Feature Selection in Hierarchical Feature Spaces

Petar Ristoski and Heiko Paulheim

University of Mannheim, Germany
Research Group Data and Web Science
{petar.ristoski,heiko}@informatik.uni-mannheim.de

Abstract. Feature selection is an important preprocessing step in data mining, which has an impact on both the runtime and the result quality of the subsequent processing steps. While there are many cases where hierarchic relations between features exist, most existing feature selection approaches are not capable of exploiting those relations. In this paper, we introduce a method for feature selection in hierarchical feature spaces. The method first eliminates redundant features along paths in the hierarchy, and further prunes the resulting feature set based on the features' relevance. We show that our method yields a good trade-off between feature space compression and classification accuracy, and outperforms both standard approaches as well as other approaches which also exploit hierarchies.

Keywords: Feature Subset Selection, Hierarchical Feature Spaces, Feature Space Compression.

1 Introduction

In machine learning and data mining, data is usually described as a vector of *features* or *attributes*, such as the age, income, and gender of a person. Based on this representation, predictive or descriptive models are built.

For many practical applications, the set of features can be very large, which leads to problems both with respect to the performance as well as the accuracy of learning algorithms. Thus, it may be useful to reduce the set of features in a preprocessing step, i.e., perform a *feature selection* [2,8]. Usually, the goal is to compress the feature space as good as possible without a loss (or even with a gain) in the accuracy of the model learned on the data.

In some cases, external knowledge about attributes exist, in particular about their hierarchies. For example, a product may belong to different categories, which form a hierarchy (such as *Headphones* < *Accessories* < *Consumer Electronics*). Likewise, hyponym and hyperonym relations can be exploited when using bag-of-words features for text classification [3], or hierarchies defined by ontologies when generating features from Linked Open Data [10].

In this paper, we introduce an approach that exploits hierarchies for feature selection in combination with standard metrics, such as *information gain* or *correlation*. With an evaluation on a number of synthetic and real world datasets,

S. Džeroski et al. (Eds.): DS 2014, LNAI 8777, pp. 288–300, 2014.

we show that using a combined approach works better than approaches not using the hierarchy, and also outperforms existing approaches for feature selection that exploit the hierarchy.

The rest of this paper is structured as follows. In section 2, we formally define the problem of feature selection in hierarchical feature spaces. In section 3, we give an overview of related work. Section 4, we introduce our approach, followed by an evaluation in section 5. We conclude with a summary and an outlook on future work.

2 Problem Statement

We describe each instance as an n-dimensional binary feature vector $\langle v_1, v_2, ..., v_n \rangle$, with $v_i \in \{0, 1\}$ for all $1 \leq i \leq n$. We call $V = \{v_1, v_2, ..., v_n\}$ the *feature space*.

Furthermore, we denote a hierarchic relation between two features v_i and v_j as $v_i < v_j$, i.e., v_i is more specific than v_j. For hierarchic features, the following implication holds:

$$v_i < v_j \rightarrow (v_i = 1 \rightarrow v_j = 1)\,, \tag{1}$$

i.e., if a feature v_i is set, then v_j is also set. Using the example of product categories, this means that a product belonging to a category also belongs to that product's super categories. Note that the implication is not symmetric, i.e., even if $v_i = 1 \rightarrow v_j = 1$ holds for two features v_i and v_j, they do not necessarily have to be in a hierarchic relation. We furthermore assume transitivity of the hierarchy, i.e.,

$$v_i < v_j \wedge v_j < v_k \rightarrow v_i < v_k \tag{2}$$

The problem of *feature selection* can be defined as finding a projection of V to V', where $V' \subseteq V$. Ideally, V' is much smaller than V.

Feature selection is usually regarded with respect to a certain problem, where a solution S using a subset V' of the features yields a certain performance $p(V')$, i.e., p is a function

$$p : \mathcal{P}(V) \rightarrow [0, 1], \tag{3}$$

which is normalized to $[0, 1]$ without loss of generality. For example, for a classification problem, the accuracy achieved by a certain classifier on a feature subset can be used as the performance function p. Besides the quality, another interesting measure is the *feature space compression*, which we define as

$$c(V') := 1 - \frac{|V'|}{|V|} \tag{4}$$

Since there is a trade-off between the feature set and the performance, an overall target function is, e.g., the harmonic mean of p and c.

