Expert Systems with Applications 42 (2015) 6442–6456

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Wordification: Propositionalization by unfolding relational data into bags of words

Matic Perovšek [a,b,*], Anže Vavpetič [a,b], Janez Kranjc [a,b], Bojan Cestnik [a,c], Nada Lavrač [a,b,d]

[a] Jožef Stefan Institute, Ljubljana, Slovenia
[b] Jožef Stefan International Postgraduate School, Ljubljana, Slovenia
[c] Temida d.o.o, Ljubljana, Slovenia
[d] University of Nova Gorica, Nova Gorica, Slovenia

ARTICLE INFO

Article history:
Available online 24 April 2015

Keywords:
Wordification
Inductive Logic Programming
Relational Data Mining
Propositionalization
Text mining
Classification

ABSTRACT

Inductive Logic Programming (ILP) and Relational Data Mining (RDM) address the task of inducing models or patterns from multi-relational data. One of the established approaches to RDM is propositionalization, characterized by transforming a relational database into a single-table representation. This paper presents a propositionalization technique called *wordification* which can be seen as a transformation of a relational database into a corpus of text documents. Wordification constructs simple, easy to understand features, acting as words in the transformed Bag-Of-Words representation. This paper presents the wordification methodology, together with an experimental comparison of several propositionalization approaches on seven relational datasets. The main advantages of the approach are: simple implementation, accuracy comparable to competitive methods, and greater scalability, as it performs several times faster on all experimental databases. Furthermore, the wordification methodology and the evaluation procedure are implemented as executable workflows in the web-based data mining platform ClowdFlows. The implemented workflows include also several other ILP and RDM algorithms, as well as the utility components that were added to the platform to enable access to these techniques to a wider research audience.

1. Introduction

Standard propositional data mining algorithms, included in established data mining tools like Weka (Witten, Frank, & Hall, 2011), induce models or patterns learned from a single data table. On the other hand, the aim of Inductive Logic Programming (ILP) and Relational Data Mining (RDM) is to induce models or patterns from multi-relational data (De Raedt, 2008; Džeroski & Lavrač, 2001; Lavrač & Džeroski, 1994; Muggleton, 1992). Most types of propositional models and patterns have corresponding relational counterparts, such as relational classification rules, relational regression trees or relational association rules.

For multi-relational databases in which data instances are clearly identifiable (the so-called individual-centered representation (Flach & Lachiche, 1999), characterized by one-to-many relationships among the target table and other data tables), various techniques can be used for transforming a multi-relational database into a propositional single-table format (Krogel et al., 2003). After performing such a transformation (Lavrač, Džeroski, & Grobelnik, 1991), named *propositionalization* (Kramer, Pfahringer, & Helma, 1998), standard propositional learners can be used, including decision tree and classification rule learners.

Inspired by text mining, this paper presents a propositionalization approach to Relational Data Mining, called *wordification*. Unlike other propositionalization techniques (Kramer et al., 1998; Kuželka & Železný, 2011; Lavrač et al., 1991; Železný & Lavrač, 2006), which first construct complex relational features (constructed as a chain of joins of one or more tables related to the target table), used as attributes in the resulting tabular data representation, wordification generates much simpler features with the aim of achieving greater scalability.

Wordification can be viewed as a transformation of a relational database into a set of feature vectors, where each original instance is transformed into a-kind-of 'document' represented as a Bag-Of-Words (BOW) vector of weights of simple features, which can be interpreted as 'words' in the transformed BOW space. The 'words' constructed by wordification correspond to individual

* Corresponding author at: Jožef Stefan Institute, Ljubljana, Slovenia.
E-mail addresses: matic.perovsek@ijs.si (M. Perovšek), anze.vavpetic@ijs.si (A. Vavpetič), janez.kranjc@ijs.si (J. Kranjc), bojan.cestnik@temida.si (B. Cestnik), nada.lavrac@ijs.si (N. Lavrač).

http://dx.doi.org/10.1016/j.eswa.2015.04.017
0957-4174/

attribute–values of the target table and of the related tables, subsequently weighted by their Term Frequency-Inverse Document Frequency (TF-IDF) value (Jones, 1972; Salton & Buckley, 1988) (requiring real-valued attributes to be discretized first). Alternatively, instead of TF-IDF, simpler schemes can be used such as term frequency (TF) 'word' count, or the binary scheme indicating just the presence/absence of a 'word' in the 'document'.

To intuitively phrase the main idea of wordification, take two simple examples illustrating the wordification data preprocessing step in class-labeled data, where each structured data instance is transformed into a tuple of simple features, which are counts/weights of individual attribute–value pairs. Take the well-known relational domain of East–West Trains (Michie, Muggleton, Page, & Srinivasan, 1994) with cars containing different loads: one of the train's features in the BOW representation is the count/weight of rectangular loads it carries, no matter in which cars these loads are stored. Or in the standard Mutagenesis domain (Srinivasan, Muggleton, King, & Sternberg, 1994), a molecule may prove to be toxic if it contains a lot of atoms characterized by the property *atom_type = lead*, no matter how these atoms are bonded in the molecule. The main hypothesis of the wordification approach is that the use of this simple representation bias is suitable for achieving good results in classification tasks. Moreover, when using a binary scheme, this representation bias allows for simple and very intuitive interpretation in descriptive induction tasks, such as association rule learning from unlabeled multi-relational data.

Wordification suffers from some loss of information, compared to propositionalization methods which construct complex first-order features (which get values *true* or *false* for a given individual) as a chain of joins of one or more tables related to the target table. Nevertheless, despite some information loss, wordification has numerous advantages. Due to the simplicity of features, the generated hypotheses are easily interpretable by domain experts. The feature construction step in wordification is very efficient, therefore it can scale well for large relational databases. As wordification constructs each 'document' independently from the other 'documents', a large main table can be divided into smaller batches of examples, which can be propositionalized in parallel. Next, wordification can use TF or TF-IDF word weighting to capture the importance of a given feature (attribute value) of a relation in an aggregate manner, while feature dependence is modeled by constructing a-kind-of word '*n*-grams' as conjuncts of a predefined number of simple features. Finally, the wordification approach has the advantage of using techniques developed in the text mining community, such as efficient document clustering or word cloud visualization, which can now be effectively exploited in multi-relational data mining.

This paper shows that the developed wordification technique is simple, considerably more efficient and at least as accurate as the comparable state-of-the-art propositionalization methods. This paper extends our previous research (Perovšek, Vavpetič, & Lavrač, 2012, 2013) in many ways. The related work is more extensively covered. The improvements to the methodology include feature filtering by frequency, performance optimization (indexing by value), new options regarding feature weighting (next to TF-IDF, we added TF and binary), and a parallel version of the algorithm. The methodology description is now more detailed, including the formal wordification framework, the wordification algorithm pseudo code as well as time and space complexity analysis. The experimental evaluation has been substantially extended to include a comparison of three different term weighting schemes, additional propositionalization algorithms RelF (Kuželka & Železný, 2011) and Aleph (Srinivasan, 2007), as well as an additional classifier (SVM), which were applied to an extended set of experimental relational datasets. Such exhaustive experimentation has enabled us to statistically validate the experimental results by using the Friedman test and the Nemenyi post hoc test on the seven benchmark problems from the five relational domains (two of which have two database variants): IMDB,[1] Carcinogenesis (Srinivasan, King, Muggleton, & Sternberg, 1997), Financial[2] and two variants of Trains (Michie et al., 1994) and Mutagenesis (Srinivasan et al., 1994). Further experiments were done to analyze the effects of feature weighting, pruning and *n*-gram construction. In addition to the two experimental workflows developed in the web-based data mining platform ClowdFlows (Kranjc, Podpečan, & Lavrač, 2012), one workflow developed for learning and another for results evaluation and visualization, this paper introduces another wordification workflow applicable in association rule learning tasks from binarized features. The implemented workflows, which are available online through ClowdFlows, allow for methodology reuse and experiment repeatability. As a side-product of workflow development, the competing propositionalization algorithms used in experimental comparisons are also made available through ClowdFlows and can therefore be easily reused in combination with numerous pre-existing ClowdFlows components for data discretization, learning, visualization and evaluation, including a large number of Weka (Witten et al., 2011) and Orange (Demšar, Zupan, Leban, & Curk, 2004) components. Making selected RDM algorithms handy to use in real-life data analytics may therefore contribute to improved accessibility and popularity of Relational Data Mining.

The paper is organized as follows. Section 2 describes the background and the related work. Section 3 gives an informal overview of the wordification methodology, while Section 4 presents the formalism and the details of the developed wordification algorithm. The implementation of the methodology as a workflow in the ClowdFlows platform is described in Section 5. Section 6 presents the evaluation methodology implementation and the experimental results. Section 7 illustrates the utility of wordification in a descriptive induction setting of learning association rules from two real-life domains, using data from a subset of the IMDB movies database and from a database of traffic accidents. Section 8 concludes the paper by presenting the plans for further work.

2. Background and related work

Inductive Logic Programming (ILP) and Relational Data Mining (RDM) algorithms are characterized by the ability to use background knowledge in learning relational models or patterns (Džeroski & Lavrač, 2001; De Raedt, 2008; Lavrač & Džeroski, 1994; Muggleton, 1992), as by taking into account additional relations among the data objects the performance of data mining algorithms can be significantly improved.

Propositionalization (Kramer et al., 1998; Lavrač et al., 1991) is an approach to ILP and RDM, which offers a way to transform a relational database into a propositional single-table format. In contrast to methods that directly induce relational patterns or models, such as Aleph (Srinivasan, 2007) and Progol (Muggleton, 1995), propositionalization algorithms transform a relational problem into a form which can be solved by standard machine learning or data mining algorithms. Consequently, learning with propositionalization techniques is divided into two self-contained phases: (1) relational data transformation into a single-table data format and (2) selecting and applying a propositional learner on the transformed data table. As an advantage, propositionalization is not limited to specific data mining tasks such as classification, which is usually the case with ILP and RDM methods that directly induce models from relational data.

[1] http://www.webstepbook.com/supplements/databases/imdb.sql.
[2] http://lisp.vse.cz/pkdd99/Challenge/berka.htm.

The transformation to a single-table format can be achieved for the so-called individual-centered relational databases (Flach & Lachiche, 1999), i.e., databases that have a clear notion of an individual. The East–West Trains challenge (Michie et al., 1994), where the task is to classify the Trains as East-bound or West-bound, is a well-known domain in which individuals are clearly identified: each train is a single individual related with one or more cars that have different characteristics.

