A Scalable Kernel Approach to Learning in Semantic Graphs with Applications to Linked Data

Yi Huang[1,2], Maximilian Nickel[2], Volker Tresp[1], and Hans-Peter Kriegel[2]

[1] Siemens AG, Corporate Technology, Munich, Germany
[2] Ludwig-Maximilians-Universität München, Munich, Germany

Abstract. In this paper we discuss a kernel approach to learning in semantic graphs. To scale up the performance to large data sets, we employ the Nyström approximation. We derive a kernel derived from semantic relations in a local neighborhood of a node. One can apply our approach to problems in multi-relational domains with several thousand graph nodes and more than a million potential links. We apply the approach to DBpedia data extracted from the RDF-graph of the Semantic Web's Linked Open Data (LOD).

1 Introduction

In this paper we consider semantic domains that form directed labeled graphs where nodes stand for concepts such as objects, categories or attributes and links describe simple subject-predicate-object statements: a directed arc points from the subject node (e.g., representing an object or a category), to an object node (e.g., representing an object, a category or an attribute) (Figure 1). The link is labeled by the predicate. Examples of such semantic graphs are the RDF (Resource Description Framework) graphs of the Semantic Web [1], the underlying data structure in the YAGO ontology [2] and the semantic graph format in [3]. We assume that the graph only contains links that are known to exist.[3] In this paper we consider the learning task of predicting links which are not present in the semantic graph but which are likely, considering statistical patterns that are implicit in the data. For example, we might predict the likelihood that Jack wants to be friends with Jane or that Jack's income is high or that Jack is a human being, and not a parrot. Our approach assumes the definition of a kernel defined for all nodes that form the population. We assume that the number of nodes in the population can be very large and thereby we suggest the Nyström approximation to solve the scalability issue. The Nyström approximations scales quadratically in the number of statistical units in the training set and scales linearly in the rank of the approximation. We show how the Nyström approximations is applied to least squares multivariate prediction. As special cases we obtain PCA regression, PCA matrix factorization and the reduced rank penalized regression (RRPP) algorithm used in the SUNS framework [4]. We then show how a semantic graph kernel can be derived from features derived from a local neighborhood of a node. Scalability is a major concern, in particular for applications to the Semantic Web as the population and the number of features can become very large. In

[3] A link, resp. the associated subject-predicate-object statement, is sometimes referred to as a triple.

the presented approach, the scalability of the overall approach is guaranteed. First, we can control the number of instances considered in the Nyström approximation. Second we can control the rank of the approximation. Third, we can control the number of local features that are used to derive the kernel. A special case of our approach was already presented in [4] and [5]. A novel contribution here is that we discuss the approach as a general kernel approach using the Nyström approximation. Another novelty is that we apply our approach to DBpedia [6], which is based on information extracted from Wikipedia. DBpedia is part of the Linked Open Data (LOD) cloud where the term Linked Data is used to describe a method of exposing, sharing, and connecting data via dereferenceable URIs on the Web [7].

The paper is organizes as follows. In the next section we discuss related work. In Section 3 we review the kernel approximation based on the Nyström approximation and we apply it to least-squares prediction. Section 4 defines our kernel approach to learning in semantic graphs. Section 5 reports results based on DBpedia data. Section 6 presents our conclusions.

2 Related Work

Recently, there has been quite some work on the relationship between kernels and graphs. Graph kernels evaluate the similarity between graphs and can be classified into three classes: graph kernels based on walks and paths, graph kernels based on limited-size subgraphs and graph kernels based on subtree patterns [8, 9]. It is not immediately clear how those approaches can be used for link prediction. Link prediction on graphs is quite related to semi-supervised learning as surveyed in [10] where the goal is to predict node labels based on known node labels in a graph. Kernels for semi-supervised learning have, for example, been derived from the spectrum of the Graph-Laplacian. In [11, 12] approaches for Gaussian process based link prediction have been presented. Link prediction in relational graphs has also been covered from the relational learning and the ILP communities [13–15]. Kernels for semantically rich domains have been developed by [16].

Most of the discussed kernel approaches cannot easily be applied to the rich semantic domains considered here. In fact, many have been developed in the context of a single object type and a single relation type. The experimental results on the semantic kernels described in [16] are still quite limited.