For most problems, we expect the optimal features to be somewhere in the *middle* of the hierarchy, while the most general features are often too general for predictive models, and the most specific ones are too specific. The hierarchy level of the most valuable features depends on the task at hand. Fig. 1 shows a small part of the hierarchical feature space extracted for dataset Sports Tweets T (see section 5.1). If the task is to classify tweets into sports and non sports

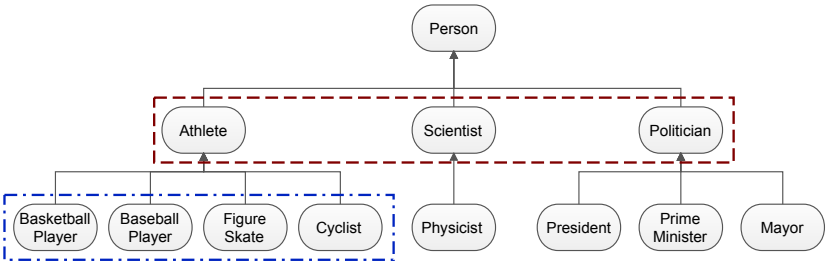


Fig. 1. An example hierarchy of binary features

related, the optimal features are those in the upper rectangle, if the task is to classify them by different kinds of sports, then the features in the lower rectangle are more valuable.

3 Related Work

Feature selection is a very important and well studied problem in the literature. The objective is to identify features that are correlated with or predictive of the class label. Generally, all feature selection methods can be divided into two broader categories: wrapper methods and filter methods (John et al. [4] and Blum et al. [1]). The wrapper methods use the predictive accuracy of a predetermined learning method to evaluate the relevance of the feature sub set. Because of their large computational complexity, the wrapper methods are not suitable to be used for large feature spaces. Filter methods are trying to select the most representative sub-set of features based on a criterion used to score the relevance of the features. In the literature several techniques for scoring the relevance of features exist, e.g., Information Gain, χ^2 measure, Gini Index, and Odds Ratio. However, standard feature selection methods tend to select the features that have the highest relevance score without exploiting the hierarchical structure of the feature space. Therefore, using such methods on hierarchical feature spaces, may lead to the selection of redundant features, i.e., nodes that are closely connected in the hierarchy and carry similar semantic information.

While there are a lot of state-of-the-art approaches for feature selection in standard feature space [8], only few approaches for feature selection in hierarchical feature space are proposed in the literature. Jeong et al. [3] propose the *TSEL* method using a semantic hierarchy of features based on WordNet relations. The presented algorithm tries to find the most representative and most effective features from the complete feature space. To do so, they select one representative feature from each path in the tree, where path is the set of nodes between each leaf node and the root, based on the *lift* measure, and use χ^2 to select the most effective features from the reduced feature space.

Wang et al. [13] propose a *bottom-up hill climbing* search algorithm to find an optimal subset of concepts for document representation. For each feature in the initial feature space, they use a kNN classifier to detect the k nearest neighbors of each instance in the training dataset, and then use the purity of those

Algorithm 1. Algorithm for initial hierarchy selection strategy.

```
Data: H: Feature hierarchy, F: Feature set, t: Importance similarity threshold,
      s:= Importance similarity measurement {"Information Gain",
      "Correlation"}
Result: F: Feature set
L := leaf nodes from hierarchy H
foreach leaf l ∈ L do
    D := direct ascendants of node l
    foreach node d ∈ D do
        similarity = 0
        if s == "Information Gain" then
            similarity = 1-ABS(IGweight(d)-IGweight(l))
        else
            similarity =Correlation(d,l)
        end
        if similarity ≥ threshold then
            remove l from F
            remove l from H
            break
        end
    end
    add direct ascendants of l to L
end
```

instances to assign scores to features. As shown in section 5.3, the approach is computationally expensive, and not applicable for datasets with a large number of instances. Furthermore, the approach uses a strict policy for selecting features that are as high as possible in the feature hierarchy, which may lead to selecting low-value features from the top levels of the hierarchy.

Lu et al. [6] describe a *greedy top-down* search strategy for feature selection in a hierarchical feature space. The algorithm starts with defining all possible paths from each leaf node to the root node of the hierarchy. The nodes of each path are sorted in descending order based on the nodes' information gain ratio. Then, a greedy-based strategy is used to prune the sorted lists. Specifically, it iteratively removes the first element in the list and adds it to the list of selected features. Then, removes all ascendants and descendants of this element in the sorted list. Therefore, the selected features list can be interpreted as a mixture of concepts from different levels of the hierarchy.

4 Approach

Following the implication shown in Eq. 1, we can assume that if two features subsume each other, they are usually highly correlated to each other and have similar relevance for building the model. Following the definition for "relevance"

by Blum et al. [1], two features v_i and v_j have similar relevance if $1 - |R(v_i) - R(v_j)| \geq t,\ t \rightarrow [0, 1]$, where t is a user specified threshold.