Most of the related work involves propositionalization through first-order feature construction (Kramer et al., 1998; Kuželka & Železný, 2011; Lavrač et al., 1991; Železný & Lavrač, 2006), where the algorithms construct complex first-order features, which then act as binary attributes in the new propositional representation of examples. One of the first propositionalization algorithms, LINUS (Lavrač et al., 1991), generates features that do not allow recursion and newly introduced variables. An improvement of LINUS, named SINUS (Lavrač & Flach, 2001), incorporates more advanced feature construction techniques inspired by feature construction implemented in 1BC (Flach & Lachiche, 1999). RSD (Železný & Lavrač, 2006) is a relational subgroup discovery algorithm composed of two main steps: the propositionalization step and the subgroup discovery step, where the output of the propositionalization step can be used also as input to other propositional learners. RSD effectively produces an exhaustive list of first-order features that comply with the user-defined mode constraints, similar to those of Progol (Muggleton, 1995) and Aleph (Srinivasan, 2007). Furthermore, RSD features satisfy the connectivity requirement, which imposes that no constructed feature can be decomposed into a conjunction of two or more features.

RELAGGS (Krogel & Wrobel, 2001), which stands for *rel*ational *agg*regation, is a propositionalization approach, which uses the input relational database schema as a basis for a declarative bias and aims to use optimization techniques usually used in relational databases (e.g., indexes). Furthermore, the approach employs aggregation functions in order to summarize non-target relations with respect to the individuals in the target table.

An early experimental comparison of propositionalization techniques was reported in Krogel et al. (2003), where RSD, SINUS and RELAGGS algorithms were compared.

Other means of propositionalization include stochastic propositionalization (Kramer et al., 1998), which employs a search strategy similar to random mutation hill-climbing: the algorithm iterates over generations of individuals, which are added and removed with a probability proportional to the fitness of individuals, where the fitness function used is based on the Minimum Description Length (MDL) principle.

Safarii[3] is a commercial multi-relation data mining tool. Safarii, extensively described in Knobbe (2005), offers a unique pattern language that merges ILP-style structural descriptions as well as aggregations. Furthermore, Safarii comes with a tool called ProSafarii, which offers several preprocessing utilities, including propositionalization via aggregation.

Ceci and Appice (2006) investigate spatial classification using two techniques: a propositionalization approach which constructs features using spatial association rules to produce an attribute–value representation. They compare the approach to a structural approach using an extended Naive Bayes classifier. They report an advantage of the structural alternative in terms of accuracy, while the propositional approach performs faster. Ceci, Appice, and Malerba (2008) present two emerging patterns based classifiers that work in the multi-relational setting: one uses a heuristic evaluation function to classify objects, while the other is based on a probabilistic evaluation. The main result of the study is that both approaches perform better than associative classification to which they were compared.

Kuželka and Železný (2011) developed RelF, which constructs a set of tree-like relational features by combining smaller conjunctive blocks. The novelty is that RelF preserves the monotonicity of feature reducibility and redundancy (instead of the typical monotonicity of frequency), which allows the algorithm to scale far better than other state-of-the-art propositionalization algorithms.

An approach that is related to propositionalization is presented by Guo and Viktor (2008). The authors propose a strategy of multi-relational learning where they neither upgrade a propositional learner to work with multiple relations or propositionalize the relations. Instead, their approach learns from multiple views (feature sets) of a RDB and then integrates the individual view learners to construct a final model. Their approach exhibits comparable classification accuracies compared to related approaches, and a faster runtime.

Recently, a propositionalization technique called Bottom Clause Propositionalization (BCP) was introduced by França, Zaverucha, and d'Avila Garcez (2014). It was integrated with C-IL^2P (Garcez et al., 1999); the combined system, named CILP++, achieves accuracy comparable to Aleph, while being faster. Compared to RSD, BCP is better in terms of accuracy when using a neural network and similar when using C4.5.

3. Informal description of the wordification approach

This section provides an informal description of the proposed approach, where wordification is illustrated on a simplified variant of the well-known East–West Trains problem (Michie et al., 1994).

The transformation from a relational database representation into a Bag-Of-Words feature vector representation is illustrated in Fig. 1, where the input to wordification is a relational database, and the output is a set of feature vectors, which can be viewed as a corpus of text documents represented in the Bag-Of-Words (BOW) vector format. Each text document represents an individual entry of the main data table. A document is described by a set of words (or features), where a word is constructed as a combination of the table name, name of the attribute and its discrete (or discretized) value[4]:

$$[table\ name]\_[attribute\ name]\_[value]. \quad (1)$$

Such constructs are called *word-items* or *witems* or simply *words* in the rest of the paper. Note that values of every non-discrete attribute need to be discretized beforehand in order to be able to represent them as word-items. For each individual, the word-items are first generated for the main table and then for each entry from the related tables, and finally joined together according to the relational schema of the database.[5] In the described transformation there is some loss of information as a consequence of building the document for each instance (each individual row in the main table) by concatenating all word-items from multiple instances (rows) of the connected tables into a single document. To overcome this loss, we extended the document construction step of the initial wordification methodology by concatenating to the document also *n*-grams of word-items, constructed as conjunctions of several word-items. These concatenations of elementary word-items represent conjunctions of features occurring together in individual instances (rows of joined tables). Technically, *n*-gram construction is performed by taking every combination of length *k* of word-items from the set of all

[3] http://www.kiminkii.com/safarii.html.

[4] See Line 4 in Algorithm 2 presented in Section 4.3.

[5] See Line 10 in Algorithm 2 presented in Section 4.3.

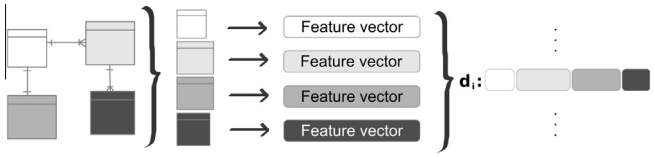

Fig. 1. The transformation from a relational database representation into a Bag-Of-Words feature vector representation. For each individual entry of the main table one Bag-Of-Words (BOW) vector d_i of weights of 'words' is constructed, where 'words' correspond to the features (attribute values) of the main table and the related tables.

word-items corresponding to the given individual, and concatenating them as follows:

$$[witem_1]\_\_[witem_2]\_\_\ldots\_\_[witem_k], \tag{2}$$

where $1 \leqslant k \leqslant n$ and—as mentioned earlier—each word-item is a combination of the table name, name of the attribute and its discrete value. The witems are concatenated in a predetermined order, each using the "_" concatenation symbol.

In the rest of this section, for simplicity, we refer to individuals as documents, to features as words, and to the resulting representation as the Bag-Of-Words (BOW) representation. For a given word w in document d from corpus D, the TF-IDF measure is defined as follows:

$$\text{tfidf}(w,d) = \text{tf}(w,d) \times \log \frac{|D|}{|\{d \in D : w \in d\}|}, \tag{3}$$

where $\text{tf}(\cdot)$ represents the number of times word w appears in document d. In other words, a word with a high TF-IDF value will be considered important for the given individual provided that it is frequent within this document and not frequent in the entire corpus. Consequently, the weight of a word provides a strong indication of how relevant is the feature for the given individual. The TF-IDF weights can then be used either for filtering out words with low importance or using them directly by a propositional learner.

In addition to the TF-IDF weigthing scheme, the implementation of wordification (described in detail in Section 5) includes also the term frequency (TF) and the binary (0/1) weighting schemes. A comparison of the three schemes can be found in the Appendix A (see Table A.6). Given that different weighting schemes do not perform significantly differently on the classification tasks used in our experiments, in the rest of the paper we use the TF-IDF scheme since this form of weighting is prevalent in text mining applications.

The wordification approach is illustrated on a modified and substantially simplified version of the well-known East–West Trains domain (Michie et al., 1994), where the input database consists of just two tables shown in Fig. 2, where we have only one East-bound and one West-bound train, each with just two cars with certain properties. Note that in the experimental section we use the standard version of the East–West Trains domain.

The TRAIN table is the main table and the Trains are the individuals. We want to learn a classifier to determine the direction of an unseen train. For this purpose the direction attribute is not preprocessed and is only appended to the resulting feature vector (list of words).

First, the corresponding two documents (one for each train t1 and t5) are generated, as shown in Fig. 3. After this, the documents are transformed into the Bag-Of-Words representation by calculating the TF-IDF values for each word of each document (using Eq. (3)) with the class attribute column appended to the transformed Bag-Of-Words table, as shown in Fig. 4. For simplicity, only unigrams and bigrams are shown in this example.

TRAIN

trainID	eastbound
t1	east
…	…
t5	west
…	…

CAR

carID	shape	roof	wheels	train
c11	rectangle	none	2	t1
c12	rectangle	peaked	3	t1
…	…	…	…	…
c51	rectangle	none	2	t5
c52	hexagon	flat	2	t5
…	…	…	…	…

Fig. 2. Example input for wordification in the East–West Trains domain.

4. Wordification methodology

This section formally describes the wordification methodology by presenting the input data model and input language bias, the relational database representation, followed by the presentation of the pseudo-code and the worst-case complexity analysis of the wordification algorithm.

4.1. Data model

A data model describes the structure of the data. It can be expressed as an entity-relationship (ER) diagram. The ER diagram, illustrated in Fig. 5, shows three relations appearing in the original East–West Trains problem (in addition to the TRAIN and CAR relationship, it includes also the LOAD relationship, which was skipped for simplicity in Fig. 2, which contains just the TRAIN and CAR relational tables). The boxes in the ER diagram indicate *entities*, which are individuals or parts of individuals. Here, the Train entity is the individual, each Car is part of a train, and each Load is part of a car. The ovals denote attributes of entities. The diamonds indicate *relationships* between entities. There is a *one-to-many relationship* from Train to Car, indicating that each train can have an arbitrary number of cars but each car is contained in exactly one train; and a *one-to-one relationship* between Car and Load, indicating that each car has exactly one load and each load is part of exactly one car.

```
t1: [car_roof_none, car_shape_rectangle, car_wheels_2,
    car_roof_none__car_shape_rectangle, car_roof_none__car_wheels_2,
    car_shape_rectangle__car_wheels_2, car_roof_peaked, car_shape_rectangle,
    car_wheels_3, car_roof_peaked__car_shape_rectangle,
    car_roof_peaked__car_wheels_3, car_shape_rectangle__car_wheels_3], east
...
t5: [car_roof_none, car_shape_rectangle, car_wheels_2,
    car_roof_none__car_shape_rectangle, car_roof_none__car_wheels_2,
    car_shape_rectangle__car_wheels_2, car_roof_flat, car_shape_hexagon,
    car_wheels_2, car_roof_flat__car_shape_hexagon,
    car_roof_flat__car_wheels_2, car_shape_hexagon__car_wheels_2], west
...
```

Fig. 3. The database from Fig. 2 in the Bag-Of-Words document representation.

	car_shape _rectangle	car_roof _peaked	car_wheels_3	car_roof_peaked__ car_shape_rectangle	car_shape_rectangle __car_wheels_3	...	class
t1	0.000	0.693	0.693	0.693	0.693	...	east
...	...	...	...	...	...	...	...
t5	0.000	0.000	0.000	0.000	0.000	...	west
...	...	...	...	...	...	...	...

