Descriptive Approach to Modelling Learning

Miguel Martinez-Alvarez, Fabrizio Smeraldi, and Thomas Roelleke

Queen Mary, University of London
{miguel, fabri, thor}@dcs.qmul.ac.uk

Abstract. The importance of Learning Algorithms in Computer Science and other fields has been increasing in the last years. On the other hand, Descriptive Approaches have significantly impacted different domains, being SQL for data access and management one of the main examples. This paper investigates a descriptive approach for learning.
In the field of IR, learning techniques and descriptive approaches have already been applied independently. For the former, one of the most significant examples is the "Learning to Rank" task, while Probabilistic Datalog, which is a representative of descriptive approaches, has been applied for solving different IR-tasks, providing a high level representation of search strategies.
The main contributions of this paper are a knn classifier implemented in PDatalog, showing that the expressiveness of 2nd generation Probabilistic Datalog is sufficient for modelling lazy-learners, its evaluation for text classification on the Reuters-21578 collection, and a descriptive modelling of polynomial Mercer Kernels.

1 Introduction

Search tasks (information needs) are often more complex than captured by keyword-based query processing (e.g "Find the companies offering Probabilistic Reasoning Technology that are funded by the Government"). Even though there is a wide range of techniques available for classification, summarization and other tasks, combining those into a maintainable and scalable framework is a challenge.

Today's IR systems are often designed for a particular purpose. Therefore, the transfer and re-use of code is difficult, re-engineering seems the most effective process. This problem is comparable to what happened in the Software Industry, when Software Engineering evolved from implementing programs focused on specific tasks, to the development of frameworks for general tasks that could be adapted for specific ones, or, for the Database community, when SQL was adopted as the standard abstraction method for accessing information. In addition, the majority of IR systems are developed using traditional languages (e.g. C/C++, Java), commonly involving complicated knowledge transfer processes and maintainability. In a dynamic and technology-oriented environment a large amount of code is generated over time, and re-use and change management are challenging.

2 Background

2.1 Descriptive Approaches

We can assert that an approach is descriptive, also known as declarative, if "it involves stating what is to be computed, but not necessarily how it is to be computed" [10].

Descriptive approaches allow to define functionality in a high-level making the implementation clearer and the knowledge transfer easier. As a result, productivity will be increase [10]. In addition, they allow quick conceptual modifications in prototyping environments.

By using a descriptive approach it is possible to define different models and tasks as modules and then "concatenate" them, processing the information as a pipeline where the output of one module is the input of the following one. This solution does not involve any coding process because the modules selection is specified in the user interface level. This solution provides the flexibility needed for specifying and combine different IR tasks and/or models. Furthermore, it is possible to represent complex objects and structured data while maintaining scalability levels beyond any database system. Probabilistic Datalog (explained in Section 2.2) is the descriptive approach used in the experiments.

This is a significant line of research to pursue as it joins the requirements and techniques of areas such as semantic web, databases, logic, and uncertain reasoning. The long-term goal is to achieve a descriptive and composable IR technology that provides a framework of modules that information "engineers" can compose into a task-specific solution for an IR task. The ultimate goal is to achieve a framework of logical building blocks that offers classifiers, retrieval models, information extractors, and other functions, and those functional blocks can be composed in a possibly web-based service infrastructure. Thereby, high-level languages can be used that are translated to PDatalog for the purpose of composition and execution.

There are numerous studies about Datalog and PDatalog related to different tasks, some of which are summarized in section 2.2. In addition, some research has been done related to abstraction layers using descriptive approaches for different tasks. For instance, [3] proposed a declarative specification language (Dyna) for modelling NLP algorithms that can be automatically compiled to C++. They concluded that it is extremely helpful for their NLP research, even if it was slower than "hand-crafted" code. Other example is the description of a general framework that synthesizes and extends deductive and semiring parsing, adapting them for translation [11]. This work shows that logics make an attractive shorthand for description, analysis and construction of translation models. It also explains that descriptive approaches could be very beneficial when implementing large-scale translation systems which the authors identify as a major engineering challenge, requiring a huge amount of resources. In addition, the logical description has helped them to understand and compare the common elements for different models/algorithms and their differences.

