which users can re-tweet the posts of other users, i.e. tweet the same post, but with "RT" tag and a link of the original user. Spammers, on the other hand, flood the stream with tweets posted from different accounts, but with very similar content. These tweets represent noise and may negatively influence the performance of the subsequent components in the workflow. Since using simple hashing of the tweets will not allow us to detect such tweets, we have employed the near-duplicate detection algorithm proposed in [18]. Specifically, we represent each tweet as a set of 5-shingles, i.e. set of all 5-character sequences contained in the tweet, and compute the Jaccard similarity of the shingle sets. If this similarity is above a given threshold, we consider the tweets as near-duplicates. By default, this method would require that each new tweet in the stream is compared to all the exiting tweets, which is unrealistic for the fast stream we have in this use case. To minimize the number of comparisons, we constantly keep an inverted index, which as keys has bi-grams (two word sequences) and as values has a set of tweets where the bi-gram is contained. Thus, when a new tweet arrives, we use the inverted index to retrieve a set of candidate near-duplicates and we only compare those to the tweet currently being processed.

Active Learning. This component implements the active learning principle and its output is a model for sentiment classification of financial tweets. The component keeps three pools of tweets: labeled, unlabeled, and query tweets. All tweets labeled so far are placed in the pool of labeled tweets. The pool of unlabeled tweets contains the most recent unlabeled tweets which are the candidates for the query pool. As the stream flows into the system, this pool is updated: new tweets come in, old flow out. The query pool, on the other hand, contains all unlabeled tweets which, according to the current model, if labeled will improve the model the most. Every new tweet in the

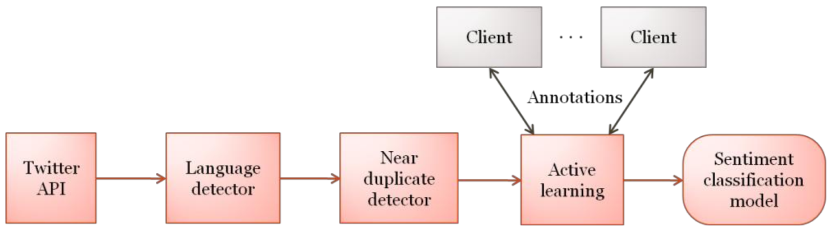


Fig. 3. Our workflow for stream-based active learning.

stream, based on the current model, is either placed into the query or unlabeled pool. With every new labeled tweet, the model is updated and accordingly, the query pool is changed. This component also exposes a Web interface through which the domain experts can label tweets (depicted as Client in Figure 3). The tweets to be annotated are taken from the query pool. The domain expert is shown one tweet at a time and can either provide a label or ask for another tweet if he is unsure about the label. This feedback is afterwards propagated to the model. As shown in the figure, many domain experts can provide annotations simultaneously. More details about the idea of active learning for sentiment analysis and the methods which we have considered for this purpose are provided in the next sections. It is important to note that due to the dynamic nature of the data, this component must include not only the application of a previously trained model, but also the model building phase. The financial sentiment indicators are constantly changing their polarity and the model must be at all times updated and in line with the current "events" evident from the stream in order to correctly capture these changes. For instance, if the word "Greece" is seen in a tweet, due to the current financial crises in the country, it may be considered as a negative financial indicator. However, before the crises, the same term was most likely a signal of neutral or positive sentiment (e.g., history and culture, holidays).

4 Active Learning for Sentiment Analysis

4.1 Sentiment Analysis

Textual information can generally be of two main types: facts and opinions. While facts consist of objective expressions, opinions are typically subjective expressions that represent people's sentiment, views, or impressions towards different events, entities, and their properties. Although factual information is very important, it is in the human nature to be interested in other people's opinions. Opinions are so important to us that every time we need to make a decision, we seek the opinions of others on the matter.

With the emergence of the Web 2.0[7], the amount of user-generated content on the Web has rapidly increased, resulting in large amounts of opinionated text. The social networks such as Twitter, Facebook, MySpace, and Digg have allowed the "word of

[7] Web 2.0, http://oreilly.com/web2/archive/what-is-web-20.html

mouth" to spread with enormous speed. This phenomenon has triggered a whole new track of research called sentiment analysis or opinion mining (in the remainder of the text we use the first term). The main problem that sentiment analysis tries to solve is to extract sentiment from text and to detect its polarity (positive or negative). Other, more complex definitions may also involve detecting the object and the object feature towards which the sentiment is expressed as well as the opinion holder.