Fig. 4. The transformed database (consisting of TF-IDF values, which are zero if the term appears in all documents) from Fig. 2 using the wordification approach. This final output can be given as an input to a propositional classifier.

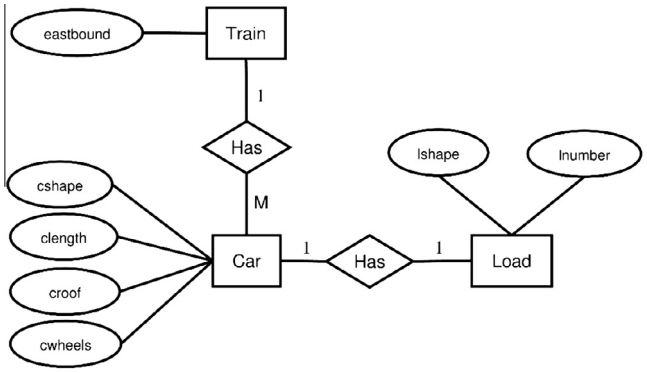

Fig. 5. Entity-relationship diagram for the East–West challenge.

Entity-relationship diagrams can be used to choose a proper logical representation for the data. If we store the data in a relational database the most obvious representation is to have a separate table for each entity in the domain, with relationships being expressed by *foreign keys*.[6] This is not the only possibility: for instance, since the relationship between Car and Load is one-to-one, both entities could be combined in a single table, while entities linked by a one-to-many relationship cannot be combined without either introducing significant redundancy or significant loss of information, e.g., introduced through aggregate attributes. Note that one-to-many relationships distinguish relational learning and Inductive Logic Programming from propositional learning.

In wordification, we use the entity-relationship diagram to define types of objects in the domain, where each entity will correspond to a distinct type. The data model constitutes a *language bias* that can be used to restrict the hypothesis space and guide the search. In most problems, only individuals and their parts exist as entities, which means that the entity-relationship model has a tree-structure with the individual entity at the root and only *one-to-one* or *one-to-many* relations in the downward direction. Representations with this restriction are called *individual-centered representations*. This restriction determines the language bias, constraining the relational database input to wordification.

4.2. Formal setting

The framework, established in this section, defines a learning setting which is very similar to the standard propositionalization problem setting. As in every propositionalization approach to Relational Data Mining, a two-step approach is implemented: (1) in the first propositionalization step the data is transformed from a relational database format to a tabular format, and (2) the tabular data is used as input for learning models or patterns by a selected propositional learner, having its own hypothesis language bias (e.g., decision trees or propositional classification rules). The formal framework described below focuses only on step (1) of the two-step wordification methodology. For simplicity, the formalization describes the setting using only unigram features.

Input. The input to wordification is a *relational database* (RDB), given as a set of relations $\{R_1, \ldots, R_n\}$ and a set of foreign-key connections between the relations denoted by $R_i \rightarrow R_j$, where R_i has a foreign-key pointing to relation R_j. The foreign-key connections correspond to the relationships in the entity-relationship diagram. For example, the `train` attribute in the CAR relation is a foreign-key referring to `trainID` in TRAIN. It defines the `CAR` $\rightarrow$ `TRAIN` connection; as expected, it is a many-to-one connection from CAR to TRAIN.

A n-ary relation R_i is formally defined as a set of *tuples*: a subset of the Cartesian product of m_i domains: $R_i \subset \prod_{j=1}^{m_i} D_{i_j} = D_{i_1} \times D_{i_2} \times \ldots \times D_{i_{m_i}}$, where a *domain* (or a *type*) is a specification of the valid set of values for the corresponding argument.

$$D_{i_j} = \{v_{i_{j_1}}, v_{i_{j_2}}, \ldots, v_{i_{j_{k_{ij}}}}\}$$

Note that for wordification we require that each domain D_{i_j} must have a finite number of unique values k_{ij}, thus discretization of continuous domains is needed.

A further requirement is that the RDB must be individual-centered. This means that a target relation $R_T \in$ RDB must exist, such that it does not have any foreign keys:

$$\nexists i : R_T \rightarrow R_i; \quad R_i \in \text{RDB}$$

Output. Having established the data model, the individual-centered data representation language bias and the relational database representation of input data, the formal output (a transformed single-relation representation $R_{T'}$) of the wordification methodology can be defined as follows:

$$R_{T'} \subset \prod_{i,j,k} D_{T'_{ijk}} = \prod_{i,j,k} \text{domain}(R_i, D_{i_j}, v_{i_{j_k}}); R_i \overset{*}{\rightarrow} R_T$$

or in other words, one domain in the resulting relation $R_{T'}$ is defined for each relation R_i (that is connected by following the foreign-key path, denoted by $\overset{*}{\rightarrow}$ to R_T), and each of its domains D_{i_j} as well as domain values $v_{i_{j_k}}$. These domains have the property

$$D_{T'_{ijk}} = \mathbb{R}_0^+$$

since they are determined by the TF-IDF formula. This final output relation (table) can be given as an input to any propositional learner.

4.3. Wordification algorithm

This section presents the wordification methodology by describing in detail the individual transformation steps in Algorithms 1 and 2.

[6] In the context of relational databases, a foreign key is a field in a relational table that matches a candidate key of another table. The foreign key can be used to cross-reference tables.

Algorithm 1. Wordification(T, p, k)

```
Input  : target table T, pruning percentage p, maximal number of witems per word k
Output: Propositionalized table R with TF-IDF values, corpus of documents D
1 D ← [];
2 W ← ∅ ;                                                    // vocabulary set
3 for ex ∈ T do
4 |   d ←wordify(T, ex, k) ;                                 // construct the document
5 |   D ← D + [d] ;                                          // append document to the corpus
6 |   W ← W ∪ keys(d) ;
7 end
8 W ←prune(W, p) ;                                           // optional step
9 return [ calculateTFIDFs(D, W), D];
```

Algorithm 2. Wordify(T, ex, k)

```
Input  : table T, example ex from table T, maximal number of witems per word k
Output: document word count d
1 d ← {};                                                    // hash with a default value 0
2 for i ← 1 to k do                                          // for every word witem length
3 |   for comb ∈ attrCombs(ex, k) do                         // attr. combinations of length k
4 |   |   d[word(comb)] ← d[word(comb)] + 1;
5 |   end
6 end
7 // for every connected table through an example
8 for secTable ∈ connectedTables(T) do
9 |   for secEx ∈ secTable do
10|   |   if primaryKeyValue(ex)=foreignKeyValue(secEx) then
11|   |   |   for (word, count) ∈ wordify(secTable, secEx, k) do
12|   |   |   |   d[word] ← d[word] + count;
13|   |   |   end
14|   |   end
15|   end
16 end
17 return d;
```

The algorithm starts recursive document construction on the instances of the main table (Lines 3–7 in Algorithm 1). First it creates word-items for the attributes of the target table (Lines 2–6 in Algorithm 2), which is followed by concatenations of the word-items and results of the recursive search through examples of the connecting tables (Lines 8–16 in Algorithm 2). As this document construction step is done independently for each example of the main table, this allows simultaneous search along the tree of connected tables. In order to perform concurrent propositionalization, Lines 3–7 in Algorithm 1 need to be run in parallel. A common obstacle in parallel computing is memory synchronization between the different subtasks, which is not the case here as concurrent processes in our implementation of wordification only need to share a cached list of subtrees. This list stores the results of subtree word concatenations in order to visit every subtree only once.

As wordification can produce a large number of features (words), especially when the maximal number of n-grams per word-items is large, we perform pruning of words that occur in less than a predefined percentage (5% on default) of documents. This reduces the size of trees by removing sections of the tree that is expected to provide little power for instance classification.

The constructed features are simple, and as we do not explicitly use existential variables in the new features (words), we instead rely on the Term Frequency-Inverse Document Frequency (TF-IDF) measure to implicitly capture the importance of a word for a given individual. In the context of text mining, TF-IDF value reflects how representative is a certain feature (word) for a given individual (document).

4.4. Time and space complexity

This section covers the worst-case complexity analysis of the wordification algorithm. Let t be the number of tables in a database. To simplify the analysis, we assume that each table is connected with exactly one other table in a one-to-many relation. Let m_i and n_i be the number of rows and the number of attributes

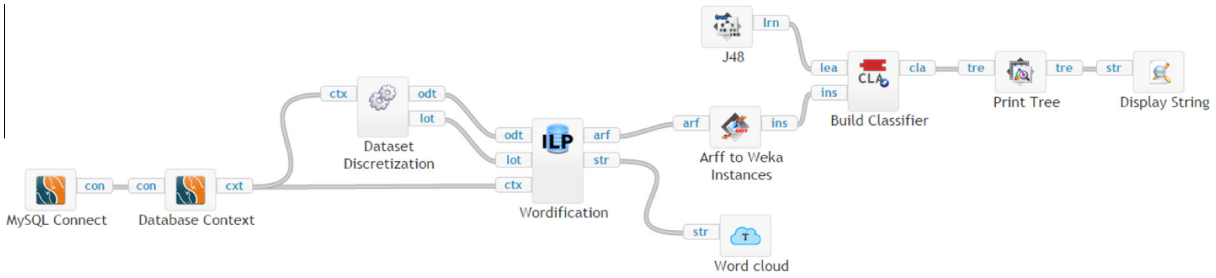

Fig. 6. Clowdflows wordification workflow with additional analyses after the wordification process. This workflow is publicly available at http://clowdflows.org/workflow/1455/. The abbreviations on the input and output stubs (which are not important for understanding the workflow) are as follows: *con* connection, *ctx* context, *odt* Orange data table, *lot* list of Orange data tables, *str* string, *arf* ARFF file, *ins* instances, *lrn* learner, *cla* classifier.

in table i, respectively. Further, let $m = \max(m_1, m_2, \ldots, m_t)$ and $n = \max(n_1, n_2, \ldots, n_t)$. The maximal number of rows is generally much higher than the number of attributes and the number of tables in a relational database: $t \ll n \ll m$. Let k be the maximal number of n-grams per word.

