kLog: A Language for Logical and Relational Learning with Kernels (Extended Abstract)*

Paolo Frasconi
Università degli Studi di Firenze, Italy
p-f@dsi.unifi.it

Fabrizio Costa
Albert-Ludwigs-Universität, Freiburg, Germany
costa@informatik.uni-freiburg.de

Luc De Raedt
KU Leuven, Belgium
luc.deraedt@cs.kuleuven.be

Kurt De Grave
KU Leuven, Belgium
kurt.degrave@cs.kuleuven.be

Abstract

We introduce kLog, a novel language for *kernel-based learning* on expressive logical and relational representations. kLog allows users to specify logical and relational learning problems declaratively. It builds on simple but powerful concepts: learning from interpretations, entity/relationship data modeling, and logic programming. Access by the kernel to the rich representation is mediated by a technique we call *graphicalization*: the relational representation is first transformed into a graph — in particular, a grounded entity/relationship diagram. Subsequently, a choice of graph kernel defines the feature space. The kLog framework can be applied to tackle the same range of tasks that has made statistical relational learning so popular, including classification, regression, multitask learning, and collective classification. An empirical evaluation shows that kLog can be either more accurate, or much faster at the same level of accuracy, than Tilde and Alchemy. kLog is GPLv3 licensed and is available at http://klog.dinfo.unifi.it along with tutorials.

1 Introduction

kLog is embedded in Prolog (hence the name) and allows users to specify different types of logical and relational learning problems in a declarative way. kLog adopts, as many other logical and relational learning systems, the learning from interpretations framework [De Raedt, 2008], which allows to naturally represent entities (or objects) and the relationships amongst them.

kLog generates a set of features starting from a logical and relational learning problem and uses these features for learning a (linear) statistical model. Learning problems are described at three different levels. The first level specifies the logical and relational learning problem. At this level, the description consists of an E/R-model describing the structure of the data and the data itself, which is similar to that of traditional SRL systems [Heckerman *et al.*, 2007]. The data at this level is then *graphicalized*, that is, the interpretations are transformed into graphs. This leads to the specification of a graph learning problem at the second level. Graphicalization is the equivalent of knowledge-based model construction. Indeed, SRL systems such as PRMs and MLNs also (at least conceptually) produce graphs, although these graphs represent probabilistic graphical models. Finally, the graphs produced by kLog are turned into feature vectors using a graph kernel, which leads to a statistical learning problem at the third level. Again there is an analogy with systems such as Markov logic as the Markov network that is generated in knowledge-based model construction lists also the features. Like in these systems, kLog features are tied together as every occurrence is counted and is captured by a single same parameter in the final linear model.

Thus the key contributions of the kLog framework are threefold: 1) kLog is a language that allows users to declaratively specify relational learning tasks in a similar way as statistical relational learning (SRL) and inductive logic programming approaches; unlike many previous approaches to SRL [De Raedt *et al.*, 2008; Getoor and Taskar, 2007], it is based on kernel methods rather than on probabilistic modeling; 2) kLog compiles the relational domain and learning task into a graph-based representation using a technique called graphicalization; and 3) kLog uses a graph kernel to construct the feature space where eventually the learning takes place. This whole process is reminiscent of knowledge-based model construction in statistical relational learning.

2 The kLog language

We illustrate the kLog framework on a real-life example using the UW-CSE dataset prepared by Domingos et al. for demonstrating the capabilities of MLNs [Richardson and Domingos, 2006]. Basic entities include persons (students or professors), scientific papers, and academic courses. Available relations specify, e.g., whether a person was the author of a certain paper, or whether he/she participated in the teaching activities

*This paper is an extended abstract of the AI Journal publication [Frasconi *et al.*, 2014].

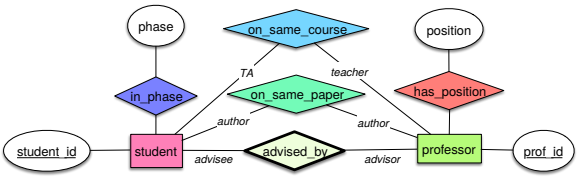

Figure 1: E/R diagram for the UW-CSE domain.

of a certain course. The goal is to predict the students' advisors, namely the binary relation advised_by between students and professors. Data comes in the form of true ground atoms, under the closed-world assumption. Since (first-order logic) functions are not allowed in the language, a ground atom is essentially like a tuple in a relational database, for example taught_by(course170,person211,winter_0102). kLog learns from interpretations. This means that the data is given as a set of interpretations (or logical worlds) where each interpretation is a set of ground atoms which are true in that world. In this illustration there are five interpretations: ai, graphics, language, systems, and theory, corresponding to different research groups in the department.

