Graph Kernels for Task 1 and 2 of the Linked Data Data-Mining Challenge 2013

Gerben Klaas Dirk de Vries

System and Network Engineering Group, Informatics Institute, University of Amsterdam, The Netherlands
g.k.d.devries@uva.nl

Abstract. In this paper we present the application of two RDF graph kernels to task 1 and 2 of the linked data data-mining challenge. Both graph kernels use term vectors to handle RDF literals. Based on experiments with the task data, we use the Weisfeiler-Lehman RDF graph kernel for task 1 and the intersection path tree kernel for task 2 in our final classifiers for the challenge. Applying these graph kernels is very straightforward and requires (almost) no preprocessing of the data.

1 Introduction

Kernel methods [1] are a popular solution to learning from structured data. Graph kernels provide an interesting approach to learning from RDF. Currently there exists some research on this topic [2, 3]. The main advantage of using graph kernels on RDF data is that the technique is very generic and can be widely applied without much preprocessing and domain knowledge of the datasets involved.

In this paper we present the application of two RDF graph kernel methods in two data-mining from RDF tasks. Both tasks are part of the linked data data-mining challenge[1], which is part of the 2013 Data-Mining on Linked Data (DMoLD) workshop. We apply these techniques with very little preprocessing of the task data.

In the rest of this paper we first briefly describe the algorithms used, which are both more elaborately described in other papers. Then we discuss some experiments with the data, which lead to classifiers used for the two tasks. We end with some conclusions.

2 Algorithms

The graph kernel approach to learning from RDF is based on the idea that RDF instances are represented by their subgraphs. By computing a kernel function on the subgraphs and then training a classifier, i.e. a support vector machine, we can predict properties for the instances.

[1] http://keg.vse.cz/dmold2013/data-description.html

For tasks 1 and 2 of the challenge we used two graph kernel algorithms. The first algorithm is an extension of the Weisfeiler-Lehman RDF (WL RDF) kernel presented in [3]. The second kernel is called the Intersection Tree Path (ITP) kernel and is presented in an accompanying paper in the DMoLD workshop [4].

The WL RDF kernel computes the number of subtrees shared between two graphs by using the Weisfeiler-Lehman test of graph isomorphism. The rewriting procedure underlying the Weisfeiler-Lehman kernel creates a multiset label for each vertex/edge, based on the labels of the neighbors of that vertex/edge. This multiset is sorted and together with the original label concatenated into a string, which is the new label. For each unique string a new (shorter) label is introduced and this replaces the original vertex label. Based on the counts of the different labels a feature vector is constructed for each instance.

The intersection tree path kernel counts all the paths starting from the instance vertex up to depth d and creates a feature vector with these counts. It is similar to the intersection subtree kernel in [2], but is a lot faster to compute.

Both algorithms that we use are extended to handle RDF literals using a bag-of-words approach, by creating a term vector for each literal. Two term vectors are compared when the vertex label, for WL RDF, or the path to the vertex, for ITP, are equal. More details can be found in [4]. When working with text and term vectors it is standard to normalize term vectors by converting them to Term Frequency-Inverse Document Frequency (TF-IDF) vectors. We also apply this normalization to our computed feature vectors, for both the WL RDF kernel and the intersection path tree kernel.

3 Experiments

In the following section we present our experiments with the Weisfeiler-Lehman RDF with bag-of-words kernel and the intersection tree path with bag-of-words kernel on task 1 and 2 of the linked data data-mining challenge. First we discuss some preprocessing that we did and then we present the results on the two tasks. During our investigation, some other graph kernels were also applied to the data, but we do not cover those here, since the two kernels discussed in this paper showed the best results.

All of the code for the experiments was written in Java and is available online.[2] For our classification algorithm we used the Java version of LibLINEAR [5] and we used SESAME[3] to handle RDF data and add extra inferenced triples using its RDFS reasoner.

3.1 Preprocessing

We have applied relatively little preprocessing to the provided train and test sets. In all the datasets we had to correct the literals of the type: `xsd:gYear`,

[2] `https://github.com/Data2Semantics/d2s-tools`
[3] `http://www.openrdf.org/`

since Sesame did not parse them properly. This correction was simple and only involved changing the full date string to just the year.

For task 1 we removed instances for which the label occurred less than 5 times, to remove outliers. The numberOfTenders values for the remaining instances were then binned in the following bins: $[0.5, 1.5), [1.5, 2.5), [2.5, 3.5), [3.5, 4.5)$, $[4.5, 5.5)$,$[5.5, 6.5)$, $[6.5, 7.5)$,$[7.5, 8.5)$, $[8.5, 9.5)$, $[9.5, 12.5)$, $[12.5, 15.5)$, $[15.5, 18.5)$, $[18.5, 23.5]$. The size of the bins follows the distribution of the values of the numberOfTenders property. We chose to introduce binning so that next to a regression algorithm we can also use a classification algorithm.

We experimented with loading additional data from DBpedia via the sameAs relations provided in the datasets. However, this did not influence performance. Considering the nature of our graph kernel approach, this is somewhat to be expected, since the links to DBpedia only add more general knowledge to the graph instead of more specific knowledge.