3 Scalable Kernel Solutions Using the Nyström Approximation

3.1 Defining the Population

A semantic graph typically consists of many different types of objects and many types of relations. In our statistical approach we only make statements on a subset of those nodes, which form the the statistical units or instances in the population. A statistical unit is an object of a certain type, e.g., a person. The population is the set of statistical units under consideration. In general, the population is application dependent. It is advantageous if the population is homogeneous. E.g., the set of all students in Munich might be a good choice whereas the set that includes all students in Munich and all professors in Berkeley might be problematic.

3.2 The Nyström Approximation

We now assume that for any two instances i and j in the population a kernel $k_{i,j}$ is defined. A subset of the population of size N, i.e., the sample, defines the training set. Let K be the kernel matrix (i.e., Gram matrix) for the training instances. In many applications N can be very large, therefore we now follow [17] and use the Nyström approximation to scale up kernel computations to large data sets.

The Nyström approximation is based on an approximation to eigen functions and starts with the eigen decomposition

$$K = UDU^\top \tag{1}$$

of the kernel matrix. The Nyström approximation to the kernel for two arbitrary instances i and j can be written as

$$k_{i,j} \approx k_{.,i}^\top \, U_r \, \text{diag}_r \, (1/d_l) \; U_r^\top \; k_{.,j}$$

where $\text{diag}_r \, (1/d_l)$ is a diagonal matrix containing the inverse of the r leading eigenvalues in D and where U_r contains the corresponding r columns of U.[4] Here, $k_{.,i}$ is a vector of kernels between instance i and the training instances.

There are two special cases of interest. First, the vector of approximate kernels between a statistical unit i and all units in the training data can be written as

$$k_{.,i} \approx U_r U_r^\top k_{.,i} \tag{2}$$

and the matrix of approximate kernels between all pairwise units in the training data is

$$K \approx U_r \; \text{diag}_r \, (d_l) \, U_r^\top . \tag{3}$$

These modified kernels can now be used in kernel approaches such as SVM learning or Gaussian process learning. In particular, the reduced rank approximation Equation 3 can greatly reduce the computational requirements [17].[5]

3.3 Example: Regularized Least Squares Solutions for Multivariate Prediction

We now assume that for an instance i we have L targets or random variables $y_i = (y_{i,1}, \ldots, y_{i,L})^\top$ available. We want to train a model of the form $\hat{y}_i = k^\top(.,i)W$ where W is an $N \times L$ weight matrix.

A regularized least squares cost function can be formulated as

$$\text{trace}(Y - KW)(Y - KW)^\top + \lambda \; \text{trace} W^\top K W$$

[4] Based on this approximation the rank of any kernel matrix is less than or equal to $r \leq N$.

[5] We use the Nyström approximation slightly differently from [17]. There, Equation 1 is used on a submatrix of K and Equation 2 is then used to approximate K.

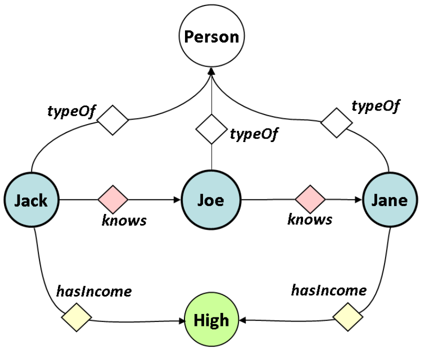

Fig. 1. Example of an RDF graph displaying a social friendship network in which the income of a person is an attribute. Concepts (resources) are represented by circular nodes and triples are represented by labeled directed links from subject node to object node. The diamond-shaped nodes stand for random variables which are in state *one* if the corresponding triples exist. Nodes representing statistical units (here: *Persons*) have a darker rim.

where $Y = (y_1, \ldots, y_N)^\top$ and where $\lambda \geq 0$ is a regularization parameter. If we use the Nyström approximation for the kernels we obtain as least squares solution for the weight matrix

$$W_{LS} = U \ \text{diag}_r \left(\frac{1}{d_l + \lambda} \right) \ U^\top Y.$$

The prediction for the training data (i.e., in smoothing or transduction) is

$$\hat{Y} = U \ \text{diag}_r \left(\frac{d_l}{d_l + \lambda} \right) \ U^\top Y$$