The first step in kLog modeling is to describe the domain using a classic database tool: entity relationship diagrams. We begin by modeling two entity sets: student and professor, two unary relations: in_phase and has_position, and one binary relation: advised_by (which is the target in this example). The diagram is shown in Figure 1.

Every entity or relationship that kLog will later use to generate features (see feature generation below) is declared using a special keyword signature. Signatures are similar to the declarative bias used in inductive logic programming systems. There are two kinds of signatures, annotated by the reserved words extensional and intensional. In the extensional case, all ground atoms have to be listed explicitly as data; in the intensional case, ground atoms are defined implicitly using Prolog definite clauses. Intensional signatures are one of the powerful features of kLog and allow programmers to introduce novel relations, using a mechanism resembling deductive databases. An intensional signature declaration must be complemented by a predicate (written in Prolog) which defines the new relation. Such relations are typically a means of injecting domain knowledge. In our illustration, it may be argued that the likelihood that a professor advises a student increases if the two persons have been engaged in some form of collaboration, such as co-authoring a paper, or working together in teaching activities. Writing Prolog predicates for defining such new relations is straightforward. When working on a given interpretation, kLog materializes all intensional predicates, that is, it computes and asserts all true ground atoms in that interpretation in the Prolog database. Intensional signatures can also be effectively exploited to introduce aggregated attributes [De Raedt, 2008] (e.g., to count the number of papers two persons have written together).

Graphicalization is our approach to capture the relational structure of the data by means of a graph. The use of an intermediate graphicalized representation is novel in the context of propositionalization, a well-known technique in logical and relational learning [De Raedt, 2008] that transforms relational data *directly* into an attribute-value format. This typically results in a loss of information, cf. [De Raedt, 2008]. Our approach transforms the relational data into an *equivalent* graph-based format, without loss of information. Graphicalization maps a set of ground atoms into a bipartite undirected graph whose nodes are true ground atoms and whose edges connect an entity atom to a relationship atom if the identifier of the former appears as an argument in the latter. The graph resulting from the graphicalization of a fragment of the ai interpretation is shown in Figure 2. Predicates that exist in the data but lack a corresponding signature (e.g., publication) do not produce nodes in the graph. However these predicates may be conveniently exploited in the bodies of the intensional signatures (e.g., on_same_paper refers to publication).

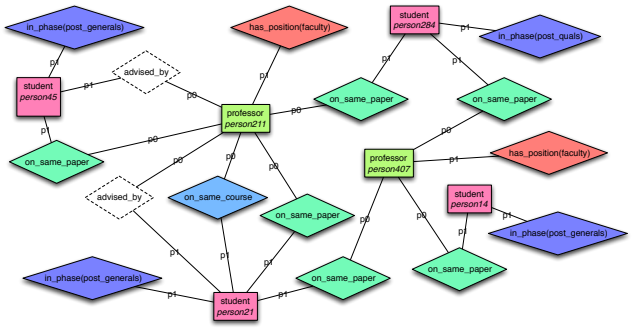

Figure 2: Graphicalized fragment of the ai interpretation.

After an interpretation z has been mapped into an undirected labeled graph G_z, a feature vector $\phi(z)$ may be extracted from G_z. Alternatively, a kernel function on pairs of graphs $K(z, z') = K(G_z, G_{z'})$ can be computed. Thus kLog directly upgrades graph-based kernels to fully relational representations. The graph kernel choice implicitly determines how the predicates' attributes are combined into features. Now that we have specified the inputs to the learning system, we still need to determine the learning problem. This is declared in kLog by designating one (or more) signature(s) as target (in this domain, the target relation is advised_by). Several library predicates are designed for training, e.g., kfold performs a k-fold cross validation. These predicates accept a list of target signatures that specifies the learning problem.

