Using Data Mining on Linked Open Data for Analyzing E-Procurement Information

Eneldo Loza Mencía and Simon Holthausen and Axel Schulz and Frederik Janssen

Abstract Understanding complex procurement information landscapes and exploring how procurement information can be used to support strategic decision-making is important with the increasing amount of information available in the WWW. In this paper, we cope with this challenge and describe how data mining techniques can be applied on semantically linked data to estimate the number of bidders in public contracts. We introduce a general approach in order to convert linked data in a relational format which can be used by traditional machine learning approaches. Afterwards, we apply common techniques such as discretization, processing of text fields, feature selection, state-of-the-art machine learning algorithms, and more. Alternatives were evaluated and compared. We estimate an accuracy of 32.69% (with a baseline of 30.34%) and costs of 0.37 (baseline: 0.49) for our best configuration, a cost sensitive ensemble of classifiers, for the DMoLD'13 competition.

1 Introduction

As more and more semantical annotated data is available, the interest in extracting valuable information out of this data receives increasing attention. Most classical machine learning approaches rely on so-called attribute-value data meaning that they require certain attributes that have nominal or numerical values assigned. In contrast, semantic data is based on a RDF graph model represented by a triple in form of subject-predicate-object tuples (the classical RDF-notation[1]). Such graphs implicitly are interlinking resources among different datasets. The most popular ex-

Eneldo Loza Mencía, Simon Holthausen, Frederik Janssen
Knowledge Engineering Group, Technische Universität Darmstadt, Germany e-mail: eneldo@ke.tu-darmstadt.de, sholthausen@web.de, janssen@ke.tu-darmstadt.de

Axel Schulz
Telecooperation Lab, Technische Universität Darmstadt, Germany
SAP Research, Darmstadt, Germany e-mail: aschulz@tk.informatik.tu-darmstadt.de

[1] Resource Description Framework, http://www.w3.org/RDF/

ample for such interlinking is the Linked Open Data (LOD) cloud, which, as of today, contains more than 31 billion triples and 500 millions of links between data sets[2]. Especially the increasing amount of Linked Open Government Data (LOGD) from the public sector [1], which contributes about 42% of the LOD cloud, could provide valuable information. When it comes to electronic procurement, which is growing with the information and communications technology, improving procurement processes and reducing costs is important. Thus, there is a need to research and understand complex procurement information landscapes and to explore how such information can be used to support strategic decision-making [6]. However, methods and tools for using that data are still to be developed.

The European Commission frequently publishes information about public procurements [6]. Especially details about tenders are published, which contain information about the organization it originates from, the type of activity etc. Tenders are bidding for public contracts published by the public sector. These contracts have been defined in an ontology[3]. The ontology shows the main features of contracts, such as duration of a contract, estimated price, offered price, time-limits etc. Furthermore, contracts may have special abilities, for instance, if a contract is too big, it is split to lots for which tenders bid separately. Thus, the interconnected nature of contracts is further increased. One important aspect of public procurement is to understand the existing market based on these contracts, with a focus on the number of bidders bidding for a contract, which may be solved using data mining techniques on publicly available data.

To do so, these data mining algorithms either have to be adapted to work with RDF-graphs or by using a different approach namely by converting the semantic data into the classical attribute-value format these techniques typically get as input. In this paper, we employ the second approach as it seems more beneficial to be able to utilize the machine learning algorithms out-of-the-box. Also, previous attempts in similar settings proved to be effective [5].

2 Data Description and Transformation

As a base for our analysis, we use a publicly available RDF dataset provided for the DMoLD challenge[4]. The dataset consists of 1658 *Contracts* for which the numbers of tenders are given. Based on this, our goal is to predict the number of tenders. In the training set, this target value ranges from 0 to 73, but only 36 of the possible values are actually present in the data. The most frequent value is 1, as can be seen in Fig. 1.