In the literature, we generally observe two approaches to solving the problem of sentiment analysis: (1) the natural language processing (NLP) and (2) the machine learning (ML) approach. The NLP approach is mostly unsupervised and uses advanced linguistic analysis (dependency parsing), rules, and knowledge resources (WordNet) to analyze opinionated text [5, 6]. The ML approach, on the other hand, takes a supervised learning approach and defines the problem as a classification task where documents/sentences are classified into predefined categories (positive, negative, and possibly neutral), based on their sentiment. However, in some studies, these approaches are combined, for example, part-of-speech tagging can be used to construct the features used for classification [7]. With the large availability of labeled data from Web sites like Trip Advisor, IMDB, Epinions, etc., the number of studies which successfully apply the ML techniques has grown significantly [8].

In this study, we perform sentiment analysis of Twitter posts (tweets). With 145 million users (reported in September 2010[8]) and growing, Twitter is no doubtingly one of the most popular sites on the Web. Moreover, it has promoted itself as a platform where people express their personal opinions and sentiment towards companies, products, events, etc. While tweets come with a lot of metadata (user information, topic tags, references to other users), the nature of the data poses some specific challenges. Tweets can be up to 140 letters long (1–2 sentences) and usually contain informal language and expressions (slang). Consequently, this makes most of the standard NLP techniques less applicable or not applicable at all. Therefore, we apply the ML approach to sentiment analysis and we define the problem as a document classification problem. However, the limited amount of manually annotated tweets makes the classical supervised learning techniques less applicable and demands the use of other more sophisticated techniques such as active learning.

4.2 Active Learning

Many of the supervised learning systems, especially in text mining, need hundreds or even thousands of labeled examples to achieve good performance. Sometimes these labels come at little or no cost at all, for example when we flag emails as spam or when we rate movies on our favorite social networking Web site. But many times, acquiring labeled instances for more complex learning tasks can be time-consuming, difficult, and expensive. On the other hand, acquiring large amounts of unlabeled instances can be easy and without any costs. The key hypothesis of active learning is that: if the learning algorithm is allowed to choose the data from which it learns – to be "curious" – it will perform better with less training [9].

[8] Twitter Statistics, http://blog.twitter.com/2010/09/evolving-ecosystem.html

As mentioned in the previous section, in our particular use case, we are not able to build an accurate sentiment classification model because we lack the availability of tweets manually annotated with their sentiment polarity. On the other hand, we can easily consume the Twitter stream of public tweets and acquire plenty of unlabeled instances at no cost. Therefore, we use the unlabeled data and active learning to build an accurate model while minimizing the number of labeled examples and thus the associated annotation costs. We have looked at several active learning methods: Active Learning with Support Vector Machines, Hierarchical Sampling for Active Learning, and K-Means Clustering and SVMs for Active Learning.

In [10], the authors propose an algorithm for Active Learning with Support Vector Machines (SVM). In each iteration of the algorithm, an SVM is trained on the data labeled so far and the instance whose feature vector is closest to the hyperplane is queried next. The main idea behind the algorithm is that choosing the instance closest to the hyperplane will maximally reduce the version space, the space of all possible hyperplanes, thus making a more efficient search through this space to find the best hyperplane. They empirically show that the algorithm significantly reduces the number of instances needed to train an efficient SVM model.

However, in [11] the authors observe that in each iteration of a closest-to-the-boundary (boundary being a hyperplane in the previously discussed method) algorithms, the selected instances increasingly diverge from the underlying data distribution, resulting in *sampling bias*. Thus, the sampled subset of instances is hardly representative. They propose a method which starts by building a hierarchical clustering tree from the data and, given the labeled instances so far, tries to find the best pruning of the tree.

The Active Learning with SVMs (AL-SVM) algorithm starts by querying random instances to set the initial hyperplane. In an attempt to combine the two approaches, the efficient search through the hypothesis space and the exploitation of the clustering structure of the data, we have augmented the AL-SVM algorithm to include k-means clustering as the initial step. To make a better sampling of the space, instead of selecting random data instances, we first cluster the data into *k* clusters, where *k* is the number of samples we want to take, and take the medoids of the clusters as the first set of instances to be labeled.