2.2 Probabilistic Datalog

PDatalog is a probabilistic logical retrieval framework that combines deterministic Datalog (a query language used in deductive databases) and probability theory [7, 18]. It was extended in [19, 22] to improve its expressiveness and scalability for modelling IR models (ranking functions). In addition, it is a flexible platform for modelling and prototyping different IR tasks. Furthermore, probabilistic versions of Datalog are regarded for the semantic web as a platform layer on which other modelling paradigms (ontology-based logic) can rest and rely upon [17, 20].

The potentially high impact of this research lies in the fact that PDatalog is an abstraction layer that gives access to more than just the modelling of learning algorithms. PDatalog has been used as an intermediate processing layer for semantic/terminological logics in different IR tasks such as *ad-hoc* retrieval [13, 14], annotated document retrieval [6] and summarization [4].

Two generations of PDatalog have been developed and a third one is being developed. The main differences between them can be summarized as follows:

In 1st generation PDatalog, probabilistic rules have to be used to model the conditional probabilities. In addition, it has no means to express the probability estimation i.e. the "learning probabilities". Therefore, this process had to be external to PDatalog. In the 2nd generation PDatalog, Bayesian goals and subgoals support, on one hand the modelling of probability estimation, and, on the other hand, the modelling of conditional probabilities. Finally, the 3rd generation will involve high-level predicates related to specific tasks.

1st Generation PDatalog The 1st generation PDatalog for IR was introduced in [8]. The main idea is to allow for probabilities in facts and rules.

Figure 1 describes the syntax utilized in traditional Datalog and in the 1st generation of PDatalog. A PDatalog rule consists of a head and a body. A head is a goal, and a body is a subgoal list. A rule is evaluated such that the head is true if and only if the body is true. So far, the syntax is the one of ordinary Datalog. Specific to the PDatalog utilized here is the specification of materialised views (':=' indicates that the goal is to be materialised) and the probabilistic facts and rules.

2nd Generation PDatalog The 2nd generation of PDatalog provides extended expressiveness using probability estimation and conditional probabilities. It also improved scalability because probabilistic rules are not required and extensional relations and assumptions can be used in order to achieve efficient and scalable programs.

A simplified version (for improving readability) of the syntax specification for the 2nd generation PDatalog is outlined in Figure 2. Everything between a pair of curly brackets, i.e. '{' and '}', is optional. The assumption between predicate name and argument list is the so-called *aggregation* assumption (aggAssump). For example, for disjoint events, the sum of probabilities is the resulting tuple

Traditional Datalog	
fact	::= NAME '(' constants ')'
rule	::= head ':-' body
head	::= goal
body	::= subgoals
goal	::= NAME '(' arguments ')'
subgoal	::= pos_subgoal | neg_subgoal
pos_subgoal	::= atom
neg_subgoal	::= '!' atom
atom	::= NAME '(' arguments ')'
argument	::= constant | variable
constant	::= NAME | STRING | NUMBER
variable	::= VAR_NAME
arguments	::= | argument ',' arguments
constants	::= | constant ',' constants
subgoals	::= | subgoal ',' subgoals
1st Generation Probabilistic Datalog	
prob_fact	::= prob fact
prob_rule	::= prob rule

Fig. 1. 1st Generation PDatalog

probability. In this case, the assumptions 'DISJOINT' and 'SUM' are synonyms, and so are 'INDEPENDENT' and 'PROD'. The assumption in a conditional is the so-called *estimation* assumption (estAssump). For example, for disjoint events, the subgoal "index(Term, Doc) | DISJOINT(Doc)" expresses the conditional probability $P(Term|Doc)$ derived from the statistics in the relation called "index". Complex assumptions such as DF (for document frequency) and MAX_IDF (max inverse document frequency) can be specified to describe in a convenient way probabilistic parameters commonly used in IR.

Expressions with complex assumptions can be decomposed in PDatalog programs with traditional assumptions only. However, for improving the readability and processing (optimization), complex assumptions can be specified. The decomposition of complex assumptions is shown in [19].

