A Descriptive Approach to Classification

Miguel Martinez-Alvarez and Thomas Roelleke

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

Abstract. Nowadays information systems are required to be more adaptable and flexible than before to deal with the rapidly increasing quantity of available data and changing information needs. Text Classification (TC) is a useful task that can help to solve different problems in different fields. This paper investigates the application of descriptive approaches for modelling classification. The main objectives are increasing abstraction and flexibility so that expert users are able to customise specific strategies for their needs.
The contribution of this paper is two-fold. Firstly, it illustrates that the modelling of classifiers in a descriptive approach is possible and it leads to a close definition w.r.t. mathematical formulations. Moreover, the automatic translation from PDatalog to mathematical formulation is discussed. Secondly, quality and efficiency results prove the approach feasibility for real-scale collections.

1 Introduction & Motivation

Nowadays, information systems have to deal with multiple sources of available data in different formats and rapidly changing requirements reflecting diverse information needs. As a result, the importance of adaptability and productivity in Information Retrieval (IR) systems is increasing. This fact is especially important in business environments when the information required at different moments can be extremely different and its utility may be contingent on timely implementation. How quickly a new problem is solved is often as important as how well you solve it.

Current systems are usually developed for specific cases, implying that too much time is spent having to rewrite and check high portions of the original implementation for other purposes. We believe that if the gap between the conceptual model and the implementation is minimised, expert users would be able to directly define specific strategies for solving their information needs. Therefore, increasing the productivity and adaptability of current approaches. Descriptive approaches are a well suited solution for this objective due to the high-level definition of models and their "Plug & Play" capabilities that allow quick changes with minimum engineering effort. This paper focuses on providing an abstraction for the classification task using a descriptive approach. Classification has been applied in several situations for different purposes and fields, making it a suitable candidate for being part of a flexible framework.

We aim to show, using different examples, how our approach provides an understandable and elegant modelling of classifiers, close (or even translatable) to its mathematical equation. This abstraction provides the flexibility and adaptability required for dynamic environments, allowing expert users to customise specific strategies. The long-term goal is to achieve a descriptive and composable IR technology that can be customised into a task-specific solution.

The remainder of this paper is structured as follows: Section 2 briefly reviews Naive-Bayes and k-NN classifiers and descriptive approaches, Section 3 presents their modelling using a descriptive approach. Furthermore, the automatic translation of these models to mathematical equations is discussed, Section 4 shows the results achieved using two standard Text Classification collections. Finally, Section 5 concludes the paper and discusses future work.

2 Background & Related work

2.1 Standard Classifiers

Text Classification is a useful task that assigns elements to one or more classes from a preselected set. It has been used in different fields for different purposes such as news categorisation or spam detection [18]. It has been traditionally based on term-based representations, viewing documents as bags of words.

Naive-Bayes Classifiers Naive-Bayes classifiers use the Bayes Theorem for inferring knowledge assuming the independence between features, given the context of a class [10]. This is a common assumption that makes the computation feasible. As an indirect result, it can be applied to larger collections. Given this assumption, the probability of a document being labelled in a class is defined in equation 1.

$$P(\mathrm{c}_k|d_i) = \frac{P(\mathrm{c}_k) \cdot P(\mathrm{d}_i|\mathrm{c}_k)}{P(\mathrm{d}_i)} = \frac{P(\mathrm{c}_k) \cdot \prod_{\mathrm{t}\in\mathrm{d}_i} P(\mathrm{t}|\mathrm{c}_k)}{P(\mathrm{d}_i)} \tag{1}$$

$$\mathrm{score}_{\mathrm{NB}}(\mathrm{c}_k, \mathrm{d}_i) := \frac{P(\mathrm{c}_k) \cdot \prod_{\mathrm{t}\in\mathrm{d}_i} P(\mathrm{t}|\mathrm{c}_k)}{\sum_k P(\mathrm{c}_k) \cdot \prod_{\mathrm{t}\in\mathrm{d}_i} P(\mathrm{t}|\mathrm{c}_k)} \tag{2}$$