Apart from optimizing the prediction accuracy of the model, the costs for misprediction have also been taken into account. The (symmetric) costs for misprediction of a value v by a value $\hat{v}$ are given by

[2] http://lod-cloud.net/state/

[3] http://opendata.cz/public-contracts-ontology

[4] http://keg.vse.cz/dmold2013/

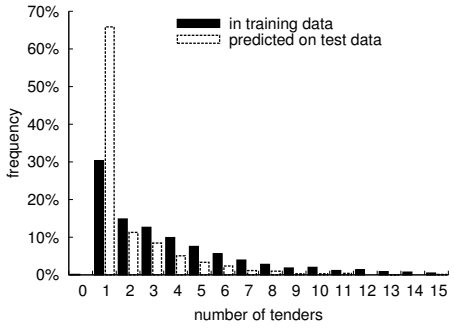

Fig. 1 Distribution of the target variable on the training data and of the predictions on the test data.

$$\mathrm{Err}(v,\hat{v}) = \frac{2}{1+\exp(-\frac{|v-\hat{v}|}{max(1,min(v,\hat{v}))})} - 1 \quad \in [0;1] \tag{1}$$

Thus, as a baseline we have 30.34% accuracy and 0.572 costs per instance by predicting the most frequent value. But better costs are obtained when the value 2 (or 3) is predicted with 0.486 (or 0.487) costs.

Transformation: In the following, we explain how the RDF data is converted into attribute-value data (or tabular data), which is the standard data type which is used by state-of-the-art machine learning algorithms. The starting point of our transformation are *Contract* instances, which are based on publicly available information published by the European Commission. The predicates from these instances are converted into attributes. We use the predicate names in the prefix-notation as attribute names. For every instance, it is checked whether the predicate is set and if this is the case, the object of the predicate is adopted as attribute value. If not, we set the instance's attribute to missing value. We iterate over all Contracts to find all existing predicates of Contract in a particular RDF graph. The result is a dataset where each Contract entity (a row) is described by a set of attributes (columns) and corresponding values (cells).

As not all attributes in the dataset are of type text string, the appropriate types have to be determined. Predicates in the RDF-file that point to a literal of type *xsd:int* are converted into numeric attributes. Nominal attributes are found after the discovery of new attributes and the iteration over all Contracts is finished. Since predicates have no information if its objects are of a closed value range, we apply a simple heuristic, which worked good for us:

- If the attribute-value of all instances has less than 20 characters, it is a nominal candidate.
- If the attribute is a nominal candidate and the number of distinct values of all instances is less than 30 percent, it is a nominal attribute.

Furthermore, many attribute values have only one distinct value, for example a Contract is always of *rdf:type* Contract. We also introduce a special nominal attribute indicating the month of a literal of type *xsd:date*.

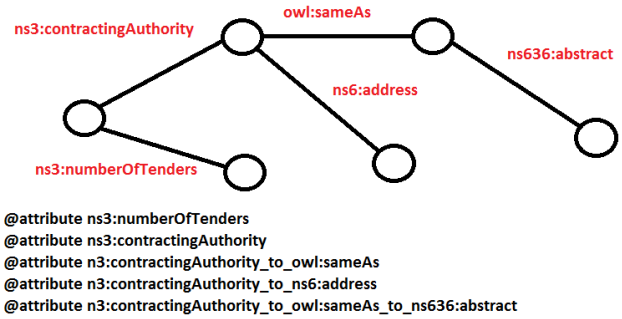

Fig. 2 Transformation of predicates into attributes. Starting from the initial contract node at the left, the predicates are followed recursively and each path is added as a new attribute (shown in the bottom and with the name prefixed with @*attribute*) with the corresponding object as value.

Tree traversal: So far, we only looked at predicates of Contracts. Many of those point to URIs that can be resolved to resources in the RDF-file which yield potential important information. We want to include the values of the predicates of those resources into our tabular dataset. For this task, we introduce a recursive algorithm:

1. Get value of predicate
2. If value is an URI, go to 3, else look at next predicate and go to 1
3. Resolve URI to resource, get all predicates, apply step 1 on all of them

To show that predicates are found following the URI of another predicate they are named in the following way: *predicateWeCameFrom_to_predicateWeGoTo*. Figure 2 illustrates this.