2.3 Machine Learning

The Machine Learning community has developed a variety of tools for different applications. Arguably among the most successful approaches to supervised learning are Bayesian methods, ensemble methods such as boosting or algorithms for the selection of experts, Support Vector Machines (SVMs) [9] and kernel methods in general. While these in no way exhaust the field, they have sufficient generality and a range of applicability that goes from multimedia to learning ranking to real–time expert selection.

In particular, incorporating kernel methods and ensemble methods in an IR framework is highly desirable because ensemble methods can act as a "wrapper" for other classifiers seen as black-boxes, while kernel methods provide an easy

goal	::= tradGoal | bayesianGoal | aggGoal
subgoal	::= tradSubgoal | bayesianSubgoal | aggGoal
tradGoal	::= see 1st Generation
tradSubgoal	::= see 1st Generation
bayesGoal	::= tradGoal '|' {estAssump} evidenceKey
bayesSubgoal	::= tradSubgoal '|' {estAssump} evidenceKey
evidenceKey	::= '(' variables ')'
aggGoal	::= NAME {aggAssump} '(' arguments ')'
aggSuboal[1]	::= NAME {aggAssump} '(' arguments ')'
tradAssump	::= 'DISJOINT' | 'INDEPENDENT' | 'SUBSUMED'
irAssump	::= 'SEMI_SUBSUMED' | 'DF' | 'MAX_IDF' | ...
probAssump	::= tradAssump | irAssump
algAssump	::= 'SUM' | 'PROD'
aggAssump	::= probAssump
estAssump	::= probAssump | complexAssump

Fig. 2. 2nd Generation PDatalog: Bayesian Goals

way to integrate multimedia content through mapping to an appropriate vector space. Unfortunately there is no unified theory for these algorithms. However, some mathematical structures underlie one or more approaches and can be used to abstract them, as sketched in figure 3.

To begin with, both Adaboost and SVMs are solutions to optimization problems; thus a sufficiently flexible minimization engine can in principle handle both. However, both methods can potentially be abstracted in much more meaningful and descriptive ways, that we try to outline below.

Adaboost was originally derived in the context of PAC (Probably Approximately Correct) learning; some algorithms of this family have already been integrated in DataLog [15]. Two alternative interpretations of Adaboost look promising from the point of view of creating a high-level operational description of the procedure. As demonstrated in [5], Adaboost can be seen as an approximation to an additive regression model that optimises likelihood on a logistic scale. In this perspective, Adaboost rounds become the iterations of a greedy backfitting algorithm that tries to minimise the difference between the current estimate and the target function according to an exponential loss. Direct minimisation of the binomial log-likelihood leads to variant of the algorithm known as LogitBoost [5], that has a comparable performance. Therefore, one can essentially cast Adaboost as a maximum-likelihood regression procedure. This approach has the advantage that the "weak" classifiers that Adaboost combines (often binary predicated) can be interpreted as (coarse) estimates of the posterior probability of each class given the data, thus linking to a Bayesian framework. In another approach, Adaboost has been derived in terms of relative entropy [12], with a strong relation to classic information theory results.

SVMs, on the other hand, can nowadays be seen to fit in a different trend in Learning Theory that strives to cast learning in geometric terms. This can be traced back at least to the seminal work by Cucker and Smale [2], that systematised learning from examples and bounds on the generalisation error in

the framework of Functional Analysis. Incidentally, this work represents a shift towards considering the topological and geometric aspects of learning. In the same ideal line, we can count subspace methods, such as have been used in information retrieval [16], and manifold learning (see for example [21]).

On the other hand, the SVM optimization problem is more easily understood in geometrical terms (maximizing the margin). SVMs implicitly map the data into a higher dimensional feature space, in which the decision function is a hyperplane. This feature space is indeed a Hilbert space defined in terms of its scalar product (i.e., the kernel). If one considers a logic over its linear subspaces, the decision function can straightforwardly be interpreted as a proposition in the logic. More in general, kernel methods provide a flexible and efficient way of designing a mapping between the raw data and propositions in the logic. Thus research on kernel engineering could be harnessed, for instance, to provide a transparent integration of different data modalities. Tapping into this potential, however, requires a descriptive layer that is sufficiently rich to supports the logical, probabilistic and geometric concepts involved.