The core idea of our SHSEL approach is to identify features with similar relevance, and select the most valuable abstract features, i.e. features from as high as possible levels of the hierarchy, without losing predictive power. In our approach, to measure the similarity of relevance between two nodes, we use the standard correlation and information gain measure. The approach is implemented in two steps, i.e, initial selection and pruning. In the first step, we try to identify, and filter out the ranges of nodes with similar relevance in each branch of the hierarchy. In the second step we try to select only the most valuable features from the previously reduced set.

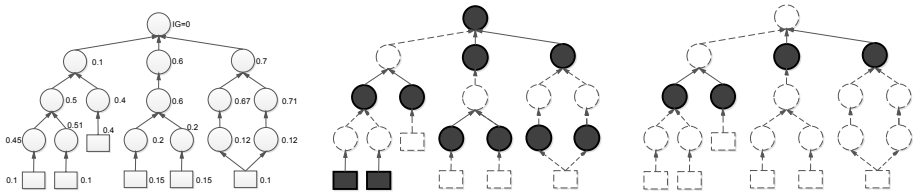
The initial selection algorithm is shown in Algorithm 1. The algorithm takes as input the feature hierarchy H, the initial feature set F, a relevance similarity threshold t, and the relevance similarity measure s to be used by the algorithm. The relevance similarity threshold is used to decide whether two features would be similar enough, thus it controls how many nodes from different levels in the hierarchy will be merged. The algorithm starts with identifying the leaf nodes of the feature hierarchy. Then, starting from each leaf node l, it calculates the relevance similarity value between the current node and its direct ascendants d. The relevance similarity value is calculated using the selected relevance measure s. If the relevance similarity value is greater or equal to the similarity threshold t, then the node from the lower level of the hierarchy is removed from the feature space F. Also, the node is removed from the feature hierarchy H, and the paths in the hierarchy are updated accordingly. For the next iteration, the direct ascendants of the current node are added in the list L.

The algorithm for pruning is shown in Algorithm 2. The algorithm takes as input the feature hierarchy H and the previously reduced feature set F. The algorithm starts with identifying all paths P from all leaf nodes to the root node of the hierarchy. Then, for each path p it calculates the average information gain of all features on the path p. All features that have lower information gain than the average information gain on the path, are removed from the feature space F, and from the feature hierarchy H. In cases where a feature is located on more than one path, it is sufficient that the feature has greater information gain than the average information gain on at least one of the paths. This way, we prevent removing relevant features. Practically, the paths from the leafs to the root node, as well as the average information gain per path, can already be precomputed in the initial selection algorithm. The loop in the lines $3 - 6$ is only added for illustrating the algorithm.

Fig. 2a shows an example hierarchical feature set, with the information gain value of each feature. Applying the initial selection algorithm on that input hierarchical feature set, using information gain as a relevance similarity measurement, would reduce the feature set as shown in Fig. 2b. We can see that all feature pairs that have high relevance similarity value, are replaced with only one feature. However, the feature set still contains features that have a rather

Algorithm 2. Algorithm for pruning strategy.

```
Data: H: Feature hierarchy, F: Feature set
Result: F: Feature set
L := leaf nodes from hierarchy H
P := ∅
foreach leaf l ∈ L do
    p = paths from l to root of H
    add p to P
end
foreach path p ∈ P do
    avg = Information gain average of path p
    foreach node n ∈ path p do
        if IGweight(n) < avg then
            remove n from F
            remove n from H
        end
    end
end
```



a) *Initial Feature Space* **b)** *SHSEL Initial Selection* **c)** *SHSEL Pruning*

Fig. 2. Illustration of the two steps of the proposed hierarchical selection strategy

small relevance value. In Fig. 2c we can see that running the pruning algorithm, removes the unnecessary features.

For n features and m instances, iterating over the features, and computing the correlation or information gain with each feature's ancestor takes $O(am)$, given that a feature has an average of a ancestors.[1] Thus, the overall computational complexity is $O(amn)$. It is, however, noteworthy that the selection of the features in both algorithms can be executed in parallel.

5 Evaluation

We perform an evaluation, both on real and on synthetic datasets, and compare different configurations of our approach to standard approaches for feature selection, as well as the approaches described in Section 3.

[1] a is 1 in the absence of multiple inheritance, and close to 1 in most practical cases.