3 Graph kernel

In principle, any graph kernel can be employed in kLog. In the current implementation, we use an extension of NSPDK [Costa and De Grave, 2010]. While the original kernel is suitable for sparse graphs with discrete vertex and edge labels, here we propose extensions to deal with soft matches and a larger class of graphs whose labels are tuples of mixed discrete and numerical types. We also introduce a variant of NSPDK with viewpoints, which can handle the case of multiple predictions within the same interpretation.

NSPDK is a convolution kernel [Haussler, 1999] where *parts* are pairs of rooted patterns. Given a graph G, an integer r, and a vertex v, an r-pattern rooted in v is the rooted

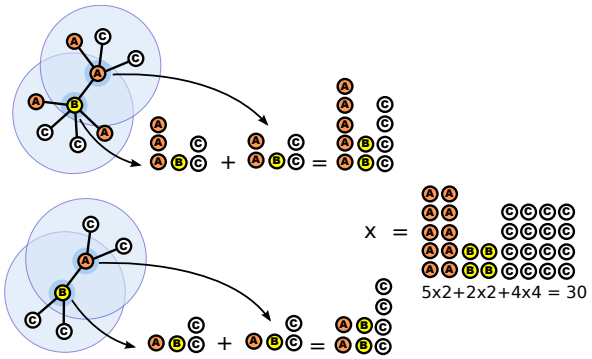

Figure 3: Illustration of the soft matching kernel between $(1,1)$-parts rooted in vertices labeled by A and B.

subgraph of G induced by vertices that can be reached from v with a shortest-path of length r. Given an integer d, an (r,d)-part is defined as a pair of r-patterns such that the shortest-path distance between their roots is exactly d. In the original formulation, NSPDK is parameterized by two integers R (maximum radius) and D (maximum distance) and $k(G,G')$ is the number of common (isomorphic) (r,d)-parts in g and G', for all $r \leq R$ and $d \leq D$.

Soft matches
The idea of counting exact pattern matches (up to isomorphism) to express graph similarity is adequate when the graphs are sparse and have low degree. For other graph classes the likelihood of finding common isomorphic patterns is low, yielding a diagonal dominant kernel prone to overfitting[1]. In these cases a better solution is to allow partial subgraph matches. In particular, we consider the frequencies of labels within a pair of r-patterns, discarding structural information (see Fig. 3).

Tuples of properties
A standard assumption in graph kernels is that vertex and edge labels are elements of a discrete domain. However, in kLog the information associated with vertices is a tuple that can contain both discrete and real values. Here we extend NSPDK to allow both a hard and a soft match type over graphs with property tuples that can be discrete, real, or a mixture of both types. The key idea is to use a canonical vertex identifier (see [Frasconi *et al.*, 2014]) to characterize each element in the tuple within each (r,d)-pattern: in this way the kernel is then the sum of kernels on the tuples that appear in vertices sharing the same canonical identifier. Different specializations of the kernel on tuples can be defined, depending on the type of property values in the tuple items and the type of matching required. These specializations include soft and hard matches for tuples of discrete, real-valued, and mixed type values (see [Frasconi *et al.*, 2014] for details). Note that the kernel on real-valued properties is limited to the dot product to ensure efficient computation.

[1]A concrete example is when text information associated to a document is modeled explicitly, i.e., when word entities are linked to a document entity: in this case the degree corresponds to the document vocabulary size.

Domain knowledge bias via kernel points
At times it is convenient, for efficiency reasons or to inject domain knowledge into the kernel, to be able to explicitly select a subset of parts to be matched. We provide a declarative way to do so, by introducing the notion of *kernel points*, a subset of vertices associated with the ground atoms of specially marked signatures. In this case, patterns whose roots are not kernel points are ignored. Kernel points are typically vertices that are believed to represent information of high importance for the task at hand. Note that vertices that are not kernel points still contribute to the kernel computation when they occur in the neighborhoods of kernel points.