4.3 Experiments

To measure the performances of each of the active learning algorithms discussed in the previous section, we developed a data set of noisy labeled tweets. We consumed the Twitter Spritzer stream (~1% of all public tweets), from 23rd of February to 5th of April, and collected 50 million tweets. We used the positive (:), :-), :D, ;), etc.) and the negative (:(, :-(, :'(, etc.) emoticons to assign labels to the tweets. Tweets which contained no emoticons or both positive and negative emoticons were ignored. The idea of using emoticons for tweet sentiment annotation has already been used in several other studies [12, 13]. To further process the tweets, we have used the language detection and near-duplicate removal workflow components (Section 3), filtering out the non-English and the near-duplicate tweets. This resulted in a set of 703,584 positive and 189,695 negative tweets or 881,069 tweets in total. As it can be

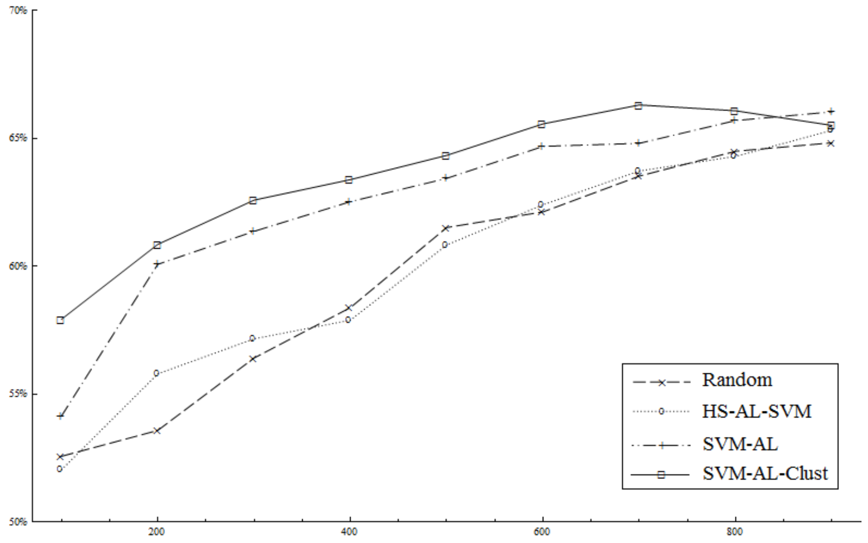


Fig. 4. Comparison between the performances of the different active learning algorithms (*x*-axis: number of labeled instances, *y*-axis: accuracy obtained with a10-fold cross-validation).

seen, the data set is highly unbalanced and the positive tweets dominate. Although this may depict the real distribution of the sentiment polarity in the stream, in these experiments we are more interested in measuring the ability of each algorithm to predict the sentiment polarity according to the features and not the prior probabilities. Thus, we balanced the data set so that it contains the same number of positive and negative tweets. All negative tweets were retained, while the positive ones were evenly sampled, resulting in a data set of 379,390 tweets. To carry out the experiments, we used the temporal meta-information provided with the tweets to simulate a data stream. In this way, we ensured that the experimental setting is the same as the one encountered in a real-life application.

Figure 4 depicts the performance of each of the active learning methods. The *x*-axis shows the number of instances sampled, while the *y*-axis shows the classification accuracy of the algorithms obtained with a 10-fold cross-validation. For reference, we have also the classification accuracy of the random selection policy which represents passive learning.

In all the experiments, we employed SVM for the sentiment classification. It is important to note that the model needs to be updated every time a new instance is labeled. Using the standard SVM implementations, such as SVM^{light} or LIBSVM, would require re-training on all instances labeled so far, which is computationally too expensive to handle the fast data stream in real time. Instead, we employed the incremental SVM implementation proposed in [16], which incrementally trains the model one instance at a time. In the case of hierarchical sampling (HS-AL-SVM), we used the sampling method for choosing instances to be labeled and we used the labeled instances to train an SVM model and test its performance. For this experiment, we have used the open-source implementation made publicly available by the authors[9].

As depicted in Figure 4, the HS-AL-SVM method shows poor performance and fails to improve the random sampling. In our opinion, this is a result of the nature of the Twitter data (short texts) which makes it hard for this method to find patterns. The SVM-AL, on the other hand, shows significant improvement, ranging from 3% to 7.5%, over the random sampling. It is also interesting to note that by performing clustering instead of random sampling, as the initial step of SVM-AL, the classification performance in the first iterations is significantly improved. This also further influences the performance in several next iterations, showing an improvement over SVM-AL. Finally, we observe that as the number of instances increases, the differences in performance of all methods (incl. Random) decreases. Thus, in the case when more than 1,000 tweets are labeled, employing active learning loses its advantage over a passive learner.

5 Conclusions and Future Work

In this paper, we discussed service-oriented stream mining workflows. We pointed out several drawbacks of traditional SOAs when mining Web-generated data streams and explained how the publish-subscribe mechanism counters these shortcomings. Furthermore, we presented a use case on building a sentiment classification model from tweets in the domain of financial markets. Since the required training data (i.e., sentiment-labeled tweets about stocks and companies) is not available, we resorted to active learning (AL) to reduce the cost of labeling the data manually.