Time complexity. The upper-bound time complexity for the propositionalization step of the wordification methodology when each word is constructed from one witem ($k = 1$) is $\mathcal{O}(m \cdot n \cdot m^{t-1}) = \mathcal{O}(n \cdot m^t)$. When we use words that are combinations of up to $1 \leqslant k \leqslant n$ witems (Lines 3–5 in Algorithm 2) the complexity of the algorithm is $\mathcal{O}\left(m^t \cdot \sum_{i=1}^{k} \binom{n}{i}\right)$. As $\lim_{k \to n} \sum_{i=1}^{k} \binom{n}{i} = 2^n - 1$, the worst-case time complexity is therefore $\mathcal{O}(2^n \cdot m^t)$.

Space complexity. The space complexity for the wordification algorithm using unigram features ($k = 1$) is $\mathcal{O}(m \cdot t \cdot n)$. When we increase the maximal word length (number of witems per word) the feature space of the algorithm also increases exponentially. When the maximal word length is equal to the maximal number of attributes, the space complexity is $\mathcal{O}\left(m \cdot t \cdot \sum_{i=1}^{k} \binom{n}{i}\right)$. Following a similar reasoning as in the time complexity analysis, the worst case space complexity of the algorithm is therefore $\mathcal{O}(m \cdot t \cdot 2^n)$.

When $k \to n$ both time and space complexity are in its worst cases exponential in the number of attributes, but as evidenced from the experiments in Section 6, good performance can be achieved with $k = 1$ or $k = 2$, in which case the space complexity is linear $\mathcal{O}(m \cdot t \cdot n)$ and the time complexity is polynomial $\mathcal{O}(n \cdot m^t)$. Since t is usually small, the approach can perform orders of magnitude faster than its competitors, as demonstrated in Section 6.

5. Implementation

This section describes the implementation of the wordification methodology in the ClowdFlows platform. We briefly present the platform with its distinguishing features including the ILP module, followed by a description of the wordification workflow and its components.

5.1. The ClowdFlows platform

The ClowdFlows platform (Kranjc et al., 2012) is an open-source, web-based data mining platform that supports the construction and execution of scientific workflows. ClowdFlows differs from comparable platforms by its web based architecture. During run-time the ClowdFlows platform resides on a server (or on a cluster of machines) while its graphical user interface that allows workflow construction is served as a web application accessible from any modern web browser. ClowdFlows is essentially a cloud-based web application that can be accessed and controlled from anywhere while the processing is performed in a cloud of computing nodes.[7]

The ClowdFlows platform is written in Python using the Django framework. Its graphical user interface is implemented in JavaScript and features simple operations that allow workflow construction: adding workflow components (widgets) on a canvas and creating connections between the components to form executable workflows.

The platform has the following distinguishing features: an implementation of a visual programming paradigm, a web-service consumption module, a real-time processing module for mining data streams, the ability to easily share and publicize workflows constructed in ClowdFlows, and an ever growing roster of reusable workflow components and entire workflows.

Following a modular design, workflow components in ClowdFlows are organized into packages which allows for easier distributed development. ClowdFlows is easily extensible by adding new packages and workflow components by writing simple or complex Python functions. Workflow components of several types also allow graphical user interaction during runtime, visualization of results by implementing views in JavaScript, HTML or any other format that can be displayed in a web browser (e.g., Flash, Java Applet). The ClowdFlows packages include Weka algorithms (Witten et al., 2011), Orange algorithms (Demšar et al., 2004), text mining, as well as different ILP and RDM algorithms.

The current ILP module includes components, such as the popular ILP system Aleph (Srinivasan, 2007), as well as RSD (Železný & Lavrač, 2006), SDM (Vavpetič & Lavrač, 2013) and RelF (Kuželka & Železný, 2011). Aleph is an ILP toolkit on its own, with a wide range of functionalities: from decision tree learning to feature generation and first-order rule induction. We have extended the ClowdFlows ILP module with the implementation of the *wordification* component, which together with the existing components from different modules of the ClowdFlows platform forms the entire workflow of the wordification methodology, as can be seen in Fig. 6. Along with other components in the ILP module, wordification components can be used to construct diverse RDM workflows. As completed workflows, data, and results can also be made public by the author of the workflow, the platform can serve as an easy-to-access integration platform for various RDM workflows. Each public workflow is assigned a unique URL that can be accessed by anyone to either repeat the experiment, or use the workflow as a template to design another workflow.

5.2. Wordification workflow

This section describes the main components of the wordification workflow, which is shown in Fig. 6. The implementation

[7] A public installation of ClowdFlows is accessible at http://clowdflows.org.

Table 1
Table properties of the experimental data.

	# Rows	# Attributes
Trains		
Cars	63	9 (10)
Trains	20	2
Carcinogenesis		
Atom	9064	5
canc	329	2
sbond_1	13,562	4
sbond_2	926	4
sbond_3	12	4
sbond_7	4134	4
Mutagenesis 42		
Atoms	1001	5
Bonds	1066	5
Drugs	42	7
Ring_atom	1785	3
Ring_strucs	279	3
Rings	259	2
Mutagenesis 188		
Atoms	4893	5
Bonds	5243	5
Drugs	188	7
Ring_atom	9330	3
Ring_strucs	1433	3
Rings	1317	2
IMDB		
Actors	7118	4
Directors	130	3
Directors_genres	1123	4
Movies	166	4
Movies_directors	180	3
Movies_genres	408	3
Roles	7738	4
Financial		
Accounts	4500	4
Cards	892	4
Clients	5369	4
Disps	5369	4
Districts	77	16
Loans	682	7
Orders	6471	6
tkeys	234	2
Trans	1,056,320	10

allows the user to provide as input a relational database by connecting to a MySQL database server. First, the user is required to select the target table from the initial relational database, which will later represent the main table in the wordification component of the workflow. Second, the user is able to discretize each table using one of the various discretization techniques provided. These discretized tables are used by the wordification widget, where the transformation from the relational tables to a corpus of documents is performed. The workflow components are described in more detail below.

MySQL Connect. Since relational data is often stored in SQL databases, we use the MySQL package to access the training data by connecting to a MySQL database server. The MySQL Connect widget is used for entering information required to connect to a database (e.g., user credentials, database address, database name, etc.) in order to retrieve the training data from a MySQL database server and automatically construct the background knowledge and the training examples.

Database Context. This widget enables a selection of tables and columns that will be used in the next steps of the methodology. The information is carried to the connected widgets through the so-called database context objects. These objects also contain the detected table relationships. In case that the input relational database does not have predefined primary and foreign keys between the tables, the user is given an option for simple table connection search through the names of the attributes.

Table 2
Majority classifier performance for each dataset.

Domain	CA [%]
Trains	50.00
Mutagenesis 42	69.05
Mutagenesis 188	66.50
IMDB	73.49
Carcinogenesis	55.32
Financial	86.75

Dataset Discretization. The sole task of this widget is to convert continuous attributes to categorical, by discretizing the continuous attributes. Dataset Discretization widget supports three discretization methods: using equal-width intervals, using equal-frequency intervals, and class-aware discretization proposed by Fayyad and Irani (1993) that uses MDL and entropy to find the best cut-off points. Dataset Discretization widget can take as input either a single data set or a list of multiple datasets. In the latter case, discretization of all continuous attributes of every dataset is performed.

Wordification. The wordification widget transforms the relational database to a corpus of documents for the main table. As an input it takes three arguments: the target (main) table, a list of additional tables and a database context, which contains the relations between the tables. The widget first indexes the examples of every table by their primary and foreign keys' values. This step is required for performance optimization of data retrieval operations when searching for connecting instances from different (connected) tables in the word-item concatenation step. Next, recursive document construction for every individual is performed. The algorithm starts on every example of the main table: it creates word-items for its attributes, followed by concatenations of the word-items and results of the recursive search through the connecting tables. When searching along the tree of connected tables, the algorithm stores the results of subtree word concatenations for every instance. Consequently, the algorithm iterates over every subtree only once. The user can run the widget with different parameters: maximal number of n-grams per word, as well as the pruning threshold parameter. The wordification widget produces two outputs: a set of generated word documents and an *arff* table with calculated TF-IDF values for every example of the main table.

Word Cloud. A set of generated word documents can be displayed using word cloud visualization, enabling improved domain understanding.

Data Mining. After the wordification step the user can perform various types of analysis, depending on the task at hand (classification, clustering, etc). In the example workflow shown in Fig. 6, the *arff* output is used as input to build and display a J48 decision tree.

6. Experiments

This section presents the evaluation of the wordification methodology. After describing the relational databases used in this study, we describe the experiments performed on these datasets and provide a comparison of wordification to other propositionalization techniques. In comparison with the experimental setting described in Perovšek, Vavpetič, Cestnik, and Lavrač (2013), a larger number of datasets is used, and very favorable results are obtained by using decision tree learner J48, compared to relatively poor results reported in previous work, where the Naive Bayesian classifier assuming feature independence was used. In addition to the J48 tree learner, we also tested the LibSVM learner.

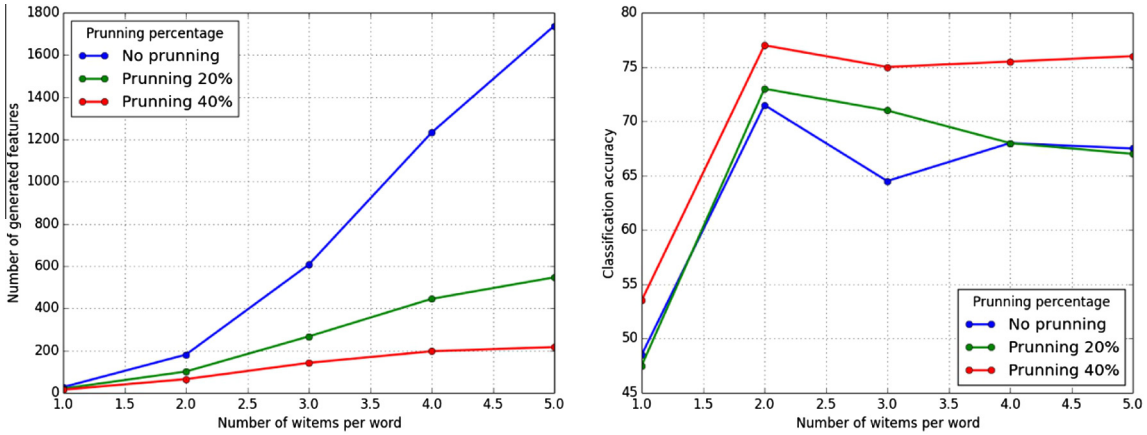

Fig. 7. Experiments on the wordification propositionalization step. Left: Size of the feature space in correlation with the number witems per word. Right: Classification accuracies in leave-one-out cross-validation (using the J48 decision tree learner with default parameter setting) as a function of the maximum number of *n*-grams per word.