3 Modelling Learning in Probabilistic Datalog

The implementation of learning algorithms using a probabilistic logical approach presents the difficulty of modelling a huge number of methods with various theoretical foundations. The different nature of the methods implies that some of the techniques are easier to model than others. For example, Bayesian methods can be directly express in PDatalog, while the adaptation of SVM is not trivial.

For those methods that cannot be represented, two options would be considered. Firstly, research about the creation of a probabilistic approximation of the algorithm is required. Secondly, if that approximation cannot be found, the expressiveness of the language will be revised, studying how it could be modified. Some of the features we would want to include in the 3rd generation of PDatalog are the possibility of using sample spaces and fixed point criteria for allowing the implementation of some learning techniques based on optimization.

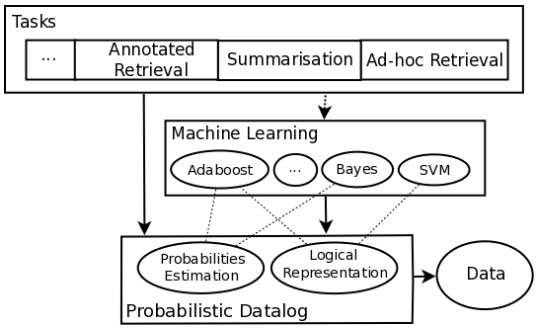

Fig. 3. Overview of a module-based IR System

Figure 3 shows a general diagram that illustrates how machine learning could be combined in a general IR-System implemented in PDatalog. The top level represents how different modules can be used for solving tasks such as annotated retrieval, summarization or ad-hoc retrieval. The intermediate layer models the machine learning algorithms that can be access by other modules if required. This it allows other modules to be tuned used ML techniques. The bottom layer illustrates the characteristics of PDatalog that support the definition of the different modules of the system.

Using this approach, the modules can use any fact derived in the others as long as they are selected for execution. As a result, different configurations can be specified by selecting appropriate sets of modules, without changing any code.

3.1 K Nearest Neighbours Classifier

A kNN classifier implemented in PDatalog is presented in figure 4. The code shown has been used for text classification on the Reuters-21578 collection (results in section 4.2). A sample of the relations "tf" (modelling the frequency of each term in a document) and "part_of" (assigning training documents with its correct classes) is also illustrated.

The functionality required for the kNN classifier has been composed from three different modules: Firstly, a document-frequency-based feature selection is applied, obtaining the terms that appear in most documents.

Secondly, a similarity measure is provided by a module that define a modification of the dice similarity following formula 1. This module computes, for each training document and a test document, the product of the weight of matching terms divided by the sum of the importance of all terms in the training document.

$$dice\_sim(testD, trainD) = \frac{\sum w(\text{term}, \text{trainDoc}) w(\text{term}, \text{testDoc})}{\sum w(\text{term}, \text{trainDoc})} \qquad (1)$$

where "w(t,d)" represent the weight of the term "t" in the document "d", tf-idf in our case.

Finally, a module is needed for defining the exact behaviour of the kNN classifier. It assigns the similarity score respect to each of the "k" most similar documents to the classes labelled for them. After this, these scores are aggregated given a final score for each of the classes previously chosen (Formula 2).

$$class\_score(class, testD) = \sum_{\text{trainDoc} \in \text{class}} sim(\text{testDoc}, \text{trainDoc}) \qquad (2)$$

Although this is the most common technique, there are other strategies for calculating the score of each class. For example, counting the number of neighbours from each class, without taken into account the score of their similarity.