Viewpoints
When we are interested in predictions about individual interpretations (like for example in the classification of small molecules), the above approach is immediately usable in conjunction with plain kernel machines like SVMs. When moving to more complex tasks involving, e.g., classification of entities or tuples of entities, one option is to resort to a structured output technique, where $f(x) = \arg\max_y w^\top \phi(x,y)$, where $\phi(x,y)$ is the feature vector associated with interpretation (x,y). Alternatively, we may convert the structured output problem into a set of independent subproblems, each consisting of predicting a single ground atom $c \in y$ of the target relation. In this case, we call the *viewpoint* of c, W_c, the set of vertices that are adjacent to c in the graph. In order to define a suitable kernel, we first consider the mutilated graphs where all vertices in y, except c, are removed. We then define a kernel on mutilated graphs, following the same approach of the NSPDK, but with the *additional constraint* that the first root must be in W_c. In this way, we do not obtain truly collective predictions. Still, even in this reduced setting, the kLog framework can be exploited in conjunction with meta-learning approaches that surrogate collective prediction. For example, Prolog predicates in intensional signatures can effectively be used as expressive relational templates for stacked graphical models [Kou and Cohen, 2007] where input features for one instance are computed from predictions on other related instances.

4 kLog in practice

We now illustrate how kLog can be applied in different types of learning problems. Details on results and experimental settings are given in the full paper [Frasconi *et al.*, 2014].

Predicting a single property of one interpretation
A simple example in this setting is the prediction of the biological activity of small molecules, a major task in chemoinformatics, where graph kernels (see [Vishwanathan *et al.*, 2010] and references therein) offer state-of-the-art solutions. From the kLog perspective the data consists of several interpretations, one for each molecule. To evaluate kLog we used two data sets: Bursi [Kazius *et al.*, 2005] and Biodegradability [Blockeel *et al.*, 2004].

The learning task in Bursi is to discriminate between mutagens and nonmutagens. Our results are stable (within .03 AUROC) with respect to the choice of kernel hyperparameter (maximum radius and distance) and SVM regularization and essentially match the best results reported in [Costa

and De Grave, 2010] (AUROC 0.92 $\pm$ 0.01 using functional groups and 0.9 $\pm$ 0.01 using atoms and bonds) even without composition with a polynomial kernel. By comparison, Tilde [Blockeel and De Raedt, 1998], accessing the same set of Prolog atoms as kLog, obtained 0.63 $\pm$ 0.09 (functional groups) and 0.8 $\pm$ 0.02 (atoms and bonds).

The Biodegradability data set contains 328 compounds and the regression task is to predict their half-life for aerobic aqueous biodegradation. We estimated prediction performance in the setting described in [Blockeel *et al.*, 2004]. When using functional groups, kLog obtained an RMSE of 1.07 $\pm$ 0.01. For comparison, the best RMSE obtained by kFOIL is 1.14 $\pm$ 0.04 (kFOIL was shown to outperform Tilde and S-CART in [Landwehr *et al.*, 2010]).

Link prediction

We focus on the application of kLog in the UW-CSE domain discussed in Section 2. We evaluated prediction accuracy according to the leave-one-research-group-out setup of [Richardson and Domingos, 2006]. The whole 5-fold procedure runs in about 20 seconds on a single core of a 2.5GHz Core i7 CPU. Compared to MLNs, kLog in the current implementation has the disadvantage of not performing collective assignment but the advantage of defining more powerful features thanks to the graph kernel. The recall-precision curve of kLog dominates that of MLN.

In a second experiment, we predicted the relation advised_by starting from partial information (i.e., when relations Student (and its complement Professor) are unknown, as in [Richardson and Domingos, 2006]). In this case, we created a pipeline of two predictors. Our procedure is reminiscent of stacked generalization [Wolpert, 1992]. In the first stage, a leave-one-research-group-out cross-validation procedure was applied to the training data to obtain predicted groundings for Student (a binary classification task on entities). Predicted groundings were then fed to the second stage which predicts the binary relation advised_by. Again, the recall-precision curve of kLog dominates that of MLN. Since kLog is embedded in the programming language Prolog, it is easy to use the output of one learning task as the input for the next one as illustrated in the pipeline. This is because both the inputs and the outputs are relations. Relations are treated uniformly regardless of whether they are defined intensionally, extensionally, or are the result of a previous learning run. Thus kLog satisfies what has been called the *closure* principle in the context of inductive databases [De Raedt, 2002]; it is also this principle together with the embedding of kLog inside a programming language (Prolog) that turns kLog into a true programming language for machine learning [Mitchell, 2006; De Raedt and Nijssen, 2011; Rizzolo and Roth, 2010].