Let us first present the five relational databases used in the experiments: Trains (in two variants), Carcinogenesis, Mutagenensis with 42 and 188 examples, IMDB, and Financial. Table 1 lists the characteristics of the datasets. All the datasets can be downloaded from a web page,[8] making them easily reusable in other experiments.

Trains. The well-known East–West Trains challenge is an ILP problem of predicting whether a train is East-bound or West-bound. A train contains a variable number of cars that have different shapes and carry different loads. We have considered two versions of the data for our experiments: the original dataset from the East–West Trains challenge and a modified dataset where every car also has its position as an additional attribute. In both datasets we have considered East-bound Trains as positive examples.

Carcinogenesis. The problem addressed by Srinivasan et al. (1997) is to predict carcinogenicity of a diverse set of chemical compounds. The dataset was obtained by testing different chemicals on rodents, where each trial would take several years and hundreds of animals. The dataset consists of 329 compounds, of which 182 are carcinogens.

Mutagenesis In this task the goal is to predict mutagenicity of aromatic and heteroaromatic nitro compounds (Debnath, Lopez de Compadre, Debnath, Shusterman, & Hansch, 1991). Predicting mutagenicity is an important task as it is very relevant to the prediction of carcinogenesis. The compounds from the data are known to be more structurally heterogeneous than in any other ILP dataset of chemical structures. The database contains 230 compounds of which 138 have positive levels of mutagenicity and are labeled as 'active'. Others have class value 'inactive' and are considered to be negative examples. We took the datasets of the original Debnath paper (Debnath et al., 1991), where the data was split into two subsets: a 188 compound dataset and a smaller dataset with 42 compounds.

IMDB The complete IMDB database is publicly available in the SQL format. This database contains tables of movies, actors, movie genres, directors, and director genres. The database used in our experiments consists only of the movies whose titles and years of production exist on the IMDB's top 250 and bottom 100 chart.[9] The database therefore consists of 166 movies, along with all of their actors, genres and directors. Movies present in the IMDB's top 250 chart were considered as positive examples, while those in the bottom 100 were regarded as negative.

Financial. This is a publicly available dataset, which was artificially constructed as part of the PKDD'99 Discovery Challenge. The classification task addressed is the prediction of successful loans. The dataset consists of 8 tables describing clients of a bank, their accounts, transactions, permanent orders, granted loans and issued credit cards.

Table 2 presents the performance of the majority classifier for each of the described datasets. This should serve as a baseline for the classification results reported in the following subsections.

6.1. Evaluation of feature construction and filtering

The experiments, enabling the analysis of the feature generation step of wordification were performed on the original East–West Trains challenge dataset using different parameter settings: using elementary word-items and complex word-items constructed from up to 5-grams of witems.

The left plot in Fig. 7 shows that the size of the feature space in the non-pruned version of wordification increases exponentially as the maximal number of witems per word increases. Note that wordification also implements a pruning technique where words that occur in less than a predefined percentage of documents are pruned. As shown in Fig. 7, using higher thresholds for feature filtering drastically reduces the dimensionality of the data, resulting in more efficient learning.

We have also applied different wordification settings in the classification task on the Trains dataset. The classification accuracies using the J48 decision tree of leave-one-out cross-validation for different parameters are shown on the right side of Fig. 7 (the reason for using leave-one-out instead of the standard 10-fold cross-validation setting is a very small number of instances in the Trains dataset). The results show that using larger *n*-grams of witems only marginally improves the classification accuracies, but results in longer run-times of the propositionalization step because of a larger feature space. In this specific domain, pruning performs favorably in terms of classification accuracy, though as the experiments in the Appendix A show (Table A.7), this observation is only applicable to small domains, while for larger domains with more potential witem combinations this observation does not hold.

6.2. Comparative evaluation of propositionalization techniques

This section describes the experiments performed in the evaluation of different propositionalization approaches on binary classification tasks, using the datasets from the five relational domains.

[8] http://kt.ijs.si/janez_kranjc/ilp_datasets/.
[9] As of July 2, 2012.

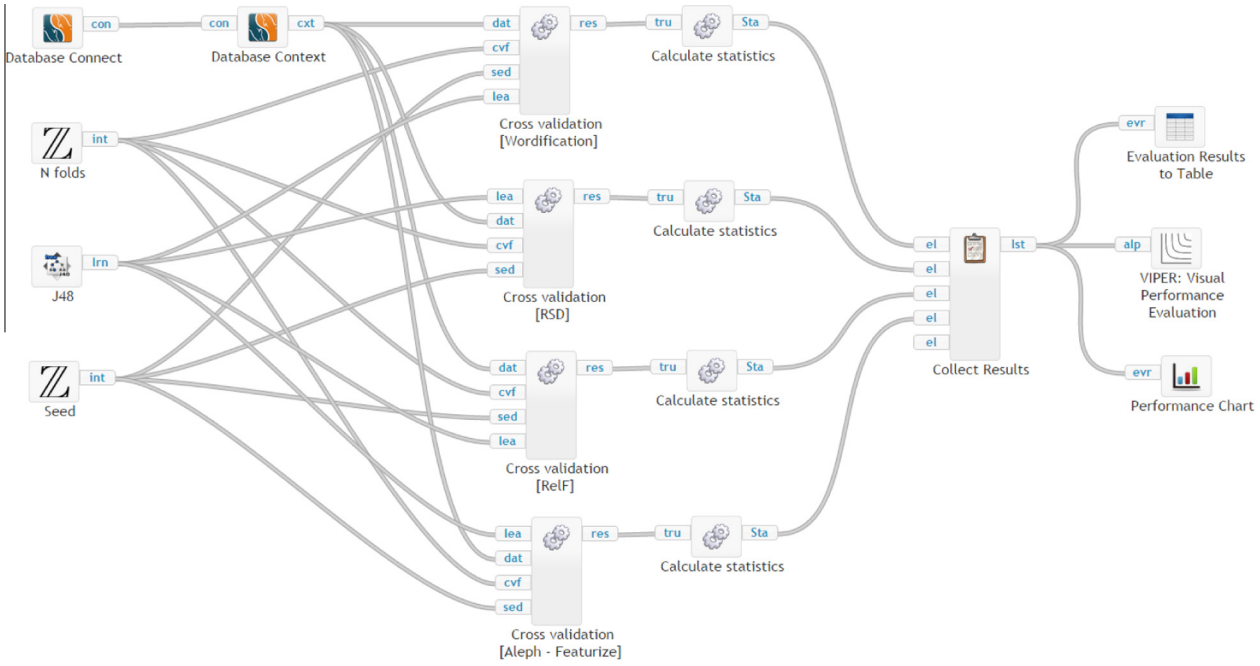

Fig. 8. Evaluation workflow for evaluating and comparing Wordification, Aleph, RSD, and RelF, implemented in the ClowdFlows data mining platform. The abbreviations on the input and output stubs are as follows: *con* connection, *ctx* context, *dat* full dataset for cross-validation, *cvf* number of cross-validation folds, *sed* random seed, *lrn/lea* learner instance, *res* cross-validation results, *sta* classification statistics, *evr* evaluation results.

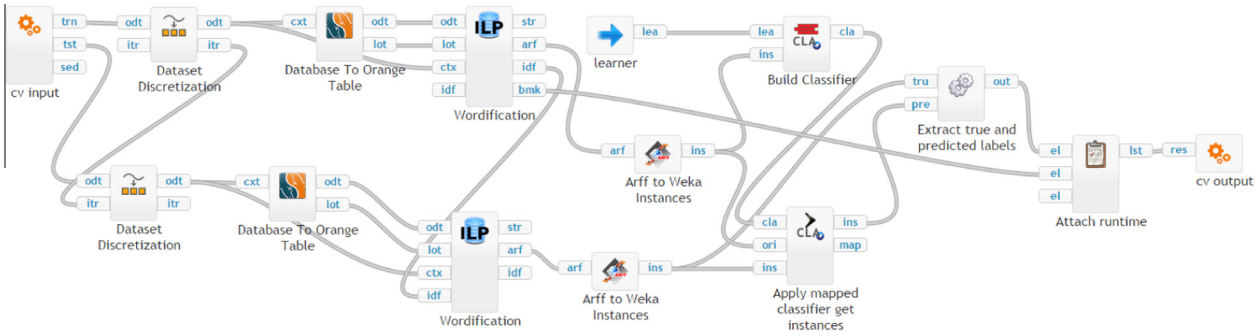

Fig. 9. The cross-validation subprocess from Fig. 8. This workflow gets a training set (input *trn*) and a test set (input *tst*) as input and is executed for each fold. There are two Wordification widgets in the workflows: one responsible for constructing the features on the Trains set and the other on the test set. The connection between the two widgets is needed for transferring the IDF weights learned on the training set, which are used for feature construction on the test set. The results of each step are collected by the *cv output* widget. Other widget input and output abbreviations are not important for understanding the workflow.

Fig. 8 shows the full experimental workflow (from connecting to a relational database management server to visualizing the experimental results and evaluation). This evaluation workflow is available online[10] in the ClowdFlows platform, which enables ILP researchers to reuse the developed workflow and its components in future experimentation.

The first step of the evaluation methodology is to read the relational data, stored in an SQL database, using the MySQL package widgets. Data then enters the cross-validation subprocess (Fig. 9), where the following steps are repeated for each fold (we used 10-fold cross-validation). First, discretization of the training fold of the relational database is performed. We have arbitrarily selected equi-distance discretization with 3 intervals of values to discretize the continuous attributes of the experimental relational datasets, such that none of the techniques was given an advantage. Then a propositionalization technique is applied to the training data and the results are formatted in a way to be used by the Weka algorithms. The J48 decision tree and the LibSVM learners were selected with their default parameter settings to perform binary classification.