All classifiers that make this assumption are usually referred to as Naive-Bayes, even if there are differences between them in the probabilities computation [10].

k-Nearest Neighbours (k-NN) Classifiers K-NN is a lazy learning instance-based method that has been used for classification and pattern recognition tasks for the last 40 years [18]. It categorises documents into classes taking into account what train examples are the "nearest" based on a similarity measure. There are several strategies for, given a document, computing the score for each class. One of the most common is presented in Equation 4 where the score of a class is the

sum of similarity scores for those documents labelled in the class, observing only the k most similar documents. Although this is the most common technique, there are others such as counting the number of neighbours from each class, without taken into account the score of their similarity.

$$\text{sim}(d_i, d_j) := \frac{d_i \cdot d_j}{||d_i|| \cdot ||d_j||} \tag{3}$$

$$\text{score}_{\text{k-NN}}(c, d_i) := \sum_{d_j \in c} \text{sim}_k(d_i, d_j) \tag{4}$$

Two parameters are needed, the number of neighbours (k) and the similarity algorithm. The former varies depending on the collection while, for the latter, the cosine similarity is the most common option.

2.2 Descriptive Approaches

Descriptive approaches allow to define high-level functionality making the implementation clearer and the knowledge transfer easier. As a result, productivity will be increased [8]. Models and tasks can be defined as modules and then "concatenated", processing the information as a pipeline where some outputs of one module are the input of the following one. This combination does not involve any coding process due to the paradigm's "Plug & Play" capabilities offered by its functional syntax. This solution provides the flexibility needed for specifying and quickly combining different IR tasks and/or models. Furthermore, it is possible to represent complex objects and structured data.

Research has been done related to abstraction layers using descriptive approaches for different tasks. For instance, a declarative specification language (Dyna) has been used for modelling Natural Language Processing (NLP) algorithms [2], concluding that it is extremely helpful, even if it is slower than "handcrafted" code. Other example is the description of a framework that synthesises and extends deductive and semiring parsing, adapting them for translation [9]. This work shows that logic make an attractive shorthand for description, analysis and construction of decoding algorithms for translation. 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. Among descriptives approaches, Probabilistic Logics has been applied for modelling and reasoning in different environments several times [7]. The language that has been used in this paper, Probabilistic Datalog(explained in section 2.3), is one of its representatives. Similar languages such Problog [14] and P-Log [6] have also been used for modelling and reasoning. In addition, although there are is research specifically related to the task of modelling classifiers using a descriptive approach [13, 1], they are focused in learning rules and/or use domain ontologies.

2.3 Probabilistic Datalog

The specific language that is used for the modelling of classifiers is Probabilistic Datalog (PDatalog). It is based on Probabilistic Logics and it combines Datalog (query language used in deductive databases) and probability theory [5, 15]. It was extended to improve its expressiveness and scalability for modelling ranking models [16].

```
1  #P(grade|degree): Learned from knowledge base.
2  p_grade_degree SUM(Grade, Degree) :−
3          grade(Student, Grade, Degree)|(Degree);
4
5  #P(grade|person): Inferred using P(grade|degree)
6  p_grade_person (Grade, Person) :−
7          p_grade_degree(Grade, Degree)
8          & register (Person, Degree);
9
10 # Given... grade(John, B, Art);
11 #              grade(Mary, B, Maths); grade(Anna, A, Art);
12 #              register (Matt, Art);  register (Mike, Maths);
13
14 # Results...  0.5 p_grade_person(A, Matt);
15 #              0.5 p_grade_person(B, Matt);
16 #              1.0 p_grade_degree(B, Mike);
```

Fig. 1. PDatalog example code

It is a flexible platform that has been used as an intermediate processing layer for semantic/terminological logics in different IR tasks such as *ad-hoc* retrieval [11, 12], annotated document retrieval [4] and summarisation [3]. Furthermore, Datalog has been applied for Information Extraction [19], highlighting the advantages of its declarative paradigm. Figure 1 shows and example that computes the probability of a student obtaining a specific grade ($P(grade|student)$) based on probabilities of grades given subjects from the previous year.