and in general

$$\hat{y}_i = k^\top(., i) W_{LS}.$$

We now consider some special kernels. Assume that for each instance i, in addition to the random variables of interest y_i, we also have covariates x_i available. Covariates might, for example, represent aggregated information. If the kernel can be written as an inner product of the covariates $k^x_{i,j} = x_i^\top x_j$, our Nyström approximation is equivalent to regularized PCA regression in that covariate space. Another interesting solution is when $k^y_{i,j} = y_i^\top y_j$ in which case our Nyström approximation is equivalent to regularized matrix reconstruction via PCA, often used in collaborative filtering. Note that in the latter case the low rank Nyström approximation is not only a necessity to obtain a scalable solution but is also necessary to obtain valid predictions at all: with $\lambda \to 0$ and $r = N$ we would obtain the trivial $\hat{Y} = Y$. Finally, with $k^z_{i,j} = z_i^\top z_j$ where $z_i = (\alpha x_i^\top, y_i^\top)^\top$, we obtain the reduced rank penalized regression (RRPP) algorithm in the SUNS framework [5]. Here, α is a positive weighting factor balancing the influence of the two information sources.

4 Kernel for Semantic Graphs

So far the discussion has been quite general and the Nyström approximation can be used for any kernel defined between instances in the population. As discussed in Section 2, there are a number of interesting kernels defined for nodes in a graph but most of them are not directly applicable to the rich domain of a semantic graph with many different node types and many different relation types. An exception is [16], which defines kernels exploiting rich ontological background knowledge.

We here present the kernel based on the SUNS framework [18]. The random variables represent the likelihood of links where the statistical unit is the subject or object. Additional features describe aggregated information. Although features are explicitly calculated, a kernel approach is still preferred since in the applications that we are considering the number of features can be quite large whereas N, the size of the sample, can be controlled more easily.

4.1 The Random Variables or Targets in the Data Matrix

Figure 1 shows a simple semantic graph with nodes *Person, Jack, Joe, Jane, High* and relation types *typeOf, knows, hasIncome*. We now introduce for each potential triple a *triple node* drawn as a diamond-shaped node in Figure 1. A triple node is in state *one* (*true*) if the triple is known to exist and is in state *zero* (*false*) if the triple is known not to exist. Graphically, one only draws the triple nodes in state *one*, i.e., the existing triples.

We now associate some triples with statistical units. The idea is to assign a triple to a statistical unit if the statistical unit appears in the triple. Let's consider the statistical unit *Jane*. Based on the triples she is participating in, we obtain *(?personA, typeOf, Person), (Joe, knows, ?personA)*, and *(?personA, hasIncome, High)* where *?personA* is a variable that represents a statistical unit. The expressions form the random variables (outputs) and define columns in the data matrix.[6] By considering the remaining statistical units *Jack* and *Joe* we generate the expressions (columns), *(?personA, knows, Jane)* and *(Jack, knows, ?personA)*. We will not add *(Jane, knows, ?personA)* since Jane considers no one in the semantic graph to be her friend. We iterate this procedure for all statistical units in the sample and add new expressions (i.e., columns in the data matrix), if necessary. Note that expressions that are not represented in the sample will not be considered.

In [4] the triples associated with a statistical unit were denoted as *statistical unit node set* (SUNS). The data matrix formed with the N statistical units as rows and the random variables as columns is denoted as Y. Note, that Y contains random variables derived for multiple different predicates.

4.2 Non-random Covariates in the Data Matrix

The columns in the data matrix that we have derived so far represent truth values of actual or potential triples. Those triples are treated as random variables in the analysis.

[6] Don't confuse a random variable representing the truth value of a statement with a variable in a triple, representing an object.

If the machine learning algorithm predicts that a triple is very likely, we can enter this triple in the semantic graph. We now add columns to the data matrix that provide additional information for the learning algorithm but which we treat as covariates or fixed inputs.

First, we derive simplified relations from the semantic graph. More precisely, we consider the expressions derived in the last subsection and replace constants by variables. For example, from *(?personA, knows, Jane)* we derive *(?personA, knows, ?personB)* and count how often this expression is true for a statistical unit *?personA*, i.e., we count the number of friends of person *?personA*.