5.1 Datasets

In our evaluation, we used five real-world datasets and six synthetically generated datasets. The real-world datasets cover different domains, and are used for different classification tasks. Initially, the datasets contained only the instances with a given class label, which afterwards were extended with hierarchical features.

For generating the hierarchical features, we used the RapidMiner Linked Open Data extension [11], which is able to identify Linked Open Data resources inside the given datasets, and extract different types of features from any Linked Open Data source. In particular, we used DBpedia Spotlight [7], which annotates a text with concepts in DBpedia, a structured data version of Wikipedia [5]. From those, we can extract further features, such as the types of the concepts found in a text. For example, when the concept *Kobe Bryant* is found in a text, we can extract a hierarchy of types (such as *Basketball Player* < *Athlete* < *Person*), as well as a hierarchy of categories (such as *Shooting Guards* < *Basketball* < *Sports*). The generation of the features is independent from the class labels of the instances (i.e., the classification task), and it is completely unbiased towards any of the feature selection approaches.

The following datasets were used in the evaluation (see Table 1):

- *Sports Tweets T* dataset was used for existing Twitter topic classifier[2], where the classification task is to identify sports related tweets. The hierarchical features were generated by extracting all types of the discovered DBpedia concepts in each tweet.
- *Sports Tweets C* is the same dataset as the previous one, but using categories instead of types.
- The *Cities* dataset was compiled from the Mercer ranking list of the most and the least livable cities, as described in [9]. The classification task is to classify each city into high, medium, and low livability. The hierarchical features were generated by extracting the types for each city.
- The *NY Daily* dataset is a set of crawled news texts, which are augmented with sentiment scores[3]. Again, the hierarchical features were generated by extracting types.
- The *StumbleUpon* dataset is the training dataset used for the StumbleUpon Evergreen Classification Challenge[4]. To generate the hierarchical features, we used the Open Directory Project[5] to extract categories for each URL in the dataset.

To generate the synthetic datasets, we start with generating features in a flat hierarchy, i.e. all features are on the same level. The initial features were

[2] https://github.com/vinaykola/twitter-topic-classifier/blob/master/training.txt

[3] http://dws.informatik.uni-mannheim.de/en/research/identifying-disputed-topics-in-the-news

[4] https://www.kaggle.com/c/stumbleupon

[5] http://www.dmoz.org/

Table 1. Evaluation Datasets

Name	Features	# Instances	Class Labels	# Features
Sports Tweets T	DBpedia Direct Types	1,179	positive(523); negative(656)	4,082
Sports Tweets C	DBpedia Categories	1,179	positive(523); negative(656)	10,883
Cities	DBpedia Direct Types	212	high(67); medium(106); low(39)	727
NY Daily Headings	DBpedia Direct Types	1,016	positive(580); negative(436)	5,145
StumbleUpon	DMOZ Categories	3,020	positive(1,370); negative(1,650)	3,976

generated using a polynomial function, and then discretizing each attribute into a binary one. These features represent the *middle layer* of the hierarchy, which are then used to build the hierarchy upwards and downwards. The hierarchical feature implication (1) and the transitivity rule (2) hold for all generated features in the hierarchy. By merging the predecessors of two or more neighboring nodes from the middle layer, we are able to create more complex branches inside the hierarchy. We control the depth and the branching factor of the hierarchy with two parameters D and B, respectively. Each of the datasets that we use for the evaluation contains 1000 instances, and contains 300 features in the middle layer. The datasets are shown in Table 2.

5.2 Experiment Setup

In order to demonstrate the effectiveness of our proposed feature selection in hierarchical feature space, we compare the proposed approach with the following methods:

- *CompleteFS*: the complete feature set, without any filtering.
- *SIG*: standard feature selection based on information gain value.
- *SC*: Standard feature selection based on feature correlation.
- *TSEL Lift*: tree selection approach proposed in [3], which selects the most representative features from each hierarchical branch based on the *lift* value.
- *TSEL IG*: this approach follows the same algorithm as *TSEL Lift*, but uses information gain instead of lift.
- *HillClimbing*: bottom-up hill-climbing approach proposed in [13].We use $k = 10$ for the kNN classifier used for scoring.
- *GreedyTopDown*: greedy based top-down approach described in [6], which tries to select the most valuable features from different levels of the hierarchy.
- *initialSHSEL IG* and *initialSHSEL C*: our proposed initial selection approach shown with Algorithm 1, using information gain and correlation as relevance similarity measurement, respectively.
- *pruneSHSEL IG* and *pruneSHSEL C*: our proposed pruning selection approach shown with Algorithm 2, applied on previously reduced feature set, using *initialSHSEL IG* and *initialSHSEL C*, respectively.