3 Modelling Classification in Probabilistic Datalog

Probabilistic Logics allows to model more compact and shorter definitions than other approaches, minimising the gap w.r.t. the mathematical formulation. This fact implies that the processes of knowledge transfer and maintainability will be easier. Furthermore, this abstraction leads to the possibility of experts users modelling specific and complex information needs. The main challenges of this approach are the efficiency/scalability and the expressiveness. The reason for the latter is that the increase in abstraction also implies that certain operations

cannot be modelled. Therefore, a balance between abstraction and expressiveness is needed. Moreover, the specific case of modelling classifiers presents the additional difficulty of modelling a huge number of methods with various theoretical foundations where the different nature of the methods implies that some of the techniques are easier to model than others.

Before illustrating the modelling of classifiers, Tables 2 and 3 show a sample of data representation in tabular and probabilistic logical format for the relations tf_sum (normalised term-doc occurrences by the number of terms) and $part\_of$ which models the labelling of documents and classes.

Fig. 2. Tabular Data Representation

tf_sum		
Value	Term	Document
0.23	economy	d40
0.52	expectation	d23
0.12	provider	d23
0.16	reuters	d1

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

Fig. 3. Probabilistic Logical Data Representation

```
1  0.23 tf_sum(economy, d40);
2  0.52 tf_sum(expectation, d23);
3  0.12 tf_sum(provider, d23);
4  0.16 tf_sum(reuters, d1);
```

```
1  part_of(d40, cocoa);
2  part_of(d23, grain);
3  part_of(d23, wheat);
4  part-of(d1, oil);
```

Figures 4 and 5 illustrate the general modelling of Naive-Bayes and k-NN classifiers (the modelling of cosine similarity is also shown as the similarity function). The modelling of Naive-Bayes requires the relation $term$[1] which models each word occurrence in a document and $part\_of$. Rules for representing the term and class space are specified (is_term, is_class), as well as the occurrences of terms in classes ($term\_class$) and all the combinations between terms and classes ($term\_class\_full$). There are two different estimations for $P(t|c)$ based on Laplace and minimum probability smoothing. In the first case the relation $term\_class$ is augmented[2] adding one extra occurrence for each term and class. The final $P(t|c)$ estimation is computed by using the bayes expression, dividing each tuple in $term\_class$ by the sum of all tuples with the same class. The minimum probability smoothing adds a fixed probability to estimation based on the term-class space. $P(d|c)$ is computed by aggregating all the tuples in p_t_c for a given test document using the PROD expression. The last steps are multiplying this value by the probability of the class and normalising of the results.

[1] Automatically created based on a set of documents.

[2] The same head in different rules implies that the tuples from both rules are united

```
1  # preliminary and term−class relations
2  is_term FIRST(T) :− term(T, D);
3  is_class FIRST(T) :− part_of(D, C);
4  term_class(T, C) :− term(T, D) & part_of(D, C);
5  term_class_full (T, C) :− is_term(T) & is_class (C);

7  #Laplace estimation for P(t|c)
8  term_class_laplace (T,C) :− term_class(T,C);
9  term_class_laplace (T,C) :− term_class_full (T,C);
10 p_t_c_laplace SUM(T,C) :− term_class_laplace(T,C) | (C);

12 # Minimum probability estimation for P(t|c)
13 p_t_c_aux_min (T,C) :− term_class(T,C) | (C);
14 p_t_c_aux_min (T,C) :− minProb() & term_class_full(T, C);
15 p_t_c_min SUM(T,C) :− p_t_c_aux_min(T,C);

17 # Generic computation given P(t|c)
18 p_d_c PROD(D, C) :− test_term(T, D) & p_t_c(T,C);
19 p_c_d(C, D) :− p_c(C) & p_d_c(D, C);
20 score_nb(C, D) :− p_d_c SUM(D,C)|(D);
```