tf		
Value	Term	Document
23	economy	d40
5	expectation	d23
12	provider	d23
7	reuters	d1

part_of		
Value	Document	Class
1	d1	cocoa
1	d5	grain
1	d5	wheat
1	d5	oil

(c) Basic/Extensional Data Representation

```
1  _sort (doc_frequency);
2  # 1. Select the 3000 terms that appear in more documents.
3  df_selected (Term) : doc_frequency(Term):3000;

5  # 2. For each tuple, is_selected (Term) is 1 if the term is one of the selected
       terms
6  is_selected (Term) :- df_selected(Term)|(Term);

8  #3. Compute the term frequency for the selected terms
9  selected_term(Term, Doc) :- selected(Term) & tf(Term, Doc);
10 sum_selected_term(Term, Doc) :- SUM selected_term(Term, Doc);
```

(d) Feature Selection

```
1  # 1. Normalize factor: Sum of feature weights ( tfidf ) for each training document
2  sum_tfidf SUM(TrainDoc) :- tfidf_train(Term, TrainDoc);
3  inv_norm_factor(TrainDoc) :- sum_tf_idf INV (TrainDoc);

5  # 2. Product of tfidf for the matching terms with each training document
6  match(TrainDoc, Term) :- tfidf_test(Term) & tfidf_train(Term, TrainDoc);
7  sum_match SUM(TrainDoc) :- match(TrainDoc, Term);

9  # 3. Similarity score
10 similarity (TrainDoc) :- sum_match(TrainDoc) & inv_norm_factor(TrainDoc);
```

(e) Similarity Measure

```
1  _sort ( similarity ); _sort (sum_top_k_classes);
2  # 1. Select the k training documents most-similar to the test document
3  top_similar (TrainDoc) :- similarity(TrainDoc):30;

5  # 2. Associate test document to the classes of the top-retrieved training docs
6  top_k_classes (Class) :- top_retrieved (TrainDoc) & part_of(TrainDoc, Class);

8  # 3. Aggregate scores for each class
9  sum_top_k_classes SUM(Class) :- top_k_classes(Class) | DISJOINT();
```

(f) Classification Strategy

Fig. 4. Modelling of kNN classifier with document-frequency-based feature selection, dice similarity measure and weighted neighbours

3.2 Implementing polynomial Mercer Kernels in Probabilistic Datalog

Central to SVMs and other Kernel methods such as Kernel PCA is the use of a Mercer Kernel as a non-linear scalar product. Using a Mercer kernel $K(\boldsymbol{u}, \boldsymbol{v})$ implicitly amounts to carrying out a non-linear mapping Φ of vectors $\boldsymbol{u}$, $\boldsymbol{v}$ from the original data space $\mathbb{R}^n$ into a higher–dimensional feature space $\mathbb{F}$, in which the standard scalar product is computed: $K(\boldsymbol{u}, \boldsymbol{v}) = \Phi(\boldsymbol{u}) \cdot \Phi(\boldsymbol{v})$ (see for instance [1]). The advantage is that the mapping Φ never needs to be computed explicitly, which allow handling high-dimensional or infinite-dimensional feature spaces $\mathbb{F}$ efficiently and transparently.

As an example, we provide an implementation in Probabilistic Datalog (Figure 3.2) of two kernels widely used in SVM classifiers, belonging to the polynomial family $K(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u} \cdot \boldsymbol{v} + c)^d$. Namely, we implement the homogeneus quadratic kernel $K_H = (\boldsymbol{u} \cdot \boldsymbol{v})^2$ and the non-homogeneous quadratic kernel $K_N = (\boldsymbol{u} \cdot \boldsymbol{v} + 1)^2$.

In the case of a two–feature data space $\mathbb{R}^2$ we can make the mapping Φ explicit by expanding the definition of the kernel:

$$K_H = (\boldsymbol{u} \cdot \boldsymbol{v})^2 = (u_1 v_1 + u_2 v_2)^2 \tag{3}$$

$$= (u_1^2 v_1^2 + 2 u_1 u_2 v_1 v_2 + u_2^2 v_2^2) = \Phi_H(\boldsymbol{u}) \cdot \Phi_H(\boldsymbol{v}) \tag{4}$$

from which we conclude that the implicit map is

$$\Phi_H(\boldsymbol{u}) = (u_1^2, \sqrt{2} u_1 u_2, u_2^2) \tag{5}$$

that transforms vectors in the data space $\boldsymbol{R}^2$ into vectors in $\mathbb{F}_H = \mathbb{R}^3$.