For all algorithms involving a threshold (i.e., SIG, SC, and the variants of SHSEL), we use thresholds between 0 and 1 with a step width of 0.01.

For conducting the experiments, we used the RapidMiner machine learning platform and the RapidMiner development library. For SIG and SC, we used the built-in RapidMiner operators. The proposed approach for feature selection, as

Table 2. Synthetic Evaluation Datasets

Name	Feature Generation Strategy	# Instances	Classes	# Features
S-D2-B2	D=2; B=2	1,000	positive(500); negative(500)	1,201
S-D2-B5	D=2; B=5	1,000	positive(500); negative(500)	1,021
S-D2-B10	D=2; B=10	1,000	positive(500); negative(500)	961
S-D4-B2	D=4; B=2	1,000	positive(500); negative(500)	2,101
S-D4-B5	D=4; B=5	1,000	positive(500); negative(500)	1,741
S-D4-B10	D=4; B=10	1,000	positive(500); negative(500)	1,621

well as all other related approaches, were implemented in a separate operator as part of the RapidMiner Linked Open Data extension. All experiments were run using standard laptop computer with 8GB of RAM and Intel Core i7-3540M 3.0GHz CPU. The RapidMiner processes and datasets used for the evaluation can be found online[6].

5.3 Results

To evaluate how well the feature selection approaches perform, we use three classifiers for each approach on all datasets, i.e., Naïve Bayes, k-Nearest Neighbors (with $k = 3$), and Support Vector Machine. For the latter, we use Platt's sequential minimal optimization algorithm and a polynomial kernel function [12]. For each of the classifiers we were using the default parameters values in RapidMiner, and no further parameter tuning was undertaken. The classification results are calculated using stratified 10-fold cross validation, where the feature selection is performed separately for each cross-validation fold. For each approach, we report accuracy, feature space compression (4), and their harmonic mean.

Results on Real World Datasets. Table 3 shows the results of all approaches. Because of the space constrains, for the *SIG* and *SC* approaches, as well as for our proposed approaches, we show only the best achieved results. The best results for each classification model are marked in bold. As we can observe from the table, our proposed approach outperforms all other approaches in all five datasets for both classifiers in terms of accuracy. Furthermore, we can conclude that our proposed approach delivers the best feature space compression for four out of five datasets. When looking at the harmonic mean, our approach also outperforms all other approaches, most often with a large gap. From the results for the harmonic mean we can conclude that the *pruneSHSEL IG* approach, in most of the cases, delivers the best results

Additionally, we report the runtime of all approaches on different datasets in Fig. 3. The runtime of our approaches is comparable to the standard feature selection approach, *SIG*, runtime. The *HillClimbing* approach has the longest runtime due to the repetitive calculation of the kNN for each instance. Also, the standard feature selection approach *SC* shows a long runtime, which is due to the computation of correlation between all pairs of features in the feature set.

[6] `http://dws.informatik.uni-mannheim.de/en/research/feature-selection-in-hierarchical-feature-spaces`