The test set is handled as follows. First, the data is discretized to the intervals determined on the training set. Second, the features produced by the given propositionalization approach on the training set are evaluated on the test set to produce a propositional representation of the test data. Note that this process is slightly different for wordification, since the features do not have to be evaluated. We do however need the IDF values calculated on the training set. Finally, these test examples are classified by the classifiers trained on the training data. The results of each step are then collected to be returned at the end by the cross-validation subprocess.

Every propositionalization algorithm was run with its default settings. A non-parallel version of wordification was run using only the elementary words (maximal number of witems per word was set to 1) and without pruning, as none of our datasets required this.

[10] http://clowdflows.org/workflow/4018/.

Table 3
Classifier evaluation on different databases. The bolded items indicate the best results.

Domain	Algorithm	J48		LibSVM		Time [s]
		CA [%]	AUC	CA [%]	AUC	
Trains without position	Wordification	50.00	0.50	50.00	0.50	**0.11**
	RelF	**65.00**	**0.65**	**80.00**	**0.80**	1.04
	RSD	**65.00**	**0.65**	75.00	0.75	0.53
	Aleph – Featurize	60.00	0.60	65.00	0.65	0.40
Trains	Wordification	**95.00**	**0.95**	50.00	0.50	**0.12**
	RelF	75.00	0.75	75.00	0.75	1.06
	RSD	60.00	0.60	**80.00**	**0.80**	0.47
	Aleph – Featurize	55.00	0.55	70.00	0.70	0.38
Mutagenesis 42	Wordification	**97.62**	**0.96**	**78.57**	**0.65**	**0.39**
	RelF	76.19	0.68	76.19	0.62	2.11
	RSD	**97.62**	**0.96**	69.05	0.50	2.63
	Aleph – Featurize	69.05	0.50	69.05	0.50	2.07
Mutagenesis 188	Wordification	68.62	0.55	**81.91**	**0.78**	**1.65**
	RelF	**75.00**	**0.68**	68.62	0.54	7.76
	RSD	68.09	0.54	71.28	0.58	10.10
	Aleph – Featurize	60.11	**0.68**	60.11	0.68	19.27
IMDB	Wordification	**81.93**	**0.75**	73.49	0.50	**1.23**
	RelF	69.88	0.66	73.49	0.50	32.49
	RSD	74.70	0.59	73.49	0.50	4.33
	Aleph – Featurize	73.49	0.50	73.49	0.50	4.96
Carcinogenesis	Wordification	**62.31**	**0.61**	**60.79**	**0.58**	**1.79**
	RelF	60.18	0.59	56.23	0.52	16.44
	RSD	60.49	0.59	56.23	0.52	9.29
	Aleph – Featurize	55.32	0.50	55.32	0.50	104.70
Financial	Wordification	86.75	0.50	**86.75**	0.50	**4.65**
	RelF	**97.85**	**0.92**	86.70	0.50	260.93
	RSD	86.75	0.50	79,06	0.50	533.68
	Aleph - Featurize	86.75	0.50	**86.75**	0.50	525.86

RSD was specified to construct features with a maximum length of a feature body of 8. None of the constructed features were discarded as the minimum example coverage of the algorithm was set to 1.

Aleph was run in the feature construction mode (named AlephFeaturize in the evaluation workflow) with coverage as the evaluation function and maximal clause length of 4. The minimal number of positive examples was set to 1 and the maximal number of false positives to 0.

RelF, the most relevant of the algorithms in the TreeLiker software (Kuželka & Železný, 2011), was run in the default setting as well, but it is not clear from the documentation what exactly are the default parameter values. RelF expects a feature *template* from the user. In this case, we constructed relatively simple templates (enabling features with depth 1), since constructing and selecting more complex templates is out of scope for the analysis in this paper. It should be noted that templates more finely tuned to a particular domain could yield significantly better results. RelF also supports continuous attributes, but since in our experiments all approaches were given a discretized dataset, this feature could not be exploited.

6.3. Results comparison

The results of the experiments on multiple datasets, presented in Table 3, show the classification accuracy and the ROC AUC obtained by the J48 and LibSVM learners (when applied on the data obtained as a result of propositionalization approaches), as well as the run-times needed for propositionalization. The run-time performance for each algorithm was done by measuring the time an algorithm took to propositionalize the full database in each domain. The results show that the wordification methodology achieves scores comparable to the state-of-the-art propositionalization algorithms RSD and RelF, as well as compared to propositionalization performed by using features constructed by Aleph, while the run-time required for transforming the database into its propositional form is much faster.

In terms of classification accuracy obtained by the J48 classifier, wordification performs favorably compared to other propositionalization techniques, except on the Trains dataset (without the car's position attribute) and the Financial dataset. Poor performance on the Trains dataset can be explained by examining the J48 tree in the wordified Trains dataset with the position attribute, where the J48 classifier puts the cars_position_3 attribute into the root of the decision tree. Because of the absence of this attribute in the first dataset and the usage of only unigram words, the decision tree failed to find a clear distinction between the positive and negative examples (this problem can be solved by using bigrams of witems). Similar results were obtained using the LibSVM classifier where wordification achieved the best results on every dataset except for the two variants of the Trains data.

From the point of view of run-times, wordification is clearly the most efficient system, as it outperforms other techniques on every dataset. The true value of the wordification methodology, its low time-complexity, shows even more drastically on larger datasets, such as Carcinogenesis and Financial datasets, where it achieves comparable classification results in up to 100-times faster manner (compared to RSD or Aleph feature construction).

In order to statistically compare classification accuracies of multiple propositionalization approaches (separately for each of the classifiers) on multiple datasets, we applied the Friedman test (Friedman, 1937) using significance level $\alpha = 0.05$ and the corresponding Nemenyi post hoc test (Nemenyi, 1963). This approach is used as an alternative to the *t*-test, which is proven to be

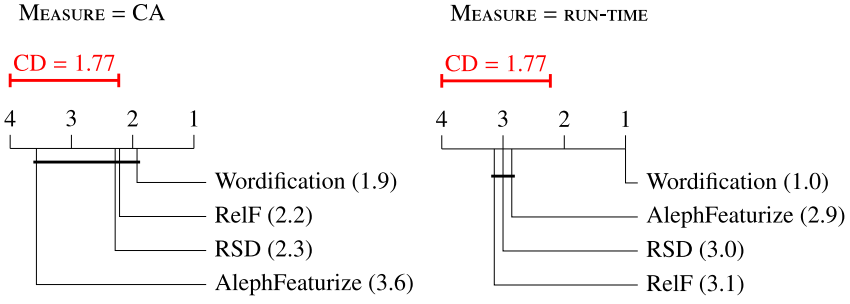

Fig. 10. Critical distance diagram for the reported classification accuracy (left; not enough evidence to prove that any algorithm performs better) and run-time (right; significant differences for $\alpha = 0.05$) results. The numbers in parentheses are the average ranks.

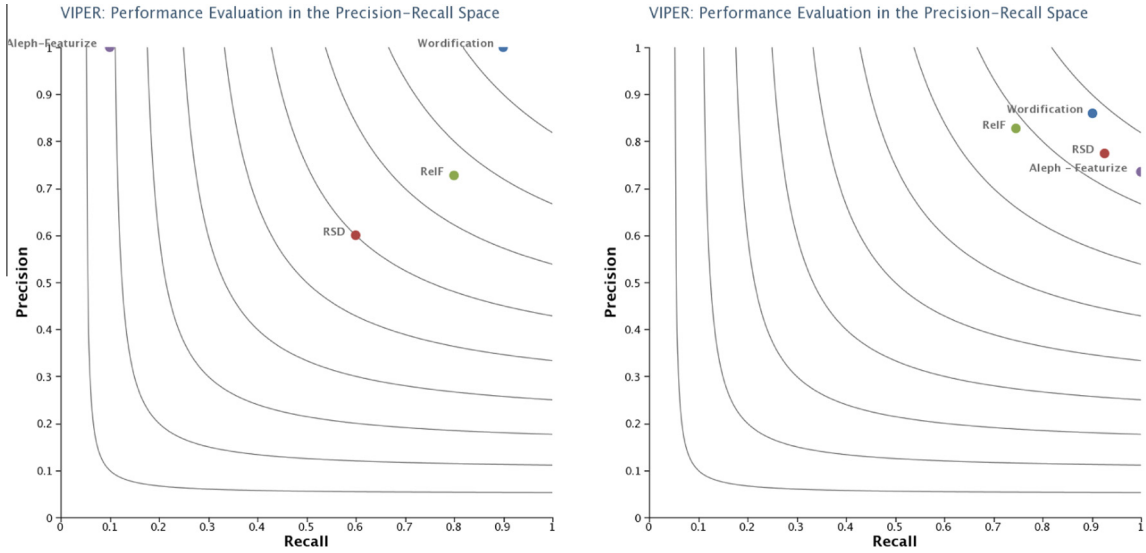

Fig. 11. The VIPER visualization showing evaluations of the standard J48 algorithm after applying propositionalizaton techniques. In the Trains dataset (left), 'East' was selected as the target class, while in the IMDB dataset (right) positive class was selected as the target.

Table 4
Table properties of the experimental data.

	# Rows	# Attributes
IMDB		
Movies	166	4
Roles	7738	2
Actors	7118	4
Movie_genres	408	2
Movie_directors	180	2
Directors	130	3
Director_genres	243	3
Accidents		
Accident	102,756	10
Person	201,534	10

Table 5
Document properties after applying the wordification methodology.

Domain	Individual	# Examples	# Words	# Words after filtering
IMDB	Movie	166	7453	3234
Accidents	Accident	102,759	186	79

inappropriate for testing multiple algorithms on multiple datasets (Demšar, 2006).

The Friedman test ranks the algorithms for each dataset, the best performing algorithm getting the rank of 1, the second best rank 2, etc. In the case of ties, average ranks are assigned. The Friedman test then compares the average ranks of the algorithms. The null-hypothesis states that all the algorithms are equivalent and so their ranks should be equal. If the null-hypothesis is rejected, we can proceed with a post hoc test, in our case the Nemenyi test. The Nemenyi test is used when we want to compare multiple algorithms to each other. The performance of the algorithms is significantly different if the average ranks differ by at least the *critical distance* (CD), as defined by Demšar (2006). This test can be visualized compactly with a critical distance diagram; see Fig. 10 for classification accuracy (CA) and run-time, when using J48 as the selected classifier (omitting AUC due to similar results obtained as for CA).