Known issues: There are a few problems we did not solve during the transformation of the RDF file.

1. Predicates with multiple objects are not covered yet. Because the order in which multiple objects appear can differ between values of the same predicate it does not make sense to convert them (for instance, into *ns5:troupStrength1*, *ns5:troupStrength2*, etc.). Instead, we take the first occurring object of a predicate. If the attribute we created is of type text string, we choose the property that is in English (in other words, annotated with @*en*).
2. Literals are not required to have a certain type, even if the predicate name and the majority of its values suggest a certain type. For example, the majority of the predicate *dbpprop:size* suggests that its literal type is *xsd:int*. Indeed, most literals are of that type. But there are some exceptions, for example in an instance of type *yago:MajorCommandsOfTheUnitedStatesAirForce*, the value of *dbpprop:size* is "Nearly 57,000 personnel"@en. In that case, we first try to convert the value into a numeric value, and if that fails, we set the value to missing.
3. We did not resolve *sameAs* predicates and did not merge the resources because this results in more properties having multiple values. A property could have been set in the original resource as well as the *sameAs*-resource, which results in the same problem as in point 1.

3 Preprocessing and Experiments

We used the Weka package [7] for all experiments, which is a very popular machine learning framework including many state-of-the-art algorithms. We could have used a different framework but since we are primarily interested in applying existing algorithms rather than implementing new ones this choice was arbitrary. If not stated otherwise, we used the C4.5 decision tree implementation, in Weka referred to as J48, with default settings and sometimes in the unpruned version, the Ripper rule learner implementation JRip with defaults, k-NN with k selected based on an internal cross-validation and leaving the remaining parameters untouched, LibSVM [2] with RBF kernel, and LibLinear [4] in default configuration. Please see the documentation in [7] for further details on the used approaches.

In the following, the general proceeding in our experimentation is shown. However, as we used an explorative approach in order to find a good working combination of approaches, decisions were not necessarily taken in this order. Table 1 shows the most important and interesting results we obtained in our series of experiments. The next sections discuss our decisions which lead to our final approach.

Class variable: Roughly speaking, there are two possible ways to treat the class variable: either considering it as a numeric attribute and treat the problem as a regression problem, or by discretizing the value to obtain a classification problem. Since the data becomes very sparse for higher number of tenders and most of the instances have a rather low number of tenders, we chose the second approach. Furthermore, we wanted to take advantage of the accurate probability estimations many classification models produce in order to optimize the predictions with respect to minimal costs.

Size of the graph: If we restrict the transformation to only direct predicates to the *Contract* elements, we obtain 3 nominal, 5 text (string), and 3 date attributes. We therefore expand the traversing to 4 hops, which resulted in 387 numeric and nominal and 797 string attributes.

Missing values and nominal attributes: Symbolic learning algorithms, such as J48 and JRip, are inherently able to process missing values and non-binary nominal attributes. Statistical approaches, such as SVMs, which work on a real valued feature space, have commonly more difficulties to handle such type of attributes and often require a transformation into numeric values. We use straight-forward approaches for both cases, as we will see in the following.

Missing nominal values are replaced by an additional value indicating that the value is missing. This is a simple but effective solution [8]. Our pragmatic assumption for replacing missing numeric values is that predicates are left out in the RDF data if the values are zero, so that we consequently replace these by zero. This assumption will hardly hold for most of our numeric attributes, since it is more reasonable that the value is unknown or the numeric predicate is just not applicable. Though we do not estimate great improvements, we find it worth to try more differentiating approaches at this step in future experiments.

An effective method for transforming nominal into numeric values for statistical classifiers is binarization, i.e., the addition of binary attributes (0 or 1) for each

possible value of a nominal attribute. This approach can also be advantageous for symbolic approaches if an attribute has a high number of possible values. J48 for instance creates one branch for each possible value in the learned decision tree, even if some values were hardly seen or the differentiation does not contribute to better discrimination. This increases the model size and reduces accuracy. Hence, we often just experimented with all algorithms on the binary data.