3.2 Task 1

For the first task we used the default SVC classification and SVR regression algorithms in LibLINEAR. The classification algorithm was used with the binned version of the numberOfTenders property as classes. For the regression algorithm we used the numberOfTenders value directly. The training dataset was split into 80% data for training and 20% for testing. The part for training was again split in 80% for training and 20% for validation, in order to optimize the parameters of the algorithms. These splits were repeated 10 times. For classification and regression we optimized the C parameter from $10^{-4}, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10^4$ and in regression we also optimized the p parameter from $10^{-6}, 10^{-5}, 10^{-4}$, $0.001, 0.01$. During optimization the evaluation function given for this task[4] (T1) was used to test the trained model. In the classification case we used the average of a bin as prediction for the numberOfTenders. Before training the numberOfTenders property was removed from the data. We test three extraction depths for the subgraphs: $1, 2, 3$.

Results for this experiment are given in Table 1. The T1 binned scores are the evaluation function scores for the classification algorithm. The other three scores are for the regression algorithm, which also include the Mean Squared Error (MSE) and Mean Absolute Error (MAE). The scores in bold are the scores for the kernel and settings that we used in the final classifier.

The scores for the different kernels and settings do not differ much. For our final classifier we chose the WL RDF with bag-of-words kernel, with depth 2 and $h = 2$, since this showed good results for all 4 scores. For our final classifier, which we used on the challenge test set, we used the binned version of the numberOfTenders property.

[4] `http://keg.vse.cz/dmold2013/data-description.html`

Table 1. Results for the task 1 experiments.

depth	T1 binned	T1	MSE	MAE	T1 binned	T1	MSE	MAE
	Weisfeiler-Lehman RDF with Bag-of-Words							
	$h = 0$				$h = 2$			
1	1.38	1.41	11.50	2.32	1.37	1.40	11.51	2.31
2	1.38	1.41	11.23	2.32	**1.37**	**1.40**	**11.32**	**2.29**
3	1.38	1.40	11.22	2.28	1.38	1.40	11.19	2.29
	$h = 4$				$h = 6$			
1	1.37	1.40	11.25	2.31	1.38	1.40	11.48	2.30
2	1.38	1.41	11.26	2.30	1.38	1.40	11.64	2.31
3	1.38	1.40	12.06	2.39	1.38	1.41	11.79	2.34
	Intersection Tree Path with Bag-of-Words							
1	1.38	1.41	11.6	2.35				
2	1.37	1.41	11.51	2.30				
3	1.39	1.42	11.84	2.35				

3.3 Task 2

Task 2 is a binary classification problem and we used the default SVC classification algorithm in LibLINEAR to train a classifier. Since this dataset is smaller than the task 2 dataset we used a cross-validation setup for the experiments. Per kernel we did a 10-fold cross-validation which was repeated 10 times. Within each fold the C parameter was optimized by doing 10-fold cross-validation. Before training the multicontracts property was removed from the data.

Table 2 presents the results for this experiment. We provide the accuracy and F1 scores. Again, the bold scores indicate the kernel and settings that we used for the final classifier. The intersection tree path kernel achieves better scores than the WL RDF kernel (especially F1). The baseline accuracy is 0.81 and F1 is 0.50. Given the scores that we achieved, we can conclude that this task is difficult.

4 Conclusion

We have presented the application of Weisfeiler-Lehman RDF and intersection path tree kernels, both extended with bag-of-words term vector for the literals, to task 1 and 2 of the linked data data-mining challenge. For task 1 the final classifier was trained using the WL RDF with bag-of-words kernel with the values of the numberOfTenders property binned into 13 bins. The final classifier for task 2 was trained with the intersection tree path with bag-of-words kernel. The application of both kernels in the two tasks was very straightforward. The most complicated preprocessing step was the binning of the numberOfTenders values.

Table 2. Results for the task 2 experiments.

depth	acc.	F1	acc.	F1	acc.	F1	acc.	F1
	Weisfeiler-Lehman RDF with Bag-of-Words							
	$h = 0$		$h = 2$		$h = 4$		$h = 6$	
1	0.83	0.53	0.82	0.50	0.82	0.50	0.83	0.51
2	0.84	0.57	0.84	0.57	0.84	0.57	0.83	0.56
3	0.84	0.58	0.81	0.56	0.79	0.53	0.79	0.54
	Intersection Tree Path with Bag-of-Words							
1	0.82	0.50						
2	**0.85**	**0.60**						
3	0.85	0.60						

Acknowledgments This publication was supported by the Dutch national program COMMIT.

References

1. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, New York, NY, USA (2004)
2. Lösch, U., Bloehdorn, S., Rettinger, A.: Graph kernels for rdf data. In Simperl, E., Cimiano, P., Polleres, A., Corcho, Ó., Presutti, V., eds.: ESWC. Volume 7295 of Lecture Notes in Computer Science., Springer (2012) 134–148
3. de Vries, G.K.D.: A fast approximation of the weisfeiler-lehman graph kernel for rdf data. In Blockeel, H., Kersting, K., Nijssen, S., Zelezný, F., eds.: ECML/PKDD. (2013)
4. de Vries, G.K.D., de Rooij, S.: A fast and simple graph kernel for rdf. In: DMoLD. (2013)
5. 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** (2008) 1871–1874