Table 3. Results on real world datasets

	Sports Tweets T			Sports Tweets C			StumbleUpon			Cities			NY Daily Headings		
	NB	KNN	SVM	NB	KNN	SVM	NB	KNN	SVM	NB	KNN	SVM	NB	KNN	SVM
Classification Accuracy															
CompleteFS	.655	.759	.797	.943	.920	.946	.582	.699	.730	.625	.562	.684	.534	.586	.577
initialSHSEL IG	**.836**	.768	**.824**	**.974**	.768	**.953**	.661	.709	.733	.671	.609	.674	**.688**	.629	.635
initialSHSEL C	.819	.765	.811	.946	**.937**	**.953**	.689	**.723**	.732	.640	.671	.683	.547	.580	.596
pruneSHSEL IG	.791	**.793**	.773	.909	.909	.946	**.717**	.695	**.737**	**.687**	.669	**.689**	**.688**	**.659**	**.671**
pruneSHSEL C	.786	.791	.772	.946	.918	.935	.711	.707	.732	.656	**.687**	.646	.665	**.659**	.661
SIG	.819	.788	.814	.966	.936	.940	.681	.707	.729	.656	.640	.671	.675	.652	.668
SC	.816	.765	.813	.937	.918	.932	.587	.711	.726	.625	.656	.677	.534	.583	.606
TSEL Lift	.641	.740	.787	.836	.855	.893	.570	.613	.690	0	0	0	.498	.544	.565
TSEL IG	.632	.734	.782	.923	.909	.935	.579	.661	.724	.640	.580	.580	.521	.560	.610
HillClimbing	.528	.647	.742	.823	.836	.876	.548	.653	.683	.622	.562	.551	.573	.583	.530
GreedyTopDown	.658	.788	.800	.943	.929	.944	.582	.698	.727	.625	.562	.679	.534	.570	.595
Feature Space Compression															
initialSHSEL IG	.456	.207	.222	.318	.708	.288	.672	.843	.642	.781	**.902**	.779	.858	.322	.631
initialSHSEL C	.231	.173	.290	.321	.264	.228	**.993**	.445	.644	.184	.121	.116	.285	.572	.790
pruneSHSEL IG	**.985**	**.986**	**.969**	.895	**.907**	**.916**	.976	.957	**.975**	**.823**	.466	.452	.912	.817	.817
pruneSHSEL C	.971	.965	.965	**.897**	.857	.861	.966	**.968**	.959	.305	.265	.308	.519	.586	.566
SIG	.360	.741	.038	.380	.847	.574	.940	.615	.604	.774	.775	04	.240	.289	.565
SC	.667	.712	.635	.887	.710	.792	.585	.821	.712	.631	.704	.598	.632	.927	.620
TSEL Lift	.247	.247	.247	.511	.511	.511	.412	.412	.412	0	0	0	**.956**	**.956**	**.956**
TSEL IG	.920	.920	.920	.522	.522	.522	.471	.471	.471	.126	.126	.126	.926	.926	.926
HillClimbing	.770	.770	.770	.748	.748	.748	.756	.756	.756	.817	.817	**.817**	.713	.713	.713
GreedyTopDown	.136	.136	.136	.030	0.030	.030	.285	.285	.285	.048	.048	.048	.135	.135	.135
Harmonic Mean of Classification Accuracy and Feature Space Compression															
initialSHSEL IG	.590	.326	.350	.480	.737	.442	.666	.770	.684	.722	**.727**	**.723**	**.764**	.426	.633
initialSHSEL C	.360	.282	.427	.479	.412	.368	.814	.551	.686	.286	.205	.199	.375	.576	.679
pruneSHSEL IG	**.877**	**.879**	**.860**	.902	**.908**	**.931**	**.827**	.805	**.840**	**.749**	.549	.546	.784	**.729**	**.737**
pruneSHSEL C	.869	.869	.858	**.921**	.886	.896	.820	**.817**	.830	.416	.383	.417	.583	.620	.610
SIG	.500	.764	.073	.545	.889	.713	.789	.658	.660	.710	.701	08	.354	.401	.612
SC	.734	.738	.713	.911	.801	.856	.586	.762	.719	.628	.679	.635	.579	.716	.613
TSEL Lift	.356	.370	.376	.634	.640	.650	.479	.493	.516	0	0	0	.655	.693	.711
TSEL IG	.749	.817	.846	.667	.663	.670	.520	.550	.571	.211	.207	.207	.667	.698	.735
HillClimbing	.626	.703	.756	.784	.790	.807	.636	.701	.718	.706	.666	.658	.635	.641	.608
GreedyTopDown	.225	.232	.232	0.058	.058	.058	.383	.405	.409	.089	.088	.089	.216	.219	.221

Results on Synthetic Datasets. Table 4 shows the results for the different synthetic datasets. Our approaches achieve the best results, or same results as the standard feature selection approach *SIG*. The results for the feature space compression are rather mixed, while again, our approach outperforms all other approaches in terms of the harmonic mean of accuracy and feature space compression. The runtimes for the synthetic datasets, which we omit here, show the same characteristics as for the real-world datasets.

Overall, *pruneSHSEL IG* delivers the best results on average, with an *importance similarity threshold* t in the interval $[0.99; 0.9999]$. When using correlation, the results show that t should be chosen greater than 0.6. However, the selection of the approach and the parameters' values highly depends on the given dataset, the given data mining task, and the data mining algorithm to be used.