Second, we consider a simple type of aggregated covariate from outside a SUNS. Consider first a binary triple *(?personA, knows, Jane)* . If Jane is part of another binary triple, in the example, *(?personA, hasIncome, High)* then we form the expression *(?personA, knows, ?personB)* $\wedge$ *(?personB, hasIncome, High)* and count how many rich friends a person has. A large number of additional covariates are possible but so far we restricted ourselves to these two types. The matrix formed with the N statistical units as rows and the covariates as columns is denoted as X and the complete data matrix becomes the matrix (X, Y).

Covariates are of great importance, in particular if statistical units are rather disconnected. For example, to predict social status of two professors at different universities in different countries, it might be relevant how many students they administer, but not exactly which students, or it might be important that they are the dean of some department, but not of which department. In social network terms: it might be relevant that they play the same roles.

5 Experiments with DBpedia Data

5.1 DBpedia Data

DBpedia [6] is part of LOD and contains structured information extracted from Wikipedia. At the time of writing this paper, it describes more than 3.4 million concepts, including 312,000 persons, 413,000 places and 94,000 music albums. DBpedia does not only serve as a “nucleus for the web of data”, but also holds great potential to be used in conjunction with machine learning approaches. Yet, even though DBpedia already provides a great value, it is still limited in the information it provides and in terms of quality. For example, although there are many cities covered in DBpedia, most information, like its most famous citizens and its most spectacular sights, is not very useful for machine learning purposes. Here we report results using a population consisting of all members of the German Bundestag to evaluate our approach. This population has been created by collecting all triples that are returned by the SPARQL query

```
SELECT ?s ?p ?o WHERE {
  ?s ?p ?o .
  ?s skos:subject dbp-cat:Members_of_the_German_Bundestag
}
```

5.2 Data Quality

A great benefit of LOD data is that by one simple SPARQL query the sample is defined. While DBpedia has great potential for machine learning, there are also challenges when these machine learning approaches are applied to DBpedia data. The first issue is related to the problem of incomplete data. It is very common for subjects in a DBpedia population to share only a subset of predicates. For instance, only 101 of 293 members of the German Bundestag represented in DBpedia have an entry for the predicate `dbp-ont:party` or `dbp-prop:party`. Therefore, in order to handle DBpedia data, a machine learning algorithm has to be able to deal with missing or incomplete data. The second issue is related to noisy predicates. For predicates it is often the case that there are semantical duplicates, e.g. `dbp-prop:party` and `dbp-ont:party`. While duplicate predicates are not a big problem by default, they can become a challenge when they are used inconsistently, which can greatly increase the preprocessing effort. Third, even more serious than noisy predicates are noisy objects. E.g. the Christian Democratic Union of Germany was represented by the literals `"CDU"` and `"Christian Democratic Union"` or the resources `dbpedia: Christian_Democratic_Union` and
`dbpedia: Christian_Democratic_Union_(Germany)`. Thus the true members of the CDU would have been divided into four distinct subsets and this needs to be resolved prior to learning. Finally, we have to consider the scale. The sample can get quite large when all available DBpedia data in a population is used.

5.3 Predicting Party Membership

In the following experiments the learning challenge was to correctly predict the political party for each subject, where the party is identified by the object of the predicate `dbp-prop:party`. Duplicate predicates would bias the experiments as they are heavily correlated with the target predicate. Therefore predicates like `dbp-ont:party` or `dbp-ont:Person/party` were removed. Moreover, predicate-object pairs that are very closely related to a party membership like `(?s, skos:subject, dbp-cat:Politicians_of_the_Social_Democratic_Party_of_Germany)` or `(?s, rdf:type, yago:GermanGreenPartyPoliticians)` were also removed. Rare features were sometimes pruned. In order to demonstrate the aforementioned challenges associated with DBpedia data, we conducted the following experiments

- ORIG: The original data from DBpedia (version 3.5.1). After pruning, this data set had $N = 293$ units, i.e., rows and 804 columns.
- DISAMB: In this experiment the objects of the target predicate were manually disambiguated solving the noisy objects problem. After the disambiguation exactly one concept (resource) for each party (CDU, CSU, SPD, FDP, Alliance ’90/The Greens, The Left, Centre Party) remained in the data set. Thus, for each statistical unit we estimate $L = 8$ variables. Furthermore, in the original data set only 101 of 293 statistical units had an entry for `dbp-prop:party` `dbp-ont:party`. Since machine learning algorithms benefit from a larger number of examples we

manually added the party for the remaining 192 units. After pruning, this data set had 802 columns.