The described statistical test was performed using J48 for the three reported measures: classification accuracy, AUC and run-time. The validation yielded the following. For classification accuracy and AUC, there is not enough evidence to prove that any propositionalization algorithm on average performs better than the others (Fig. 10 left, for significance level $\alpha = 0.05$), even though wordification achieves the best results on 5 out of 7 benchmarks. This is due to the fact that the test takes into account the order of *all* algorithms, not only one versus the others.

We repeated the same statistical analysis for the LibSVM results, where the conclusion ended up the same. For classification accuracy and AUC, there is not enough evidence to prove that any propositionalization algorithm on average performs better than the others, even though wordification also achieves the best results on 5 out of 7 benchmarks.

For run-time, the results are statistically significant in favor of wordification; see the critical distance diagram in the right part of Fig. 10. The diagram tells us that the wordification approach performs statistically significantly faster than other approaches, under the significance level $\alpha = 0.05$. Other approaches fall within the same critical distance and no statistically significant difference was detected.

As shown in Fig. 8, the results of the Cross Validation widget (precision, recall, F-score) are connected to the input of the VIPER (Visual Performance Evaluation) widget. VIPER is an alternative evaluation visualization (Sluban, Gamberger, & Lavrač, 2014), implemented in the ClowdFlows data mining platform, which displays the results as points in the two dimensional precision-recall space (for the selected target class). Fig. 11 presents the VIPER performance visualization, evaluating J48 and LibSVM results after applying wordification, RSD, RelF and Aleph feature construction as propositionalizaton techniques. The results are presented in the so-called precision-recall space, where each point represents the result of an algorithm. Points closer to the upper-right corner have higher precision and recall values. F-measure values are presented as isolines (contour lines) in the precision-recall space, which allows a simple comparison of algorithm performances.

From the results shown in Fig. 11 we can conclude that in terms of precision and recall J48 achieves best results using the wordification propositionalization. Using the wordification methodology, not only a higher percentage of positive examples was retrieved (higher recall score), but also a slightly higher percentage of correctly classified examples of the target class (higher precision score) compared to other propositionalization techniques.

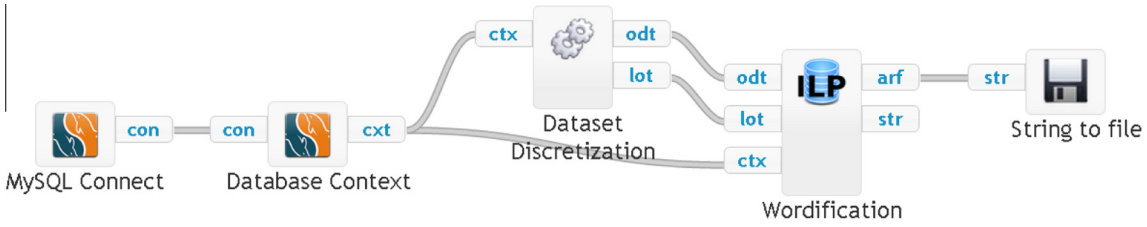

Fig. 12. Clowdflows wordification workflow used for feature construction before applying association rule learning. This workflow is publicly available at http://clowdflows.org/workflow/3969/. The abbreviations (not important for understanding the workflow) on the input and output stubs are as follows: *con* connection, *ctx* context, *odt* Orange data table, *lot* list of Orange data tables, *str* string, *arf* ARFF file, *ins* instances, *lrn* learner, *cla* classifier.

goodMovie ← director_genre_drama, movie_genre_thriller, director_name_AlfredHitchcock.
(Support: 5.38% Confidence: 100.00%)

movie_genre_drama ← goodMovie, actor_name_RobertDeNiro.
(Support: 3.59% Confidence: 100.00%)

director_name_AlfredHitchcock ← actor_name_AlfredHitchcock.
(Support: 4.79% Confidence: 100.00%)

director_name_StevenSpielberg ← goodMovie, movie_genre_adventure, actor_name_TedGrossman.
(Support: 1.79% Confidence: 100.00%)

Fig. 13. Examples of interesting association rules discovered in the IMDB database.

noInjuries ← accident_trafficDensity_rare, accident_location_parkingLot.
(Support: 0.73% Confidence: 97.66%)

person_gender_male ← person_vehicleType_motorcycle.
(Support: 0.11% Confidence: 99.12%)

Fig. 14. Examples of interesting association rules discovered in the accidents database.

7. Applications

This section presents results of association rule learning experiments on two real-life relational databases: a collection of best and worst movies from the Internet Movie DataBase (IMDB) and a database of Slovenian traffic accidents. Tables 4 and 5 list the characteristics of both databases.

The preprocessing procedure was performed on the two databases as follows. First, the wordification step was applied. As shown in Fig. 12, we used ClowdFlows to read the relational data from the MySQL database, discretize continuous attributes and apply the propositionalization step. Due to lack of support for association rule learning in the ClowdFlows platform, the results of the wordification feature construction step were saved as an ARFF file and imported into RapidMiner (Mierswa, Wurst, Klinkenberg, Scholz, & Euler, 2006). Using RapidMiner we first removed irrelevant features (which have the same value across all the examples), which resulted in the reduction of the features to less than half of the original (see Table 5). In order to prepare the data for association rule mining, we also binarized the data: after experimenting with different TF-IDF thresholds, features with a higher TF-IDF weight than 0.06 were assigned the value *true* and *false* otherwise.

7.1. IMDB database

The complete IMDB database is publicly available in the SQL format.[11] This database contains tables of movies, actors, movie genres, directors, director genres.

The evaluation database used in our experiments consists only of the movies whose titles and years of production exist on IMDB's top 250 and bottom 100 chart. The database therefore consisted of 166 movies, along with all of their actors, genres and directors. Movies present in the IMDB's top 250 chart were added an additional label *goodMovie*, while those in the bottom 100 were marked as *badMovie*. Additionally, attribute age was discretized; a movie was marked as *old* if it was made before 1950, *fairlyNew* if it was produced between 1950 and 2000 and *new* otherwise.

After preprocessing the dataset using the wordification methodology, we performed association rule learning. Frequent itemsets were generated using RapidMiner's FP-growth implementation (Mierswa et al., 2006). Next, association rules for the resulting frequent itemsets were produced. Among all the discovered rules, several interesting rules were found. Fig. 13 presents some of the interesting rules selected by the expert. The first rule states that if the movie's genre is thriller and is directed by Alfred Hitchcock, who is also known for drama movies, then the movie is considered to be good. The second rule we have selected concludes that if the movie is good and Robert De Niro acts in it, than it must be a drama. The third interesting rule shows that Alfred Hitchcock acts only in the movies he also directs. The last rule concludes that if Ted Grossman acts in a good adventure movie, then the director is Steven Spielberg. Note that Ted Grossman usually plays the role of a stunt coordinator or performer.

[11] http://www.webstepbook.com/supplements/databases/imdb.sql.

7.2. Traffic accident database

The second dataset consists of all accidents that happened in Slovenia's capital Ljubljana between years 1995 and 2005. The data is publicly accessible from the national police department website.[12] The database contains the information about accidents along with all the accident's participants.

The data already contained discretized attributes, so further discretization was not needed. Similarly to the IMDB database, preprocessing using wordification methodology, FP-growth itemset mining and association rule mining were performed. Fig. 14 presents some of the interesting rules found in the Slovenian traffic accidents dataset.

The first rule indicates that if the traffic is rare and the accident happened in a parking lot, then no injuries occurred. The second rule implies that whenever a motorcycle is involved in an accident, a male person is involved.

8. Conclusions

This paper presents the propositionalization technique called wordification, which aims at constructing a propositional table using simple and easy to understand features. This methodology is inspired by text mining and can be seen as a transformation of a relational database into a corpus of documents, where document 'words' are constructed from attribute values by concatenating each table name, attribute name and value (called word-item or witem in this paper) into a single named-entity. As is typical for propositionalization methods, after the wordification step any propositional data mining algorithm can be applied.

As shown in the experiments on seven standard ILP datasets, the proposed wordification approach using the J48 and LibSVM classifiers performs favorably (in terms of accuracy and efficiency), compared to state-of-the-art propositionalization algorithms (RSD, RelF) as well as compared to propositionalization performed by using features constructed by Aleph. In addition, the proposed approach has the advantage of producing easy to understand hypotheses, using much simpler features than RSD and other systems, which construct complex logical features as conjunctions of first-order literals. It is interesting to observe that in wordification feature simplicity is compensated by the mechanism of feature weighting, inherited from text mining, which successfully compensates for the loss of information compared to complex relational features constructed by other propositionalization algorithms. In our experiments we also considered feature construction using *n*-grams. However, our preliminary experiments indicate that in larger domains this technique should be coupled with feature selection algorithms, which we plan to address in our further work.

Other advantages of wordification, to be explored in further work, include the capacity to perform clustering on relational databases; while this can be achieved also with other propositionalization approaches, wordification may successfully exploit document similarity measures and word clouds as easily understandable means of cluster visualization.

The implementation of the entire experimental workflow (from connecting to a relational database management server to visualizing the experimental results and evaluation) in the web-based data mining platform ClowdFlows is another major contribution, which will enable ILP researchers to reuse the developed software in future experimentation. To the best of our knowledge, this is the only workflow-based implementation of ILP algorithms in a platform accessible through a web browser, enabling simple workflow adaptation to the user's needs. Adding of new ILP algorithms to the platform is also possible by exposing the algorithm as a web service. This may significantly contribute to the accessibility and popularity of ILP and RDM methods in the future.

In terms of reusability of the workflows, accessible by a single click on a web page where the workflow is exposed, the ClowdFlows implementation of propositionalization algorithms is a significant step towards making the ILP legacy accessible to the research community in a systematic and user-friendly way. An additional building block in this vision is the incorporation of the VIPER visual performance evaluation engine, which enables algorithm comparison in terms of precision and recall, simplifying the experimental comparisons and results interpretation.

In future work, we will address other problem settings (such as clustering) and use the approach for solving real-life relational problems. Moreover, we plan to use the approach in a more elaborate scenario of mining heterogeneous data sources, involving a mixture of information from databases and text corpora. We will also further investigate the strength of *n*-gram construction and feature weighting, as used in the text mining community, in propositional and Relational Data Mining, as our results indicate that these mechanisms may successfully be used to compensate for the loss of information compared to constructing complex logical features.