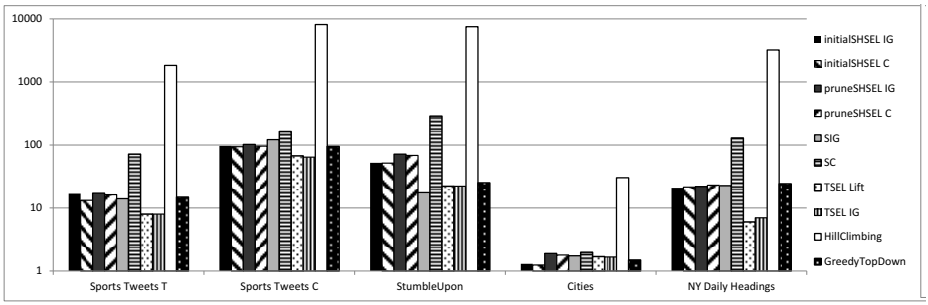


Fig. 3. Runtime (seconds) - Real World Datasets

Table 4. Results on synthetic datasets

	S_D2_B2			S_D2_B5			S_D2_B10			S_D4_B2			S_D4_B5			S_D4_B10		
	NB	KNN	SVM	NB	KNN	SVM	NB	KNN	SVM	NB	KNN	SVM	NB	KNN	SVM	NB	KNN	SVM
Classification Accuracy																		
CompleteFS	.565	.500	.500	.700	.433	.530	.600	.466	.610	.666	.600	.560	.566	.566	.600	.533	.466	.630
initialSHSEL IG	**1.0**	.833	.880	**1.0**	.766	.850	**1.0**	**.866**	.890	**1.0**	**.866**	.880	**1.0**	**.936**	.870	.956	.733	.910
initialSHSEL C	.666	.633	.833	.700	.633	.740	.666	.633	.780	.766	.666	.740	.600	.633	.730	.633	.533	.860
pruneSHSEL IG	**1.0**	**.933**	**.920**	**1.0**	**.800**	**.910**	**1.0**	.833	**.960**	**1.0**	**.866**	**.960**	**1.0**	.866	**.980**	**1.0**	**.800**	**.986**
pruneSHSEL C	.866	.666	.910	.866	.700	.900	.800	.766	.900	.800	.766	.910	.933	.833	.880	.933	.666	.930
SIG	.960	.900	.830	**1.0**	**.800**	.900	.930	.766	.933	**1.0**	.833	.933	**1.0**	.900	.966	**1.0**	.733	.966
SC	.700	.700	.733	.700	.666	.733	.730	.600	.700	.733	.666	.700	.700	.666	.766	.700	.700	.733
TSEL Lift	.553	.500	.540	.633	.666	.630	.400	.500	.540	.566	.533	.540	.500	.566	.510	.466	.533	.480
TSEL IG	.866	.533	.810	.666	.566	.700	.733	.500	.770	.766	.666	.720	.533	.600	.700	.500	.566	.710
HillClimbing	.652	.633	.630	.633	.636	.580	.633	.566	.640	.676	.566	.620	.676	.534	.586	.689	.523	.590
GreedyTopDown	.666	.600	.800	.703	.633	.780	.633	.466	.830	.703	.566	.820	.752	.700	.850	.833	.500	.830
Feature Space Compression																		
initialSHSEL IG	.846	.572	.864	**.948**	.907	.880	.861	.810	**.886**	.914	.789	.740	.929	.868	.746	.912	.750	.918
initialSHSEL C	.267	.557	.875	.104	.888	**.938**	.279	**.956**	.656	.441	.893	.890	.831	.742	.786	.627	.805	.805
pruneSHSEL IG	**.930**	.911	.796	.925	**.933**	.824	**.899**	.944	.850	**.956**	.877	.877	.955	**.969**	.863	.956	.873	.791
pruneSHSEL C	.697	.896	.800	.639	.636	.667	.781	.696	.823	.795	.776	.849	.692	.726	.742	.731	.826	.750
SIG	.922	**.922**	.861	.842	.842	.753	.865	.930	.865	.886	.595	.708	.891	.719	.525	.900	.704	.704
SC	.717	.909	**.880**	.693	.900	.159	.750	.692	.869	.379	.493	.769	.628	.736	.742	.667	.727	.289
TSEL Lift	.750	.750	.750	.706	.706	.706	.687	.687	.687	.857	.857	.857	.827	.827	.827	.814	.814	.814
TSEL IG	.836	.836	.836	.866	.866	.866	.856	.856	.856	.926	**.926**	**.926**	**.965**	.965	**.965**	**.970**	**.970**	**.970**
HillClimbing	.770	.770	.770	.751	.751	.751	.805	.805	.805	.792	.792	.792	.776	.776	.776	.795	.795	.795
GreedyTopDown	.399	.399	.399	.370	.370	.370	.356	.356	.356	.470	.470	.470	.404	.404	.404	.438	.438	.438
Harmonic Mean of Classification Accuracy and Feature Space Compression																		
initialSHSEL IG	.917	.679	.872	.973	.831	.865	.925	.837	.888	.955	.826	.804	.963	.901	.803	.933	.741	**.914**
initialSHSEL C	.381	.592	.853	.182	.739	.827	.394	.762	.713	.560	.763	.808	.697	.683	.757	.630	.641	.832
pruneSHSEL IG	**.964**	**.922**	**.854**	**.961**	**.861**	**.865**	**.946**	**.885**	**.901**	**.977**	**.871**	**.916**	**.977**	**.915**	**.918**	**.977**	**.835**	.878
pruneSHSEL C	.773	.764	.851	.736	.666	.766	.790	.729	.859	.797	.771	.878	.795	.776	.805	.819	.737	.830
SIG	.940	.911	.845	.914	.820	.820	.896	.840	.898	.940	.694	.805	.942	.799	.680	.947	.718	.815
SC	.708	.791	.800	.696	.766	.262	.740	.642	.775	.500	.567	.733	.662	.700	.754	.683	.713	.415
TSEL Lift	.636	.600	.628	.667	.685	.665	.505	.579	.605	.682	.657	.662	.623	.672	.631	.593	.644	.604
TSEL IG	.851	.651	.822	.753	.685	.774	.790	.631	.810	.839	.775	.810	.687	.740	.811	.660	.715	.820
HillClimbing	.706	.695	.693	.687	.689	.654	.709	.665	.713	.730	.660	.695	.723	.633	.668	.738	.631	.677
GreedyTopDown	.499	.479	.533	.485	.467	.502	.456	.404	.499	.564	.514	.598	.526	.513	.548	.574	.467	.573