- AGE: In this experiment the age of each politician was added as a continuous feature, by subtracting the birth year (when available) from the year 2010. To prevent that the age values dominated the remaining columns, age values were normalized. After pruning this data set had 804 columns.
- WEIGHT: We used a weighting coefficient of $\alpha = 0.4$ to put less importance on the covariates (see Section 3).
- STATE: The predicates `dbp-prop:birthPlace` or `dbp-ont:birthPlace` specify the city or village of birth. For the members with no entry here, we filled in the entry manually. Naturally, the birthplace is not a useful attribute for our task, whereas the state of the birthplace can be quite valuable, since in Germany, there are clear local party preferences. Filling in the state information from the birthplace information can easily be done by exploiting geographical part-of-relationships with OWL reasoning.
- TEXT: Finally associated textual information was exploited by tokenizing the objects of the predicates `rdf:comment` and `dbp-prop:abstract` and by adding one column for each occurring token. When a token was present for a particular statistical unit, the entry was set to one, else to zero. After pruning the data set had 2591 columns.
- ALL: In this experiment all previously described approaches were combined. Since the number of attributes changed, we also changed the weighting factor to $\alpha = 0.2$. After pruning this data set had 2623 columns.

Except for ORIG, the basis for all experiments was the DISAMB data set. To evaluate how well the party membership is predicted, we performed leave-one-out cross-validation by iterating over all subjects. In each iteration we set all `dbp-prop:party` entries for the subject of the current iteration to *zero* and used predicted estimates for ranking. As evaluation measures we used NDCG and bpref [19], the latter being often used in TREC tracks designed for evaluation environments with incomplete relevance data.

Figure 2 shows the results for NDCG and bpref. As expected, the results obtained from the raw data were worst with a score of 0.722. The effect of data cleaning from disambiguation improved the score by 7 points. A small improvement in score can be achieved by adding the age. This shows that age is a weak predictor of party membership, at least in this Bundestag data set. Furthermore, an improvement in score can be achieved by putting more weight on the quantity of interest, i.e., the party membership. The textual description sometimes contains strong hints on party membership and the score improves to 0.928. The state information is also quite relevant as an input, which is well explained by the peculiarities of German politics. Finally, quite a high score of 0.963 is achieved by a combination of all methods.

6 Conclusions and Outlook

We discussed a kernel approach for learning in semantic graphs. To scale up the performance to large data sets, we employed the Nyström approximation. Furthermore, we

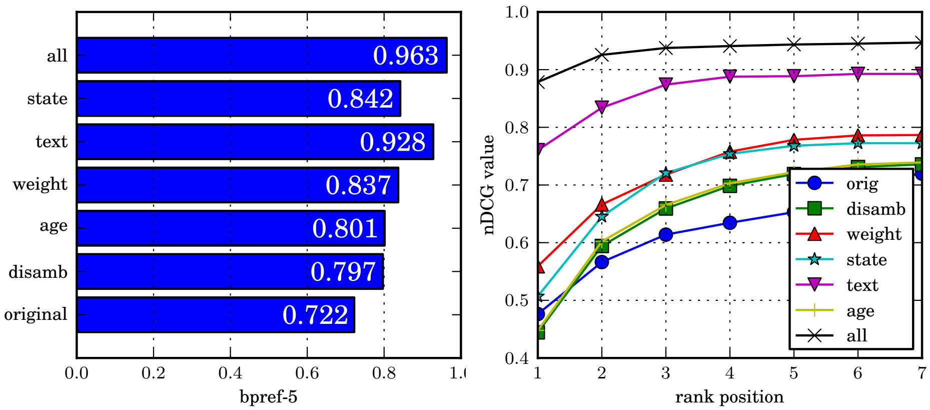

Fig. 2. Evaluation results for bpref and NDCG on the Bundestag population

presented a kernel for semantic graphs derived from a local neighborhood of a node and applied the approach to learning on the RDF-graph of the Semantic Web's Linked Open data (LOD).

To evaluate our approach, we applied it to data extracted from DBpedia. Here the data is quite noisy and considerable preprocessing is needed to yield good results. Also, by including textual data the prediction results were considerably improved. This improvement can already be observed even if a simple keyword based representation is being used without any sophisticated information extraction. Some of the data preprocessing steps can easily be executed with ontological (OWL-) reasoning, such as the generalization from city to state. In fact, materialization of facts derivable from logical reasoning is recommended as a preprocessing step. Other preprocessing steps, such as the calculation of age from the birthday and the current date, were done algorithmically.

