A Wordification Approach to Relational Data Mining: Early Results

Matic Perovšek[1,2], Anže Vavpetič[1,2], Nada Lavrač[1,2,3]

[1] Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia
[2] Jožef Stefan International Postgraduate School, Ljubljana, Slovenia
[3] University of Nova Gorica, Nova Gorica, Slovenia
{matic.perovsek, anze.vavpetic, nada.lavrac}@ijs.si

Abstract. This paper describes a propositionalization technique called *wordification*. Wordification is inspired by text mining and can be seen as a transformation of a relational database into a corpus of documents. As in previous propositionalization methods, after the wordification step any propositional data mining algorithm can be applied. The most notable advantage of the presented technique is greater scalability - the propositionalization step is done in time linear to the number of attributes times the number of examples for one-to-many databases. Furthermore, wordification results in easily understandable propositional feature representation. We present our initial experiments on two real-life datasets.

Keywords: propositionalization, text mining, association rules

1 Introduction

Unlike traditional data mining algorithms, which look for models/patterns in a single table (propositional patterns), relational data mining algorithms look for models/patterns in multiple tables (relational patterns). For most types of propositional models/patterns there are corresponding relational patterns, for example: relational classification rules, relational regression tree, relational association rules, and so on.

For individual-centered relational databases, where there is a clear notion of individual, there exist techniques for transforming such a database into a propositional or single-table format. This transformation, called *propositionalization* [2, 4], can be used by traditional propositional learners, such as decision tree or classification rule learners.

In this paper we introduce a novel propositionalization technique called *wordification*. Wordification is inspired by text mining techniques and can be seen as a transformation of a relational database into a corpus of documents, where each document can be characerized by a set of properties describing the entries of a relational database.

Unlike other propositionalization techniques [2, 3, 4, 6], which search for good (and possibly complex) relational features to construct a subsequent propositional representation, this methodology focuses on a large number of simple

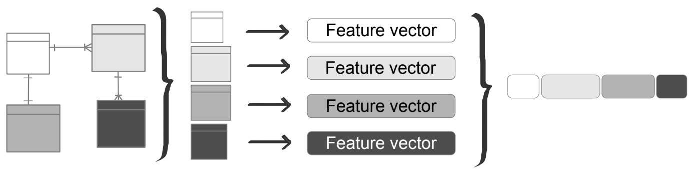

Fig. 1. Document (or feature vector) construction for one individual.

features with the aim of greater scalability. Since the feature construction step is very efficient, it can scale well for large relational databases. In fact, the presented methodology transforms a given database in time linear to the number of attributes times the number of examples for one-to-many databases. Furthermore, due to the simplicity of features, the generated features are easily interpretable by domain experts. On the other hand, this methodology suffers from the loss of information, since the generated features do not explicitly connect relations through variables. Instead, by using the Term Frequency–Inverse Document Frequency (TF-IDF) [1], it tries to capture the importance of a certain feature (attribute value) of a relation in an aggregate manner.

In this paper we report our initial experiments on two real-life relational databases: a collection of best and worst movies from the Internet Movie DataBase (IMBD) and a database of Slovenian traffic accidents.

The rest of the paper is organized as follows. In Section 2 we present the new wordification methodology. Section 3 presents the initial experiments and Section 4 concludes the paper by presenting some ideas for further work.

2 Wordification

This section presents the two main steps of the wordification propositionalization technique. First, a straightforward transformation from a relational database to a textual corpus is performed (Fig.1). One instance (i.e., one entry of the main table) of the initial relational database is transformed into one text document and the features (attribute values), describing the instance, constitute the words of this document. One document is constructed simply as a list of attribute-value pairs - words (or features) constructed as a combination of the table's name and the attribute's name with its discrete value:

$$[table\ name]\_[attribute\ name]\_[attributevalue].$$

Note that every attribute needs to be discretized beforehand in order to be able to represent every value as a word. For each instance, the features are first generated for the main table and then for each entry from the additional tables, and finally concatenated together.