Fig. 4. Modelling of Naive-Bayes Family in PDatalog

On the other hand, for the modelling of k-NN, relations $final\_test\_weight$ and $final\_weight$ model the importance of terms for testing and training documents. The specific weighting schema to be used is specified by the user (i.e. *ltc*). The first two rules of the modelling compute the euclidean normalisation for term-document weight w.r.t. the same document (tuples sharing the same D). The the last line for the cosine modelling computes the product of train and test documents sharing the same term. The final aggregation (*score_knn*) needs *top_similarity* to be customised, specifying the number of neighbours k and a similarity measure. For instance, cosine similarity with 45 neighbours (top_similarity(D1, D2) :- cosine(D1, D2):45).

```
1  # Cosine definition
2  vec1_norm(T, D) :− final_test_weight(T, D)|EUCLIDEAN(D);
3  vec2_norm(T, D) :− final_weight(T, D)|EUCLIDEAN(D);
4  cosine SUM(D1, D2) :− vec1_norm(T, D1) & vec2_norm(T, D2);

6  # k−NN score
7  score_knn SUM(C,D1) :− top_similarity(D1,D2) & part_of(D2,C);
```

Fig. 5. Modelling of k-NN PDatalog

The "Plug & Play" capabilities of our approach allow to use and customise any strategies or algorithm that has been modelled before with minimum engineering effort. For each of the algorithms modelled in the framework, a list of defined predicates is presented as a predicate dictionary (example given in Appendix B). In addition to the enumeration of available predicates, it specifies the requirements for making their computation possible. This includes the mapping rules for customisation (i.e. specifying number of neighbours) and the relations that have to be defined (i.e. term relation).

Figure 6 illustrates this idea where the predicates are obtained from specific models for classification and other IR-related tasks. As a result, an expert user would be able to create specific (multi task) models using these predicates without any engineering effort using a high-level language increasing the productivity and adaptability. As en example, Figure 7 illustrates the high-level specification of a k-NN algorithm with *ltc* weights, cosine similarity and 45 neighbours.

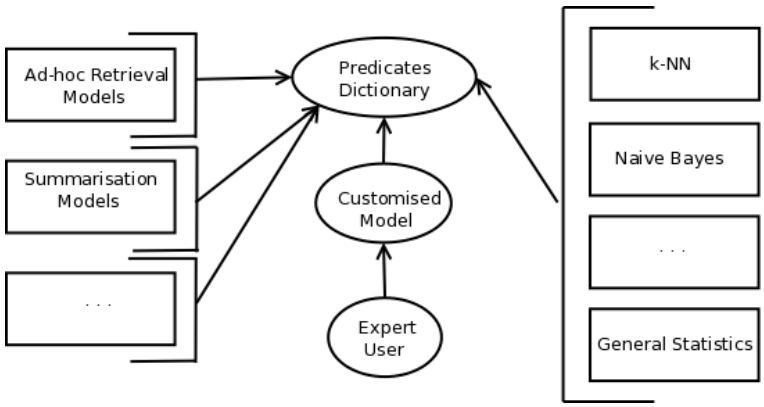

Fig. 6. Dictionary-Based Architecture

```
1  # Representation algorithms.
2  final_weight (T,D) :- ltc (T,D);
3  final_test_weight (T,D) :- test_ltc (T,D);

5  # Similarity based on cosine
6  _sort (cosine);
7  top_similarity (D1, D2) :- cosine(D1, D2):45;

9  # Final score
10 score(C, D) :- score_knn(C, D);
```

Fig. 7. Example of Strategy Customisation

3.1 Proving the Correctness of PD Programs

One of the benefits of high-level modelling is that not only the definitions are close to the mathematical formula but it is possible to analytically corroborate if they represent the same concept. Mathematical proof of the correctness of Naive-Bayes (with minimum probability estimation) and k-NN with cosine similarity are discussed in Propositions 1 and 2 respectively. Translations from PDatalog expressions to mathematical formulas is provided in Appendix B.