Entity classification

The WebKB data set [Craven *et al.*, 1998] consists of academic Web pages from four computer science departments to be categorized (we used four classes: research, student, course, and faculty). Because of the relationships has, that associates words to web pages, vertices representing webpages have large degree in the graphicalized interpretations. In this domain we can therefore appreciate the flexibility of the soft match kernel. We compared kLog to MLN and to Tilde on the same task. For both systems we used logical formulae or language bias that encode the same domain knowledge exploited in kLog. For MLN, we followed the weight averaging protocol described in [Lowd and Domingos, 2007]. The average multiclass accuracies in the leave-one-university-out setting are 0.88 for kLog and MLN, and 0.86 for Tilde. The average F_1 measures are 0.88 for kLog, 0.81 for MLN, and 0.78 for Tilde. Although the accuracies of the three methods are essentially comparable, their requirements in terms of CPU time are dramatically different: using a single core of a second generation Intel Core i7, kLog took 36s, Alchemy 27,041s (for 100 iterations, at which the best accuracy is attained), and Tilde: 5,259s.

Domains with a single interpretation

The Internet Movie Database (IMDb) collects information about movies and their cast, people, and companies working in the motion picture industry. We focus on predicting, for each movie, whether its first weekend box-office receipts are over US$2 million, a learning task previously defined in [Neville and Jensen, 2003]. The learning setting defined so far (learning from independent interpretations) is not directly applicable since train and test data occur within the same interpretation. The notion of *slicing* in kLog allows us to overcome this difficulty. Intuitively, slicing sorts ground atoms according to a total order so that training atoms for the target relation are always *before* test atoms in that order. We sliced the data set according to production year, and starting from year $y = 1997$ until year $y = 2005$, we trained on the frame $\{y-1, y-2\}$ and tested on the frame $\{y\}$. kLog obtained an AUROC of 0.93 $\pm$ 0.03. In the same setting and using the same background knowledge, MLN and Tilde obtained AUROC 0.85 $\pm$ 0.03 and 0.87 $\pm$ 0.04, respectively. On this data set, Tilde was the fastest system, completing all training and test phases in 220s, followed by kLog (1,394s) and Alchemy (12,812s). However, the AUC obtained by kLog is consistently higher across all prediction years.

5 Conclusions

kLog tightly integrates logical and relational learning with kernel methods and constitutes a principled framework for statistical relational learning based on kernel methods rather than on graphical models. There are unanswered questions and interesting open directions for further research. One important aspect is the possibility of performing collective classification (or structured output prediction). Such learning problems can be naturally defined within kLog's semantics but developing clever and fast algorithms for this purpose is an interesting open issue. The graph kernel that is currently employed in kLog is just one of many possible choices. Another direction for future research is the implementation of a library of kernels suitable for different types of graphs (e.g., small-world networks or graphs with high-dimensional labels), as well as the integration of other existing graph kernels.

kLog is actively used for developing applications. We are exploring applications of kLog in natural language processing [Verbeke *et al.*, 2012a; 2012b; 2014; Kordjamshidi

et al., 2012] and computer vision [Antanas *et al.*, 2012; 2013].

Acknowledgments

Tom Schrijvers, KU Leuven SF/09/014, GOA/08/008, PRIN 2009LNP494, ERC StG 240186 'MiGraNT', IWT SBO 120025 'InSPECtor'.

References

[Antanas *et al.*, 2012] L. Antanas, P. Frasconi, F. Costa, T. Tuytelaars, and L. De Raedt. A relational kernel-based framework for hierarchical image understanding. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 171–180. Springer, November 2012.

[Antanas *et al.*, 2013] L. Antanas, M. Hoffmann, P. Frasconi, T. Tuytelaars, and L. De Raedt. A relational kernel-based approach to scene classification. In *WACV*, pages 133–139, 2013.

[Blockeel and De Raedt, 1998] H. Blockeel and L. De Raedt. Top-down induction of first order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, 1998.

[Blockeel *et al.*, 2004] H. Blockeel, S. Dzeroski, B. Kompare, S. Kramer, B. Pfahringer, and W.V. Laer. Experiments in Predicting Biodegradability. *Applied Artificial Intelligence*, 18(2):157–181, 2004.

[Costa and De Grave, 2010] F. Costa and K. De Grave. Fast neighborhood subgraph pairwise distance kernel. In *Int. Conf. on Machine Learning*, pages 255–262, 2010.