Because we do not explicitly use existential variables in our features, we instead rely on the Term Frequency–Inverse Document Frequency (TF-IDF) measure to implicitly capture the importance of a certain feature for a given instance. In the context of text mining, TF-IDF reflects the representativeness of a certain word (feature) for the given document (instance). In the rest of this section we refer to instances as documents and to features as words.

Table 1. Example input for the standard East-West trains domain.

Train

tid	direction
1	east
2	west

Car

cid	tid	shape	roof	wheels
1	1	rectangle	none	2
2	1	rectangle	peaked	3
3	2	rectangle	none	2
4	2	hexagon	flat	2

Load

lid	cid	shape
1	1	rectangle
2	1	rectangle
3	2	circle
4	3	circle
5	4	circle
6	4	hexagon

1 [car_shape_rectangle, car_roof_none, car_wheels_2, load_shape_rectangle, load_shape_rectangle, car_shape_rectangle, car_roof_peaked, car_wheels_3, load_shape_circle] east
2 [car_shape_rectangle, car_roof_none, car_wheels_2, load_shape_circle, car_shape_hexagon, car_roof_flat, car_wheels_2, load_shape_circle, load_shape_hexagon] west

Fig. 2. The database from Table 1 in document representation.

For a given word w in a document d from corpus D, the TF-IDF is defined as follows: $\text{tfidf}(w, d) = \text{tf}(w, d) \times \log \frac{|D|}{|\{d \in D : w \in d\}|}$, where $tf(\cdot)$ represents the frequency of word w in document d. In other words, a certain word will have a high TF-IDF (i.e., the feature is given a high weight for this instance), if it is frequent within this document (instance) and infrequent in the given corpus (the original database). In other words, the weight of a word gives a strong indication of how relevant is the feature for the given individual. The TF-IDF weights can then be used either for filtering words with low importance or using them directly by the propositional learner.

The technique is illustrated on a simplified version of the well-known East-West trains domain, where the input database consists of two tables shown in Table 1; we have one east-bound and one west-bound train, each with two cars with certain properties. The `Train` table is the main table and the trains are the instances. We want to learn a classifier to determine the direction of an unseen train. For this purpose the class attribute (train direction) is not preprocessed and is only appended to the resulting feature vector of words.

First, the corresponding two documents (one for each train) are generated (Fig. 2). After this, the documents are transformed into a bag-of-words representation by calculating the TF-IDF values for each word of each document, with the class attribute column appended to the transformed bag-of-words table.

```
def wordification(table,individual)
    words=[]
    for att,val in table[individual]:
        words.append(table_att_val)
    #loop through tables which contain current table's foreign key
    for secondary_table in table.secondary_tables():
        words.extend(wordification(secondary_table,individual))
    return
#STEP1
documents={}
for individual in main_table:
    documents[individual]=wordification(main_table,individual)
#STEP2
feature_vectors=tfidf(documents)
results=propositional_learner(feature_vectors)
```

Fig. 3. Pseudo-code of the wordification algorithm.

Table 2. Table properties of the experimental data.

IMDB	#rows	#attributes
movies	166	4
roles	7,738	2
actors	7,118	4
movie_genres	408	2
movie_directors	180	2
directors	130	3
director_genres	243	3

Accidents	#rows	#attributes
accident	102,756	10
person	201,534	10

Table 3. Document properties after applying the wordification methodology.

Domain	Individual	#examples	#words	#words after filtering
IMDB	movie	166	7,453	3,234
Accidents	accident	102,759	186	79

3 Experimental results

This section presents the initial experiments of the wordification methodology. We performed association rule learning in combination with the wordification approach on two real-life datasets: the best and worst ranked IMDB movies database and the Slovenian traffic accidents database. Table 2 lists the characteristics of both databases.

