

Task Instance Classification via Graph Kernels

Mark Kröll, Andreas Rath, Nicolas Weber, Stefanie Lindstaedt, and Michael Granitzer

Know-Center GmbH, Inffeldgasse 21a/II, A-8010 Graz, Austria
mkroell, arath, nweber, slind, mgrani@know-center.at

Knowledge-intensive work plays an increasingly important role in organisations of all types. Knowledge workers contribute their effort to achieve a common purpose, they are part of (business) processes. However, techniques to provide support such as guidance or intelligent resource delivery face a lot of challenges due to the emergence of richly structured, heterogeneous datasets. Mining these datasets by application of traditional methods can lead to inappropriate conclusions about the data. Thus, to provide continuative assistance of knowledge workers graph mining techniques, that take relationships among graph objects into account, are applied. In the scope of the research project DYONIPOS, graph kernels in combination with support vector machines enable us to exploit structural information where content information is of no use.


1 Motivation

Workflow Management Systems (WFMS) have become quite popular in organizations, because they promise to solve the problems arising from complex organizational processes. This significant contribution of WFMS to increase the productivity is generally accepted [11], but WFMS still impose too many restrictions on knowledge workers [13]. Knowledge workers are guided by goals instead of tasks and prefer significant freedom in structuring their own activities. The DYONIPOS research project [12] strives to slacken the constraints by admitting more flexibility in the work process. This additional freedom enforces a strict monitoring of all user actions in order to derive the knowledge worker's context [10]. The extracted information can be used to guide the knowledge worker through the work process and to intelligently deliver useful resources. However, to provide guidance in the form of propositions of the next step or of helpful tutorials to read, the current process instance the user resides in has to be determined. Information that is based on the content of documents that have been read or written lately might not lead to conclusive results, since process instances of a common process¹ show great variations regarding their context. Hence, structural information is sometimes the only hint for categorizing task instances to tasks and process instances to processes. The paper is structured as follows: Section 2 discusses related work in this area, Section 3 describes the information gathering, the representation of the data and

¹ The process *Write a paper* possesses many derivations, e.g., concerning different research areas as computer science, medical science and so on.

the application of graph kernels for the classification task. The paper closes with concluding remarks and the list of references.



2 Related Work

Relations between graph entities provide valuable information for mining tasks. Thus, the relative ascent research areas *Link Mining* [3] and *Graph Mining* [15] successfully exploit the topological view of structured data. Haussler [4] introduced convolution kernels, a general framework for handling discrete data structures by kernel methods. Kashima et al. [6, 7] concentrated on the construction of graph kernels. Their graph kernel performs a random walk on the vertex product graph of two graphs. The idea behind this kernel is including local information, i.e., taking into account similar edges and vertices of the vicinity.

Many areas of knowledge such as bioinformatics, computational chemistry and computer vision have recently been using kernel methods and graph kernels in order to learn from structured data. One of the main applications of kernels on graph spaces is in structure classification. Chemical compounds are often represented as graphs and, therefore, graph kernels are useful to process such structures[8]. The work of Ralaivola et al.[9] presents kernels based on the idea of molecular fingerprints and can be viewed as spectral kernels derived by counting and comparing walks in the underlying graph. Vert [14] discusses the applications of kernel methods in bioinformatics ranging from the classification of tumors to the automatic annotation of proteins.

Gärtner et al. [2] use graph kernels as the covariance function between state-action pairs in a relational reinforcement learning environment. Thus, by modeling block world scenarios as graphs and applying kernel methods, a strategy in the well-known game *Tetris* can be learnt.

3 Structural Categorization of Task Instances

This section describes the gathering of the data, how the collected data is further processed, i.e. which components flow into the graph representation, and how graph kernels combination with SVMs are utilized for task instance classification.

User interactions with the system and reactions from the system to the users interactions are monitored and written into log files. These interactions represent events. Events are user inputs, such as mouse movements, mouse clicks, starting a program, creating a folder, a web search, or opening a file. A sequence of events are grouped together into blocks of events by using predefined static rules, which map a set of events to an event block. An example of an event block is “editing a document on page 2”. In this event block all events that occur from key strokes in a text editing application are aggregated and form the event block. A task instance, i.e., the execution of a predefined task, is formed by combining event

blocks that are related. Grouping together event blocks into tasks is done by calculating similarities based on common content, e.g. the bag of words approach. However, the assignment of task instances to predefined tasks cannot be based on content similarity any more. The content of two task instances belonging to the same task can vary arbitrarily, e.g., think of writing a paper in different scientific fields. Thus, a classification of task instances needs to resort to structural information, i.e., which types of event blocks occur, which resources have been used or which persons have been contacted.

The structural information of a task instance is represented as graph. Types of event blocks, used resources and contacted experts form graph nodes. The used application and the corresponding user behaviour, i.e., reading or writing, define the type of an event block. An exemplary event block type can be characterized by following tuple {Winword, Read}. Resources are differentiated the way user received them. Have resources been actively requested or have they been offered automatically as support? Based on that information an adjacency matrix of a task instance can be established.