Proposition 1. *The modelling of Naive-Bayes using PDatalog, illustrated in Figure 4, is correct w.r.t. Equation 2 assuming a minimum probability for the cases where* $P(t|c) = 0$.

Proof. It is assumed that $P(t|c)$ is computed by using the maximum likelihood in the term class occurrences space $P(t|c) = \frac{n_L(t,c)}{\sum_i n_L(t_i,c)}$, assigning a minimum probability for the cases when its zero. This is modelled using the relational Bayes expression "|" over the *term_class_min* relation that has been computed by adding the *term_class* elements and (different rules with the same head are translated as a sum aggregation) the minimum value for all possible term-class tuples. The next step, after having $P(t|c)$ represented, is applying the product (using $PROD$) over the relation, computing $P(d|c)$. After this, only a product with $P(c)$ is needed (expression & in this PDatalog case).

Proposition 2. *The modelling of k-NN in PDatalog presented in Figure 5 is correct w.r.t. Equation 4.*

Proof. Cosine similarity is modelled as a product of the euclidean normalisation of test and train documents. In both cases, the normalisation is computed as $\frac{weight(t,d)}{\sqrt{\sum_i weight(t_i,d)^2}}$. This is modelled with the expression "$EUCLIDEAN$" which represents that the normalisation is done for those tuples sharing the same variable D. The relation *top_similarity* matches perfectly the function sim_k. The SUM expression aggregates all tuples with the same C and D1 variables. Therefore, it could be translated as a sum over tuples with different values for the variable D2 (d_j in the mathematical version). Finally, the fact that only values with a value of $part\_of$ are considered means that similarity is only computed if D2 belongs to class C which is directly translated into $\sum_{d_j \in C}$.

4 Feasibility Study

Experiments have been carried out using different real-scale collections and a variety of models, all of which have been modelled using a descriptive approach. The main goal is to empirically prove that our approach achieve comparable quality than other paradigms while maintaining reasonable efficiency levels.

4.1 Experiment set-up

Two traditional text classification collections have been used for the experiments: 20newsgroups and Reuters-21578. 20 Newsgroups[3] is a collection of approximately 20,000 newsgroup documents and 20 classes, some of them extremely similar (i.e. "comp.windows.x" and "comp.os.ms-windows.misc"), with almost uniform distribution of documents over classes. The split for the collection is based on time as it is suggested.

Reuters-21578[4] contains structured information about newswire articles that can be assigned to several classes. The "ModApte" split is used and only documents that belong to classes with at least one train and one test documents are considered. As a result, there are 7770 documents for training and 3019 for testing, observing 90 classes with a highly skewed distribution over classes.

In both cases several feature selection measures, weighting schemas and variations of Naive-Bayes and k-NN algorithms are tested. The name of each row represents (in this order) feature selection measure and number of features (i.e. chi_2000) and model (i.e. knn_45_ltc_cosine). For k-NN, the model name includes the weighting schema and the similarity function. The quality achieved by each module is presented in micro and macro Precision/Recall break-even point. A (shared) server with four dual-core 3GHz Opteron and 32GB of RAM and the engine HySpirit [17] have been used for the executions. The average time per document in testing has been obtained, as it is usual, by averaging the time for classifying the testing documents one by one. Only one representative per model is represented in the table because changing the configurations almost does not have any impact in the efficiency.