The preprocessing procedure was performed on both databases as follows. First, the wordification step was applied as described in Section 2. Next, irrelevant features (which have the same value across all examples) were removed, resulting in less than half of the features (see Table 3). In order to prepare the data for association rule mining, the data was also binarized: a feature was assigned value *true* if the corresponding TF-IDF value was above 0.06, otherwise *false*.

IMDB database. The complete IMDB database is publicly available in the SQL format[1]. This database contains tables of movies, roles, actors, movie genres, directors, director genres.

The 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[2]. 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 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 [5] FP-growth implementation. Next, association rules for the resulting

[1] `http://www.webstepbook.com/supplements/databases/imdb.sql`
[2] As of July 2, 2012

`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. 4. Examples of interesting association rules discovered in the IMDB database.

frequent itemsets were produced. Among all the discovered rules, several interesting rules were found. Figure 4 presents some of the interesting rules.

The first rule states that if the movie's genre is drama and is directed by Alfred Hitchcock, who is also known for drama movies, then the movie is a good movie. The second rule concludes that if a movie is good and Robert De Niro acts in it, than it must be a drama movie. The third interesting rule shows that Alfred Hitchcock acted only in the movies he also directed. The last rule implies that if Ted Grossman acts in a good adventure movie, then the director is Steven Spielberg (Ted Grossman usually plays the role of a stunt coordinator or performer).

Traffic accident database. The second dataset consists of all accidents that happened in Slovenia's capital city Ljubljana between the years 1995 and 2005. The data is publicly accessible from the national police department's website[3]. The database is multi-relational and consists of the information about the accidents along with all the accidents' participants.

`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. 5. Examples of interesting association rules discovered in the accidents database.

The data already contained discretized attributes, so further discretization was not needed. Similarly as with the IMDB databse, preprocessing using the wordification methodology, FP-growth itemset mining and association rule mining were performed. Figure 3 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.

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

4 Conclusion

This paper presented a novel propositionalization technique called wordification. This methodology is inspired by text mining and can be seen as a transformation of a relational database into a corpus of documents. As is typical for propositionalization methods, any propositional data mining algorithm can be applied after the wordification step. The most notable advantage of the presented technique is greater scalability - the propositionalization step is done in time linear to the number of attributes times the number of examples for one-to-many databases. Moreover, the methodology allows for producing simple, easy to understand features, and consequently, easily understandable rules.

We have presented initial results on two real-life databases: the best and worst movies from the IMDB database and a database of Slovenian traffic accidents. Given that we have found some interesting patterns using our methodology, we are motivated to further explore this approach on new datasets. In future work we will apply the methodology to larger databases to explore its potential limitations. Furthermore, we will experimentally compare this methodology with other known propositionalization techniques.

Acknowledgements

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 and the FP7 European Commission projects MUSE (grant no. 296703) and FIRST (grant no. 257928).

References

[1] Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

[2] Stefan Kramer, Bernhard Pfahringer, and Christoph Helma. Stochastic propositionalization of non-determinate background knowledge. In *Proceedings of the Inductive Logic Programming Workshop*, pages 80–94. Springer, 1998.

[3] Ondřej Kuželka and Filip Železný. Block-wise construction of tree-like relational features with monotone reducibility and redundancy. *Machine Learning*, 83(2):163–192, 2011.

[4] Nada Lavrač, Sašo Džeroski, and Marko Grobelnik. Learning nonrecursive definitions of relations with LINUS. In *EWSL*, pages 265–281, 1991.

[5] Ingo Mierswa, Michael Wurst, Ralf Klinkenberg, Martin Scholz, and Timm Euler. 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)*, pages 935–940, Philadelphia, PA, USA, 20-23 August 2006. ACM Press, NY, USA.

[6] Filip Železný and Nada Lavrač. Propositionalization-based relational subgroup discovery with RSD. *Machine Learning*, 62:33–63, 2006.