Our preliminary experiments showed that AL helps significantly when only a few tweets (e.g., 100–200) are labeled. After 200 tweets are labeled, the accuracy of the SVM-AL-Clust algorithm is 7.5% higher when compared to the random selection policy. Unfortunately, when more and more tweets are labeled, the differences between the evaluated algorithms (incl. Random) diminish. Also, the tested algorithms fail to "pick up" rapidly and after labeling 800 tweets, the accuracy of any of the algorithms (incl. Random) accounts for roughly 85% of the final accuracy achieved by labeling the entire dataset (i.e., 379,390 tweets). We believe that, to some extent, this is because tweets are very short texts and consequently, given a small number of labeled tweets, the resulting bag-of-words space has a relatively low number of dimensions (i.e., words and n-grams). This makes it difficult for SVM to model the sentiment vocabulary early in the process.

Our next step is to measure the dimensionality of the bag-of-words space in each iteration of the AL loop. In addition, we will assess the orthogonality of the test set with respect to the training set. We assume that by putting forward the tweets that reduce the orthogonality between the training and test set, the model's accuracy will increase faster. On the other hand, the resulting bag-of-words space tends to be extremely sparse. We will employ various dimensionality reduction techniques in an attempt to reduce the orthogonality between the training and test set even further. Last

[9] Hierarchical sampling implementation, http://cseweb.ucsd.edu/~djhsu/codes.html

but not least, we will employ transductive learners in an attempt to make use of unlabeled data as well.

Acknowledgments. This work has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) in the context of the project FIRST under Grant Agreement n. 257928.

References

1. Das, S. R., Chen, M. Y.: Yahoo! for Amazon: Sentiment Extraction from Small Talk on the Web. 53 (9), pp. 1375-1388. (2007)
2. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? Sentiment classification using machine learning techniques.Proceedings of EMNLP'2002. (2002)
3. Dave, K., Lawrence, S., Pennock, D. M.: Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. Proceedings of the 12th WWW. (2003)
4. Cavnar, W. B., Trenkle, J. M.: N-Gram-Based Text Categorization. Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval. (1994)
5. Wilson, T., Hoffmann, P., Somasundaran, S., Kessler, J., Wiebe, J., Choi, Y., Cardie, C., et al.: OpinionFinder: A System for Subjectivity Analysis. Learning, 2-4. ACL. (2005)
6. Esuli, A., Sebastiani, F.: SentiWordNet: A publicly available lexical resource for opinion mining. Proceedings of LREC. (2006)
7. Barbosa, L., Feng, J.: Robust sentiment detection on Twitter from biased and noisy data. In Proceedings of COLING. (2010)
8. Pang, B., Lee, L.: Opinion Mining and Sentiment Analysis. 2 (1-2), pp. 1-135. (2008)
9. Settles, B.: Active Learning Literature Survey. University of Wisconsin-Madison, Computer Sciences Technical Report. (2008)
10. Tong, S., Koller, D.: Support Vector Machine Active Learning with Applications to Text Classification. ICML, (2000)
11. Dasgupta, S., Hsu, D.: Hierarchical sampling for active learning. Proceedings of the 25th ICML, p.208-215, Helsinki, Finland. (2008)
12. Bifet, A., Frank, E.: Sentiment Knowledge Discovery in Twitter Streaming Data. Discovery Science 6332, 1-15. (2010).
13. Go, A., Bhayani, R., Huang, L.: Twitter Sentiment Classification using Distant Supervision. Processing 1-6 (2009).
14. Eugster, P.T., Felber, P.A., Guerraoui, R.,Kermarrec, A. M.: The many faces of publish/subscribe. ACM Computing Surveys 35, 114-131 (2003).
15. Baldoni, R., Virgillito, A.: Distributed event routing in publish/subscribe communication systems: a survey. DIS Universita di Roma "La Sapienza". Technical Report (2005).
16. Cauwenberghs, G., Poggio, T.: Incremental and Decremental Support Vector Machine Learning. Neural Information Processing Systems (NIPS) (2001).
17. Eggen, R., Sunku, S.: Efficiency of SOAP Versus JMS.International Conference on Internet Computing, pages 99–105 (2003).
18. Broder, A., Glassman, S., Manasse, M., Zweig, G.: Syntactic Clustering of the Web. International World Wide Web Conference (Apr. 1997), 393-404. (1997).