The assignment of a task instance to a predefined task is a multiclass classification that can be approached by segregating it into binary classification problems. Kernel based methods, such as Support Vector Machines (SVMs), have proven to be quite effective in the area of supervised learning. One advantage of kernel methods lies in their modularity, i.e., various kernels can be used to define the algorithm's view on the data. Thus, a self-calculated (graph) kernel can be plugged into a kernel machine without the need to adapt the algorithm's procedure. We are using the LibSvm implementation [1] which allows the handing over of a precomputed kernel.

The graph kernel is computed in a similar way as described in [2]. The adjacency matrices of two task instances are transformed into a direct product graph [5]. The definition of graph kernel in use reads as follows:

Definition 1 Let G_1, G_2 be two graphs, E_\times denotes the adjacency matrix of their direct product $E_\times = E(G_1 \times G_2)$, and V_\times denotes the vertex set of the direct product $V_\times = V(G_1 \times G_2)$. With a sequence of weights $\lambda = \lambda_0, \lambda_1, \dots$ ($\lambda_i \in \mathcal{R}; \lambda_i \geq 0$ for all $i \in \mathcal{N}$) the product graph kernel is defined as

$$k_\times(G_1, G_2) = \sum_{i,j=1}^{|V_\times|} \left[\sum_{n=0}^{\infty} \lambda_i E_\times^n \right] \quad (1)$$

if the limit exists.

The graph kernel is computed by calculating the limit of the matrix power series in Definition 1. Equation 2 shows the dual problem of computing the limit of Eigenvalues of the adjacency matrix of the direct product graph.

$$\lim_{n \rightarrow \infty} \sum_{i=0}^n \lambda_i E^i = T^{-1} \left(\lim_{n \rightarrow \infty} \sum_{i=0}^n \lambda_i D^i \right) T \quad (2)$$



where T denotes the matrix of Eigenvectors and D the matrix of Eigenvalues.




4 Concluding Remarks

In this extended abstract we presented our approach for categorizing task instances to predefined tasks. The classification is based on structural information that is represented as graph. Similarities between graphs are calculated by applying graph kernels. First experiments with small sized graphs have proven to be promising. The evaluation of results so far is not significant lacking acceptable training and test datasets. We intend to enlarge the datasets and continue our efforts in that research direction.

5 Acknowledgements

The project results have been developed in the DYONIPOS project (DYnamic ONtology based Integrated Process OptimiSation). DYONIPOS is financed by the Austrian Research Promotion Agency (www.ffg.at) within the strategic objective FIT-IT under the project contract number 810804/9338.

The Know-C  is funded by the Austrian Competence Center program Kplus under the auspices of the Austrian Ministry of Transport, Innovation and Technology (<http://www.ffg.at>), by the State of Styria and by the City of Graz.

References

1. C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
2. T. Gärtner, K. Driessens, and J. Ramon. Graph kernels and Gaussian processes for relational reinforcement learning. volume 2835, pages 146–163, 2003.
3. L. Getoor. Link mining: a new data mining challenge. *SIGKDD Explor. Newsl.*, 5(1):84–89, 2003.
4. D. Haussler. Convolution kernels on discrete structures, 1999.
5. K. S. Imrich, W. *Product Graphs: Structure and Recognition*. John Wiley, 2000.
6. H. Kashima and A. Inokuchi. Kernels for graph classification. ICDM Workshop on Active Mining 2002, 2002.
7. H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the 20th International Conference on Machine Learning*, 2003.
8. T. A. J.-L. P. Pierre Mah, Nobuhisa Ueda and J.-P. Vert. Graph kernels for molecular structure, 2005.
9. L. Ralaivola, S. J. Swamidass, H. Saigo, and P. Baldi. Graph kernels for chemical informatics. *Neural Netw.*, 18(8):1093–1110, 2005.
10. A. Rath, M. Kröll, S. Lindstaedt, and M. Granitzer. Low-level event relationship discovery for knowledge work support. In N. Gronau, editor, *Low-Level Event Relationship Discovery for Knowledge Work Support*, volume n/a, pages 227 – 234, Berlin, Mar 2007. GITO-Verlag. ProKW2007 Productive Knowledge Work : Management and Technological Challenges.

11. U. V. Riss. Knowledge, action, and context: A process view on knowledge management. In *Wissensmanagement*, pages 555–558, 2005.
12. K. Tochtermann, D. Reisinger, M. Granitzer, and S. Lindstaedt. Integrating ad hoc processes and standard processes in public administrations. In *Proceedings of the OCG eGovernment Conference, Linz (Austria)*, 2006.
13. W. van der Aalst and M. Weske. Case handling: A new paradigm for business process support. *Data Knowl. Eng.*, 53(2):129–162, 2005.
14. J.-P. Vert. Kernel methods in genomics and computational biology, 2005.
15. T. Washio and H. Motoda. State of the art of graph-based data mining. *SIGKDD Explor. Newsl.*, 5(1):59–68, 2003.