In the DBpedia experiment, we estimated the membership in the 8 parties for each member in the Bundestag, thus $L = 8$. Although some members of the Bundestag have been in more than one party in their career, the collaborative coupling between the random variables is not contributing very much to the predictive performance. In [5], experiments in social networks are described with $L = 14425$ and a much stronger collaborative effect. As part of ongoing work we are studying a life-science domain with several hundred thousand covariates and with L greater than 3000.

Scalability of the overall approach is guaranteed. First, we can control the number of instances considered in the Nyström approximation. Second we can control the rank of the approximation. Third, we can control the number of local features that are used to derive the kernel. In our experiments, M, the number of features, was always quite high. In this case the most costly computation is the calculation of the kernel requiring N^2M operations.

Acknowledgements: We acknowledge funding by the German Federal Ministry of Economy and Technology (BMWi) under the THESEUS project and by the EU FP 7 Large-Scale Integrating Project LarKC.

References

1. Tauberer, J.: Resource Description Framework, http://rdfabout.com/
2. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: WWW 2007. (2007)
3. Strube, M., Ponzetto, S.P.: Wikirelate! computing semantic relatedness using wikipedia. In: AAAI 2006. (2006)
4. Tresp, V., Huang, Y., Bundschus, M., Rettinger, A.: Materializing and querying learned knowledge. In: Proceedings of the First ESWC Workshop on Inductive Reasoning and Machine Learning on the Semantic Web. (2009)
5. Huang, Y., Bundschus, M., Tresp, V., Rettinger, A., Kriegel, H.P.: Multivariate structured prediction for learning on the semantic web. In: Proceedings of the 20th International Conference on Inductive Logic Programming (ILP). (2010)
6. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. The Semantic Web (2008)
7. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. International Journal on Semantic Web and Information Systems (IJSWIS) (2009)
8. Vishwanathan, S.V.N., Schraudolph, N., Kondor, R.I., Borgwardt, K.: Graph kernels. Journal of Machine Learning Research - JMLR (2008)
9. Gärtner, T., Lloyd, J., Flach, P.: Kernels and distances for structured data. Machine Learning **57**(3) (2004)
10. Zhu, X.: Semi-supervised learning literature survey. Technical report, Computer Sciences TR 1530 University of Wisconsin Madison (2006)
11. Yu, K., Chu, W., Yu, S., Tresp, V., Xu, Z.: Stochastic relational models for discriminative link prediction. In: Advances in Neural Information Processing Systems (NIPS*2006). (2006)
12. Xu, Z., Kersting, K., Tresp, V.: Multi-relational learning with gaussian processes. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09). (2009)
13. Taskar, B., Wong, M.F., Abbeel, P., Koller, D.: Link prediction in relational data. In: Advances in Neural Information Processing Systems (NIPS*2003). (2003)
14. Muggleton, S., Lodhi, H., Amini, A., Sternberg, M.J.E.: Support vector inductive logic programming. In Hoffmann, A., Motoda, H., Scheffer, T., eds.: Discovery Science, 8th International Conference, DS2005. Volume 3735 of LNCS., Springer (2005)
15. Landwehr, N., Passerini, A., De Raedt, L., Frasconi: kFOIL: Learning simple relational kernels. In: National Conference on Artificial Intelligence (AAAI). (2006)
16. D'Amato, C., Fanizzi, N., Esposito, F.: Non-parametric statistical learning methods for inductive classifiers in semantic knowledge bases. In: IEEE International Conference on Semantic Computing - ICSC 2008. (2008)
17. Williams, C.K.I., Seeger, M.: Using the nystrom method to speed up kernel machines. In: Advances in Neural Information Processing Systems 13. (2001)
18. Tresp, V., Bundschus, M., Rettinger, A., Huang, Y.: Towards Machine Learning on the Semantic Web. In: Uncertainty Reasoning for the Semantic Web I. Lecture Notes in AI, Springer (2008)
19. Buckley, C., Voorhees, E.M.: Retrieval evaluation with incomplete information. In: Proc. 27th ACM SIGIR Conference on Research and Development in Information Retrieval. (2004)