[Craven *et al.*, 1998] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web, 1998.

[De Raedt and Nijssen, 2011] L. De Raedt and S. Nijssen. Towards programming languages for machine learning and data mining (extended abstract). In *19th Int. Symp. on Foundations of Intelligent Systems*, pages 25–32, 2011.

[De Raedt *et al.*, 2008] L. De Raedt, P. Frasconi, K. Kersting, and S. Muggleton, editors. *Probabilistic inductive logic programming: theory and applications*, volume 4911 of *LNCS*. Springer, Berlin, 2008.

[De Raedt, 2002] L. De Raedt. A perspective on inductive databases. *SIGKDD Explorations*, 4(2):69–77, 2002.

[De Raedt, 2008] L. De Raedt. *Logical and relational learning*. Cognitive technologies. Springer, New York, 2008.

[Frasconi *et al.*, 2014] P. Frasconi, F. Costa, L. De Raedt, and Kurt De Grave. klog: A language for logical and relational learning with kernels. *Artificial Intelligence*, 217:117–143, 2014.

[Getoor and Taskar, 2007] L. Getoor and B. Taskar, editors. *Introduction to statistical relational learning*. MIT Press, Cambridge, Mass., 2007.

[Haussler, 1999] D. Haussler. Convolution kernels on discrete structures. Technical Report 99-10, UCSC-CRL, 1999.

[Heckerman *et al.*, 2007] D. Heckerman, C. Meek, and D. Koller. Probabilistic entity-relationship models, PRMs, and plate models. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*, pages 201–238. The MIT Press, 2007.

[Kazius *et al.*, 2005] J. Kazius, R. McGuire, and R. Bursi. Derivation and validation of toxicophores for muta- genicity prediction. *J. Med. Chem*, 48(1):312–320, 2005.

[Kordjamshidi *et al.*, 2012] P. Kordjamshidi, P. Frasconi, M. Van Otterlo, M.F. Moens, and L. De Raedt. Spatial relation extraction using relational learning. In *Int. Conf. on Inductive Logic Programming*, pages 204–220, 2012.

[Kou and Cohen, 2007] Z. Kou and W.W. Cohen. Stacked graphical models for efficient inference in markov random fields. In *Proceedings of the Seventh SIAM International Conference on Data Mining*, pages 533–538, 2007.

[Landwehr *et al.*, 2010] N. Landwehr, A. Passerini, L. De Raedt, and P. Frasconi. Fast learning of relational kernels. *Machine learning*, 78(3):305–342, 2010.

[Lowd and Domingos, 2007] D. Lowd and P. Domingos. Efficient weight learning for markov logic networks. In *Knowledge Discovery in Databases: PKDD 2007*, pages 200–211. Springer, 2007.

[Mitchell, 2006] T. Mitchell. The discipline of machine learning. Technical Report CMU-ML-06- 108, Carnegie-Mellon University, 2006.

[Neville and Jensen, 2003] J. Neville and D. Jensen. Collective classification with relational dependency networks. In *Workshop on Multi-Relational Data Mining*, 2003.

[Richardson and Domingos, 2006] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006.

[Rizzolo and Roth, 2010] N. Rizzolo and D. Roth. Learning based java for rapid development of nlp systems. In *Int. Conf. on Language Resources and Evaluation*, 2010.

[Verbeke *et al.*, 2012a] M. Verbeke, P. Frasconi, V. Van Asch, R. Morante, W. Daelemans, and L. De Raedt. Kernel-based logical and relational learning with klog for hedge cue detection. In *Int. Conf. on Inductive Logic Programming*, pages 347–357, 2012.

[Verbeke *et al.*, 2012b] M. Verbeke, V. Van Asch, R. Morante, P. Frasconi, W. Daelemans, and L. De Raedt. A statistical relational learning approach to identifying evidence based medicine categories. In *Proc. of EMNLP-CoNLL*, pages 579–589, 2012.

[Verbeke *et al.*, 2014] M. Verbeke, P. Frasconi, K. De Grave, F. Costa, and L. De Raedt. klognlp: Graph kernel–based relational learning of natural language. In *Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2014.

[Vishwanathan *et al.*, 2010] S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. Graph kernels. *J. Mach. Learn. Res.*, 99:1201–1242, August 2010.

[Wolpert, 1992] D.H. Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.