Acknowledgments

We are grateful to Marko Grobelnik and Peter Ljubič for their initial work on this topic. This work was supported by the Slovenian Research Agency grants, and the FP7 European Commission project "Machine understanding for interactive storytelling" (MUSE, Grant agreement no: 296703).

Appendix A

Tables A.6 and A.7.

Table A.6
Evaluation of different feature weighting techniques. The bolded items indicate the best results.

Domain	Weighting	J48-accuracy [%]	J48-AUC
Trains without position	TF-IDF	50.00	0.50
	TF	**85.00**	**0.85**
	Binary	35.00	0.35
Trains	TF-IDF	**95.00**	**0.95**
	TF	80.00	0.80
	Binary	70.00	0.70
Mutagenesis 42	TF-IDF	97.62	0.96
	TF	97.62	0.96
	Binary	97.62	0.96
Mutagenesis 188	TF-IDF	**68.62**	**0.55**
	TF	68.09	0.54
	Binary	**68.62**	**0.55**
IMDB	TF-IDF	81.93	0.75
	TF	81.93	0.75
	Binary	81.93	0.75
Carcinogenesis	TF-IDF	62.31	0.61
	TF	62.61	0.61
	Binary	**62.92**	**0.62**
Financial	TF-IDF	86.75	0.50
	TF	86.75	0.50
	Binary	86.75	0.50

[12] http://www.policija.si/index.php/statistika/prometna-varnost.

Table A.7
Evaluation of different number of witems. The bolded items indicate the best results.

Domain	k	J48-Accuracy [%]	J48-AUC	Time [s]
Trains without position	1	50.00	0.50	**0.12**
	2	**75.00**	**0.75**	0.15
	3	**75.00**	**0.75**	0.20
Trains	1	**95.00**	**0.95**	**0.12**
	2	75.00	0.75	0.16
	3	70.00	0.70	0.22
Mutagenesis 42	1	**97.62**	**0.96**	**0.65**
	2	**97.62**	**0.96**	0.83
	3	92.86	0.88	0.88
Mutagenesis 188	1	**68.62**	**0.55**	**1.25**
	2	**68.62**	**0.55**	2.26
	3	66.49	0.50	2.68
IMDB	1	73.49	0.50	**0.16**
	2	73.49	0.50	0.20
	3	73.49	0.50	0.25
Carcinogenesis	1	**56.84**	**0.56**	**5.31**
	2	51.67	0.51	6.65
	3	52.58	0.51	7.04
Financial	1	86.75	0.50	**4.11**
	2	86.75	0.50	4.24
	3	86.75	0.50	4.38

References

Avila Garcez, A., & Zaverucha, G. (1999). The connectionist inductive learning and logic programming system. *Applied Intelligence, 11*(1), 59–77.

Ceci, M., & Appice, A. (2006). Spatial associative classification: Propositional vs structural approach. *Journal of Intelligent Information Systems, 27*(3), 191–213.

Ceci, M., Appice, A., & Malerba, D. (2008). Emerging pattern based classification in relational data mining. In S. Bhowmick, J. Kng, & R. Wagner (Eds.), *Database and expert systems applications. Lecture notes in computer science* (Vol. 5181, pp. 283–296). Berlin Heidelberg: Springer.

De Raedt, L. (2008). *Logical and relational learning*. Springer.

Debnath, A. K., Lopez de Compadre, R. L., Debnath, G., Shusterman, A. J., & Hansch, C. (1991). Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry, 34*(2), 786–797.

Demšar, J., Zupan, B., Leban, G., & Curk, T. (2004). Orange: From experimental machine learning to interactive data mining. In J.-F. Boulicaut, F. Esposito, F. Giannotti, & D. Pedreschi (Eds.), *PKDD 2004. Proceedings of 8th european conference on principles and practice of knowledge discovery in databases September 20–24, 2004. Lecture notes in computer science* (Vol. 3202, pp. 537–539). Pisa, Italy: Springer.

Demšar, J. (2006). Statistical comparison of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30.

Džeroski, S., & Lavrač, N. (Eds.). (2001). *Relational data mining*. Springer.

Fayyad, U., & Irani, K. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In R. Bajcsy (Ed.), *IJCAI-93. Proceedings of the 13th international joint conference on artificial intelligence, August 28–September 3, 1993* (pp. 1022–1029). Chambéry, France: Morgan Kaufman.

Flach, P. A., Lachiche, N. (1999). 1BC: A first-order Bayesian classifier. In: *Proceedings of the 9th international workshop on inductive logic programming* (pp. 92–103).

França, M., Zaverucha, G., & d'Avila Garcez, A. (2014). Fast relational learning using bottom clause propositionalization with artificial neural networks. *Machine Learning, 94*(1), 81–104.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association, 32*, 675–701.

Guo, H., & Viktor, H. (2008). Multirelational classification: A multiple view approach. *Knowledge and Information Systems, 17*(3), 287–312.

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation, 28*, 11–21.

Knobbe, A. J. (Ed.). (2005). *Multi-relational data mining. Frontiers in artificial intelligence and applications* (Vol. 145). IOS Press.

Kramer, S., Pfahringer, B., & Helma, C. (1998). Stochastic propositionalization of non-determinate background knowledge. In D. Page (Ed.), *ILP-98. Proceedings of the 8th international workshop on inductive logic programming, July 22–24, 1998. Lecture notes in computer science* (Vol. 1446, pp. 80–94). Madison, Wisconsin, USA: Springer.

Kranjc, J., Podpečan, V., & Lavrač, N. (2012). Clowdflows: A cloud based scientific workflow platform. In P. A. Flach, T. D. Bie, & N. Cristianini (Eds.), *ECML PKDD 2012. Proceedings (part II) of the machine learning and knowledge discovery in databases – European conference, September 24–28, 2012. Lecture notes in computer science* (Vol. 7524, pp. 816–819). Bristol, UK: Springer.

Krogel, M. A., Rawles, S., Železný, F., Flach, P. A., Lavrač, N., & Wrobel, S. (2003). Comparative evaluation of approaches to propositionalization. In T. Horváth (Ed.), *ILP 2003. Proceedings of the 13th international conference on inductive logic programming, September 29–October 1, 2003. Lecture notes in computer science* (Vol. 2835, pp. 197–214). Szeged, Hungary: Springer.

Krogel, M. A., & Wrobel, S. (2001). Transformation-based learning using multirelational aggregation. In C. Rouveirol & M. Sebag (Eds.), *ILP 2001. Proceedings of the 11th international conference on inductive logic programming, September 9–11, 2001. Lecture notes in computer science* (Vol. 2157, pp. 142–155). Strasbourg, France: Springer.

Kuželka, O., & Železný, F. (2011). Block-wise construction of tree-like relational features with monotone reducibility and redundancy. *Machine Learning, 83*(2), 163–192.

Lavrač, N., & Džeroski, S. (1994). Inductive logic programming. In *WLP* (pp. 146–160). Springer.

Lavrač, N., Džeroski, S., & Grobelnik, M. (1991). Learning nonrecursive definitions of relations with LINUS. In Y. Kodratoff (Ed.), *EWSL-91. Proceedings of the european working session on learning, March 6–8, 1991. Lecture notes in computer science* (Vol. 482, pp. 265–281). Porto, Portugal: Springer.

Lavrač, N., & Flach, P. A. (2001). An extended transformation approach to inductive logic programming. *ACM Transactions on Computational Logic, 2*(4), 458–494.

Michie, D., Muggleton, S., Page, D., Srinivasan, A. (1994). To the international computing community: A new East–West challenge. Tech. rep., Technical report, Oxford University Computing laboratory, Oxford, UK.

Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., & Euler, T. (2006). YALE rapid prototyping for complex data mining tasks. In *Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining (KDD'06)* (pp. 935–940). NY, USA: ACM Press.

Muggleton, S. (Ed.). (1992). *Inductive logic programming*. London: Academic Press.

Muggleton, S. (1995). Inverse entailment and progol. *New Generation Computing, Special issue on Inductive Logic Programming, 13*(3–4), 245–286.

Nemenyi, P. B. (1963). *Distribution-free multiple comparisons* (Ph.D. thesis).

Perovšek, M., Vavpetič, A., Cestnik, B., & Lavrač, N. (2013). A wordification approach to relational data mining. In J. Fürnkranz, E. Hüllermeier, & T. Higuchi (Eds.), *DS 2013. Proceedings of the 16th international conference on discovery science, October 6–9, 2013. Lecture notes in computer science* (Vol. 8140, pp. 141–154). Singapore: Springer.

Perovšek, M., Vavpetič, A., Lavrač, N. (2012). A wordification approach to relational data mining: Early results. In: F. Riguzzi, F. Železný (Eds.), *ILP 2012. Proceedings of late breaking papers of the 22nd international conference on inductive logic programming* (pp. 56–61). Dubrovnik, Croatia, September 17–19, 2012. Vol. 975 of CEUR Workshop Proceedings. CEUR-WS.org.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management, 24*(5), 513–523.

Sluban, B., Gamberger, D., & Lavrač, N. (2014). Ensemble-based noise detection: Noise ranking and visual performance evaluation. *Data Mining and Knowledge Discovery*, 1–39.

Srinivasan, A. (2007). Aleph manual. <http://www.cs.ox.ac.uk/activities/machinelearning/Aleph/>.

Srinivasan, A., King, R. D., Muggleton, S., & Sternberg, M. J. (1997). Carcinogenesis predictions using ILP. In N. Lavrač & S. Džeroski (Eds.), *ILP-97. Proceedings of the 7th international workshop on inductive logic programming, September 17–20, 1997. Lecture notes in computer science* (Vol. 1297, pp. 273–287). Prague, Czech Republic: Springer.

Srinivasan, A., Muggleton, S., King, R., Sternberg, M. (1994). Mutagenesis: ILP experiments in a non-determinate biological domain. In: S. Wrobel (Ed.), *Proceedings of the 4th international workshop on inductive logic programming*. Vol. 237, GMD-Studien. Gesellschaft für Mathematik und Datenverarbeitung MBH (pp. 217–232).

Vavpetič, A., & Lavrač, N. (2013). Semantic subgroup discovery systems and workflows in the SDM-toolkit. *The Computer Journal, 56*(3), 304–320.

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Železný, F., & Lavrač, N. (2006). Propositionalization-based relational subgroup discovery with RSD. *Machine Learning, 62*(1–2), 33–63.