Similarly, by expanding the definition of K_N it can be seen that

$$\Phi_N(\boldsymbol{u}) = (u_1^2, \sqrt{2} u_1 u_2, u_2^2, \sqrt{2} u_1, \sqrt{2} u_2, 1) \tag{6}$$

so that $\mathbb{F}_N = \mathbb{R}^6$.

As can be seen, polynomial kernels implicitly compute the correlation and higher order moments of the original features, which arguably explains their effectiveness. In the general case of an n–dimensional data space $\mathbb{R}^n$ we have $F_H = \mathbb{R}^{n(n+1)/2}$ and $F_N = \mathbb{R}^{(n+1)(n+2)/2}$.

4 Feasibility Study

These experiments have been realised as a starting point for proving the feasibility of using declarative approaches for modelling a specific kind of learning methods, namely classifiers. The results illustrate that kNN can be modelled in PDatalog (the implementation is shown in Figure 4). Furthermore, its performance could be comparable with the quality of algorithmic approaches. However, more learning methods have to be implemented and tested in different environments.

```
1  # 1. Modelling the dot product between U and V and a constant (1)
2  kernel_match(Feature, U, V) :− value(Feature, U) & value(Feature,V);
3  # If the next line  is ignored/commented the homogeneus kernel will be computed
4  kernel_match SUM (Feature, U, V) :− constant(Feature, U, V);

6  # 2. Aggregate of product and constant (UV + 1)
7  sum_kernel_match(Feature, U, V) :− kernel_match SUM (Feature, U, V);

9  # 3. Quadratic polynomy (UV + 1)^2 or (UV)^2
10 K_quad_polyn(U, V) :− sum_kernel_match(Feature, U, V)
11                       & sum_kernel_match(Feature, U, V);
```

Fig. 5. Modelling of quadratic Mercer kernel

4.1 Experiment set-up

For this experiment, the "ApteMod" split of Reuters-21578 collection has been used. It is a corpus that contains news-wire histories divided in 7770 documents for training and 3019 for testing. We only consider documents that belong to classes with at least one train and one test documents. Stop-words removal and stemming have been applied.

A feature selection has been carried out based in the document frequency of each term. The terms that appear in more documents have been selected. The document representation has been realized using the tf-idf of each feature previously selected with respect to each document. The algorithm for calculating the similarity between documents is a slightly modification of the dice similarity (Formula 1). Finally, the assignation of each class is realised by choosing the best class after aggregate the weight of each neighbour.

This configuration was chosen due to its simplicity in order to realize a feasibility study. Feature selection based on document frequency is comparable to the performance of more complicated approaches such as Information Gain (IG) or Chi-squared (χ^2) with up to 90% term removal [24]. The number of neighbours considered (k) and the number of features selected have been specified taken into account the ranges recommended in [9, 23].

4.2 Results

The results present the quality measures for text classification using our kNN classifier on the Reuters-21578 collection. Only one class was assigned for each testing document. We have considered this strategy for simplicity reasons, assuming that this environment is good enough for the feasibility study. However, Yang pointed out that this simplification is not optimal for kNN.

Our approach achieved 75.81% in terms of micro-averaged F1. Even though this value is not yet comparable with the quality reported in [23](81.4%) and [9](82.3%), it is close enough for the feasibility study at this stage. In addition, the differences in performance are caused, at least in part, by the fact that an inferior feature selection algorithm has been used.

K	Representation	Similarity	FS	MicroF1
30	Norm-tf-idf	dice	DF-1000	0.7518
45	Norm-tf-idf	dice	DF-2000	0.7419

Fig. 6. F1 measures for Reuters-21578

5 Discussion and Further Work

This paper outlines the benefits of incorporating learning algorithms in PDatalog. In addition, its integration in a module-based IR system allows the application of learning techniques combined with different IR models and tasks. The main contributions of this paper are to present and discuss the issues related with modelling learning in PDatalog, showing the practical example of a kNN implementation, its evaluation for text classification on the Reuters-21578 collection, and a first attempt to model Mercer kernel methods.