Processing of text: Though the dataset contains a lot of formalized knowledge in form of RDF triples, it also contains much information given implicitly in form of text. We tried to exploit this additional information by applying simple text processing approaches on the strings in the RDF files which come from description of the contracts and from Wikipedia. Hence, we applied tokenization, stop word removal, and stemming on the text data. This was also applied on URIs which were provided as text attributes. The terms were weighted by the TF-IDF measure. The document frequency was computed on the training and test set as a way of exploiting the unlabeled data. These steps resulted in 53,852 new attributes.

Though the data is sparse, even SVMs need long time to process this amount of data. The best SVM configuration in terms of accuracy (31.06%) needed 34 minutes and the best ones for costs (around 0.42) needed around one hour (k-NN needed 15 min. with comparable costs). In addition, it is well known that especially very frequent or very infrequent terms are also very noisy and are harmful to the performance, and that reducing the number of attributes has hence a positive effect [10]. This is confirmed in our experiments as the next subsection shows.

Feature Selection: As mentioned above, feature selection can increase the predictive quality of a model and additionally reduce the computational costs. The used approaches measure the dependency of an attribute with the class attribute in different but similar ways, namely according to χ^2 statistic, information gain, and the related gain ratio. The two former approaches resulted in a small set of only 14 relevant features. This strongly indicates that there is hardly useful attributes contained in the data set, which is somehow confirmed by the small margin to the baseline.

Interestingly, the most useful attributes according to the employed methods are text features, namely *postal-addres*, *build* and *sourc*. Intuitively, it is difficult to believe that these attributes may transport useful information. The only nominal attributes present are *services* and *supplies*. At the same time it becomes clear that a binary attribute is not sufficient in order to discriminate 36 classes.

A quick analysis with RelieF [7], a popular feature weighting algorithm, revealed quite different top relevant attributes, and we believe that combining more approaches like Principal Component Analysis, k-Means stacking, etc. would further help to increase diversity and find complementary attributes.

Cost Sensitiveness and Ensembles: The results obtained applying the previous steps often show a contrary behavior with respect to accuracy and classification costs. J48, e.g., is worse than the baseline regarding accuracy (29.98%), but is one of the best approaches for cost minimization (0.407). On the other extreme are SVMs with 32.03% but 0.499 costs (cf. Table 1).

It seems clear that it is generally not possible to simultaneously minimize the classification error and classification costs, since classification algorithms are

hops	missings	nominals	TF-IDF	features	learner	accuracy	costs
–	–	–	–	–	baseline	30.34	0.486
0	y	y	n	9	J48 unpruned	28.35	0.449
0	y	y	n	9	JRip	30.46	0.568
0	n	n	n	619	J48	29.86	0.428
0	n	n	n	619	LibSVM RBF C=10^1	30.40	0.572
4	n	n	y	58235	LibLinear C=10^{-3}	28.53	0.422
4	n	n	y	58235	k-NN	29.55	0.443
4	n	n	y	58235	LibSVM RBF C=10^2	31.18	0.516
4	n	n	y	38	**CS Ensemble**	32.09	**0.370**
4	n	n	y	38	CS pairw. J48	29.19	0.371
4	n	n	y	38	CS k-NN	27.14	0.382
4	n	n	y	38	pairw . J48	31.48	0.402
4	n	n	y	38	CS J48	29,31	0.406
4	n	n	y	38	Ensemble	**32.69**	0.402
4	n	n	y	38	J48	29.98	0.407
4	n	n	y	38	LibSVM C=10^3	30.70	0.409
4	n	n	y	38	LibSVM C=10^2	32.09	0.418
4	n	n	y	38	LibLinear C=10^{-3}	31.48	0.458
4	n	n	y	38	LibSVM C=1	32.03	0.499

Table 1 Results on the different versions of the data using 10 fold cross validation. In each block the best approaches with respect to accuracy and costs are shown, adding comparative approaches which were successful in the previous blocks or were interesting. For each dataset version, it is indicated the number of hops, whether missing and nominal values were present, whether strings were included and how many features were used (the last block is with feature selection). Cost sensitive prediction and pairwise coupling approaches are indicated with *CS* and *pairw.*, respectively.

trained with a static cost structure which is quite different from the cost matrix in our particular case.