6 Conclusion and Outlook

In this paper, we have proposed a feature selection method exploiting hierarchic relations between features. It runs in two steps: it first removes redundant features along the hierarchy's paths, and then prunes the remaining set based on

the features' predictive power. Our evaluation has shown that the approach outperforms standard feature selection techniques as well as with recent approaches which use hierarchies.

So far, we have only considered classification problems. A generalizing of the pruning step to tasks other than classification would be an interesting extension. While a variant for regression tasks seems to be rather straight forward, other problems, like association rule mining, clustering, or outlier detection, would probably require entirely different pruning strategies.

Furthermore, we have only regarded simple hierarchies so far. When features are organized in a complex ontology, there are other relations as well, which may be exploited for feature selection. Generalizing the approach to *arbitrary* relations between features is also a relevant direction of future work.

Acknowledgements. The work presented in this paper has been partly funded by the German Research Foundation (DFG) under grant number PA 2373/1-1 (Mine@LOD).

References

1. Blum, A.L., Langley, P.: Selection of relevant features and examples in machine learning. Artificial Intelligence 97, 245–271 (1997)
2. Dash, M., Liu, H.: Feature selection for classification. Intelligent Data Analysis 1(3), 131–156 (1997)
3. Jeong, Y., Myaeng, S.-H.: Feature selection using a semantic hierarchy for event recognition and type classification. In: International Joint Conference on Natural Language Processing (2013)
4. John, G.H., Kohavi, R., Pfleger, K.: Irrelevant features and the subset selection problem. In: ICML 1994, pp. 121–129 (1994)
5. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. Semantic Web Journal (2013)
6. Lu, S., Ye, Y., Tsui, R., Su, H., Rexit, R., Wesaratchakit, S., Liu, X., Hwa, R.: Domain ontology-based feature reduction for high dimensional drug data and its application to 30-day heart failure readmission prediction. In: International Conference on Collaborative Computing (Collaboratecom), pp. 478–484 (2013)
7. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: Dbpedia spotlight: Shedding light on the web of documents. In: Proceedings of the 7th International Conference on Semantic Systems, I-Semantics (2011)
8. Molina, L.C., Belanche, L., Nebot, À.: Feature selection algorithms: A survey and experimental evaluation. In: International Conference on Data Mining (ICDM), pp. 306–313. IEEE (2002)
9. Paulheim, H.: Generating possible interpretations for statistics from linked open data. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 560–574. Springer, Heidelberg (2012)

10. Paulheim, H., Fürnkranz, J.: Unsupervised Generation of Data Mining Features from Linked Open Data. In: International Conference on Web Intelligence, Mining, and Semantics, WIMS 2012 (2012)
11. Paulheim, H., Ristoski, P., Mitichkin, E., Bizer, C.: Data mining with background knowledge from the web. In: RapidMiner World (to appear, 2014)
12. Platt, J.C.: Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, Advances in Kernel Methods - Support Vector Learning (1998)
13. Wang, B.B., Bob Mckay, R.I., Abbass, H.A., Barlow, M.: A comparative study for domain ontology guided feature extraction. In: Australasian Computer Science Conference (2003)