We show that the modular modelling of kNN in PDatalog is not only possible, but also leads to an understandable and compact implementation. In addition, results suggest that descriptive approaches could achieve comparable quality with respect to other paradigms. However, direct comparisons, using exactly the same configuration, and experiments over more collections should be done for having stronger support for this claim.

The expected benefit of applying a descriptive approach is an increase of productivity while using learning methods and the capability of combining different tasks. Probabilistic learning algorithms such as PINs or Bayesian methods and lazy-learners like kNN can be represented using 2nd generation PDatalog. Other methods like SVM might require more expressiveness of the language or alternative processing techniques. An implementation of two polynomial Mercer kernels is provided, representing the first step forward for achieving this objective.

References

1. R. Courant and D. Hilbert. *Methods of mathematical physics*. Interscience, 1943.
2. F. Cucker and S. Smale. On the mathematical foundations of learning. *Bullettin of the AMS*, 39:1–49, 2002.
3. J. Eisner, E. Goldlust, and N. A. Smith. Compiling comp ling: practical weighted dynamic programming and the dyna language. In *HLT '05*, pages 281–290
4. J. F. Forst, A. Tombros, and T. Roelleke. Polis: A probabilistic logic for document summarisation. In *Studies in Theory of Information Retrieval*, p.201–212, 2007.
5. J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of statistics*, 28(2), 2000.
6. I. Frommholz. Annotation-based document retrieval with probabilistic logics. In ECDL 2007, pages 321–332
7. N. Fuhr. Probabilistic datalog - a logic for powerful retrieval methods. In *ACM SIGIR*, pages 282–290, 1995
8. N. Fuhr. Optimum database selection in networked ir. In *NIR'96. SIGIR'96 Workshop on Networked Information Retrieval*, 1996.

9. T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *ECML-98*, pages 137–142
10. J. W. Lloyd. Practical Advantages of Declarative Programming. In *Joint Conference on Declarative Programming*, 1994.
11. A. Lopez. Translation as weighted deduction. In *EACL 2009*, pages 532–540
12. P. Malacaria and F. Smeraldi. On Adaboost and optimal betting strategies. In *ICDM*, pages 326–332. 2009.
13. C. Meghini, F. Sebastiani, U. Straccia, and C. Thanos. A model of information retrieval based on a terminological logic. In *ACM SIGIR*, pages 298–308, 1993
14. H. Nottelmann. Pire: An extensible ir engine based on probabilistic datalog. In *ECIR*, pages 260–274, 2005.
15. H. Nottelmann and N. Fuhr. Learning probabilistic datalog rules for information classification and transformation. In *CIKM '01*, pages 387–394
16. B. Piwowarski, I. Frommholz, Y. Moshfeghi, M. Lalmas, and K. van Rijsbergen. Filtering documents with subspaces. In *32nd ECIR Conference*, 2010.
17. A. Polleres. From SPARQL to rules (and back). In *16th WWW*, pages 787–796. 2007
18. T. Roelleke and N. Fuhr. Information retrieval with probabilistic datalog. In *Uncertainty and Logics - Advanced models for the representation and retrieval of information*. Kluwer Academic Publishers, 1998.
19. T. Roelleke, H. Wu, J. Wang, and H. Azzam. Modelling retrieval models in a probabilistic relational algebra with a new operator: The relational Bayes. *VLDB Journal*, 17(1):5–37, January 2008.
20. S. Schenk. A SPARQL semantics based on Datalog. In *KI '07*, pages 160–174, 2007.
21. J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.
22. H. Wu, G. Kazai, and T. Roelleke. Modelling anchor text retrieval in book search based on back-of-book index. In *SIGIR '08 Workshop on Focused Retrieval*, pages 51–58, 2008.
23. Y. Yang and X. Liu. A re-examination of text categorization methods. In SIGIR '99, pages 42–49, 1999.
24. Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of ICML-97*, 1997.