Table 1. Quality of Classifiers

	20-newsgroups		**Reuters-21578**	
	mBEP	MBEP	mBEP	MBEP
chi_1000_bayes_log_laplace	62.63	63.88	70.28	50.84
chi_1000_bayes_log_min	63.25	64.76	72.2	53.15
chi_1000_knn_30_ltc_norm_cosine	68.18	68.13	79.33	60.93
chi_1000_knn_30_tfc_norm_cosine	66.69	66.08	80.77	62.77
chi_1000_knn_45_ltc_norm_cosine	68.1	67.78	78.99	59.95
chi_1000_knn_45_tfc_norm_cosine	65.89	65.85	80.43	64.51
chi_2000_bayes_log_laplace	63.44	65.97	71.48	48.27
chi_2000_bayes_log_min	63.53	65.55	73.19	49.57
chi_2000_knn_30_ltc_norm_cosine	**70.03**	**69.87**	81.36	62.8
chi_2000_knn_30_tfc_norm_cosine	67.9	67.76	**82.3**	64.85
chi_2000_knn_45_ltc_norm_cosine	69.57	69.32	80.59	59.95
chi_2000_knn_45_tfc_norm_cosine	67.64	67.78	82.16	**64.95**

[3] Obtained from http://people.csail.mit.edu/jrennie/20Newsgroups/
[4] Available at http://www.daviddlewis.com/resources/testcollections/reuters21578/

4.2 Results

Our models achieve comparable quality results with values reported in the literature. It provides empirical confirmation for the model correctness. As expected, k-NN outperforms Naive-Bayes and the difference between micro and macro BEP for 20-newsgroups is minimal while it is significant for Reuters-21578 which have large differences between the number of documents per class.

Table 2. Efficiency/Scalability of Classifiers in PDatalog

	20-newsgroups		**Reuters-21578**	
	train(min)	test(s/doc)	train(min)	test(s/doc)
NB	37	0.543	11	0.480
k-NN	27	0.552	9	0.873

Efficiency results show that the application of both algorithms is possible with reasonable training and testing times in both collections.

5 Discussion & Future Work

This paper has shown the benefits of modelling classifiers using a descriptive approach. The compact high-level definitions and the use of a predicate dictionary leads to a flexible framework where expert users can model specific strategies with minimum engineering effort. In addition, it allows to prove the correctness of the models due to the fact that the abstraction makes possible to translate from the modelling in PDatalog to a mathematical formulation. Proofs have been presented for illustrating how the modelling of k-NN and Naive-Bayes result in the same equations as the mathematical concept. Experimental results empirically shows that the quality results achieved by our approach are the same as in the literature and that it is feasible to be used in real-scale environments.

Future work includes the modelling of more competitive text classification algorithms (i.e. SVM) that are not possible to be modelled at the moment, mainly because of the inexistence of optimisation expressions in PDatalog. With respect to correctness checking, the next step will be the investigation of an automatic derivation of mathematical expressions from a PDatalog program.

References

1. C. Cumbo, S. Iiritano, and P. Rullo. Reasoning-Based Knowledge Extraction for Text Classification. In *Proceedings of the 7th International Conference on Discovery Science (DS'04)*, pages 380–387, 2004.
2. J. Eisner, E. Goldlust, and N. A. Smith. Compiling Comp Ling: practical weighted dynamic programming and the Dyna language. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP'05)*, pages 281–290, 2005.