In order to tackle this, we minimize a posteriori the expected costs [3]. This expects a (good) class probability estimator as classifier. Given the posterior class probabilities $p_i = P(v_i|x)$ for an instance x, the Bayes optimal prediction is

$$\arg\max_{v_j} \sum_i \mathrm{Err}(v_i, v_j) \cdot p_i \tag{2}$$

When applying this additional step on the predictions of a classifier, k-NN, e.g., is able to improve from 0.4289 to 0.3817, while the accuracy also substantially decreases from 31.42% to 27.14%. On the other hand, the results for J48 almost do not change.

The largest improvement by a single measure was achieved when using pairwise coupling and the J48 base learner, an extension of pairwise classification for obtaining accurate probabilities [9]. Another way to obtain good probabilities is to average the estimations of several classifiers. In order to avoid a bias towards a specific algorithm (type), we set up such an ensemble based on good cost minimizing configurations from each learner class. This resulted in a combination of both pruned and unpruned J48, the pairwise coupling version of J48, LibLinear with $C = 10^{-1}$, LibSVM with $C = 10^3$ and RBF kernel, and k-NN. This ensemble hardly achieved

better costs than J48 with pairwise coupling, but interestingly the accuracy improved substantially until reaching the second position, only beaten by the same ensemble but without the cost sensitive selection.

Best approach and submission to competition: The cost sensitive ensemble discussed in the previous section obtained the best overall results (bold approach in Tab. 1) and was hence selected to produce the predictions on the unlabeled test set. Fig. 1 shows the distribution of the predicted number of tenders in comparison to the original distribution on the training data. We can observe a clear divergence, which results from the general uncertainty in the predictions and hence the cost minimization in the predictions.

4 Conclusions

This paper can be seen as a first step towards high-performance machine learning on semantically linked data. We have identified several problems in the transformation of the data in relational format as well as in the post-processing. However, we have only provided a glance into possible solutions, more work needs to be done in this direction. The most effective method in our setting was to use classifier ensembles and to post-process the predictions according to the expected costs. On the task of the competition, we estimate an accuracy of 32.69% and costs of 0.37. On the other hand, we believe that there is still potential in the filtering of the features. Nevertheless, our solution is explicit for the provided dataset and no general guidelines can be given yet. The developed methods have to be still explored on other types of linked data.

References

[1] Bizer, C.: The emerging web of linked data. IEEE Intelligent Systems **24**(5), 87–92 (2009)
[2] Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001). Manual
[3] Domingos, P.: Metacost: a general method for making classifiers cost-sensitive. In: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 155–164 (1999)
[4] Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: Liblinear: A library for large linear classification. Journal of Machine Learning Research **9**, 1871–1874 (2008)
[5] Paulheim, H., Fürnkranz, J.: Unsupervised generation of data mining features from linked open data. In: International Conference on Web Intelligence and Semantics (2012)
[6] Valle, F., d'Aquin, M., Di Noia, T., Motta, E.: LOTED: Exploiting Linked Data in Analyzing European Procurement Notices. In: Proceedings of the 1st Workshop on Knowledge Injection into and Extraction from Linked Data (KIELD 2010) (2010)
[7] Witten, I.H., Frank, E.: Data mining: Practical machine learning tools and techniques (2005)
[8] Wohlrab, L., Fürnkranz, J.: A review and comparison of strategies for handling missing values in separate-and-conquer rule learning. J. Intell. Inf. Syst. **36**(1), 73–98 (2011)
[9] Wu, T.F., Lin, C.J., Weng, R.C.: Probability estimates for multi-class classification by pairwise coupling. Journal of Machine Learning Research **5**, 975–1005 (2004)
[10] Yang, Y., Pedersen, J.O.: A comparative study on feature selection in text categorization. In: Proceedings of the 14th International Conference on Machine Learning, pp. 412–420 (1997)