3. J. F. Forst, A. Tombros, and T. Roelleke. POLIS: A Probabilistic Logic for Document Summarisation. In *Proceedings of the 1st International Conference on the Theory of Information Retrieval (ICTIR'07)*, pages 201–212, 2007.
4. I. Frommholz and N. Fuhr. Probabilistic, object-oriented logics for annotation-based retrieval in digital libraries. In *Proceedings of Joint Conference on Digital Libraries (JCDL'06)*, pages 55–64, 2006.
5. N. Fuhr. Probabilistic Datalog - a logic for powerful retrieval methods. In *Proceedings of the 18th ACM SIGIR Conference on Research and development in information retrieval (SIGIR'95)*, pages 282–290, 1995.
6. M. Gelfond, N. Rushton, and W. Zhu. Combining Logical and Probabilistic Reasoning. In *Proceedings of AAAI 06 Spring Symposium*, pages 50–55, 2006.
7. A. Hunter and W. Liu. A survey of formalisms for representing and reasoning with scientific knowledge. volume 25, pages 199–222, 2010.
8. J. W. Lloyd. Practical Advantages of Declarative Programming. In *Proceedings of Joint Conference on Declarative Programming (GULP-PRODE'94)*, 1994.
9. A. Lopez. Translation as Weighted Deduction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL'09)*, pages 532–540, 2009.
10. A. McCallum and K. Nigam. A Comparison of Event Models for Naive Bayes Text Classification. In *Workshop on Learning for Text Categorization in AAAT/ICML'98*, page 41, 1998.
11. C. Meghini, F. Sebastiani, U. Straccia, and C. Thanos. A model of information retrieval based on a terminological logic. In *ACM SIGIR conference on Research and development in information retrieval*, pages 298–307, 1993.
12. H. Nottelmann. PIRE: An Extensible IR Engine Based on Probabilistic Datalog. In *Proceedings of the European Conference on Information Retrieval (ECIR'05)*, pages 260–274, 2005.
13. H. Nottelmann and N. Fuhr. Learning Probabilistic Datalog Rules for Information Classification and Transformation. In *Proceedings of International conference on Information and Knowledge Management (CIKM'01)*, pages 387–394, 2001.
14. L. D. Raedt, A. Kimmig, and H. Toivonen. ProbLog: a probabilistic Prolog and its application in link discovery. *IJCAI'07*, pages 2468–2473, 2007.
15. T. Roelleke and N. Fuhr. Information retrieval with probabilistic Datalog. In F. Crestani, M. Lalmas, and C. J. Rijsbergen, editors, *Uncertainty and Logics - Advanced models for the representation and retrieval of information*. Kluwer Academic Publishers, 1998.
16. 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.
17. T. Rolleke, R. Lubeck, and G. Kazai. The HySpirit retrieval platform. In *Proceedings of the 24th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 454, New York, NY, USA, 2001. ACM.
18. F. Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34:1–47, March 2002.
19. W. Shen, A. Doan, J. F. Naughton, and R. Ramakrishnan. Declarative information extraction using datalog with embedded extraction predicates. In *VLDB '07: International conference on Very large data bases*, pages 1033–1044, 2007.

A Mathematical Translation of PDatalog Expressions

Table 3 shows the mathematical translations of the PDatalog expressions that have been used in this paper. Let r_a, r_b be relations; let A and B be attributes sets and *m(A)* be the set of values v assigned to each attribute in A.

Table 3. Mathematical Translations of PDatalog Expressions

PDatalog expression	Mathematical Formulation
r_a FIRST(A) :- $r_b(B)$;	$r_b(B)_1$
r_a SUM(A) :- $r_b(B)$;	$\sum_{m(B)\subseteq m(A)} r_b(B)$
r_a PROD(A) :- $r_b(B)$;	$\prod_{m(B)\subseteq m(A)} r_b(B)$
r_a (A) :- $r_b(B)$|DISJOINT(K);	$\frac{r_b(B)}{\sum_{m(K)\subseteq m(B)} r_b(B)}$
r_a (A) :- $r_b(B)$|EUCLIDEAN(K);	$\frac{r_b(B)}{\sqrt{\sum_{m(B)\subseteq m(K)} r_b(B')^2}}$

B Predicate Dictionary Specification

```
term(T, D): Occurrences of terms in documents

part_of(D, C): Document-class labels

p_t_c_min(T, C): P(t|c) using minimum probability smoothing
minProbability() has to be specified

p_c_d_bayes(C, D):  Score for class-document using Naive-Bayes classifier
p_t_c(T, D) has to be specified for the estimation of P(t|c)

cosine(D1, D2): Similarity score based on cosine distance
final_test_weight(T, D) is needed for measuring the importance of terms in test documents
final_weight(T, D) is needed for computing the importance of terms in train documents

score_knn(C, D): Score for class-document using k-NN classifier
top_similarity(D1, D2) is needed modelling the k most similar documents.
```