Modelling Probabilistic Inference Networks and Classification in Probabilistic Datalog

Miguel Martinez-Alvarez[1] and Thomas Roelleke[1]

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

Abstract. Probabilistic Graphical Models (PGM) are a well-established approach for modelling uncertain knowledge and reasoning. Since we focus on inference, this paper explores Probabilistic Inference Networks (PIN's) which are a special case of PGM. PIN's, commonly referred as Bayesian Networks, are used in Information Retrieval to model tasks such as classification and ad-hoc retrieval. Intuitively, a probabilistic logical framework such as Probabilistic Datalog (PDatalog) should provide the expressiveness required to model PIN's. However, this modelling turned out to be more challenging than expected, requiring to extend the expressiveness of PDatalog. Also, for IR and when modelling more general tasks, it turned out that 1st generation PDatalog has expressiveness and scalability bottlenecks. Therefore, this paper makes a case for 2nd generation PDatalog which supports the modelling of PIN's. In addition, the paper reports the implementation of a particular PIN application: Bayesian Classifiers to investigate and demonstrate the feasibility of the proposed approach.

1 Introduction

1.1 Motivation and Background

Nowadays, there is a big productivity challenge when designing customisable IR systems which are usually developed for specific cases, having to rewrite a high portion of the original code for other purposes. This problem in IR is comparable with that happened in the Software Industry, when Software Engineering evolved from programs focused in one specific context to the developing of frameworks for general tasks that could be adapted for specific ones. We propose a generic module for PIN's based on probabilistic logic. This concept would provide a generic framework for any task that requires its use. Furthermore, thanks to the logical implementation, the functionality would be defined in a high-level, making it more understandable. In addition, we show the implementation of some classifiers and proof that the module could be adapted for specific cases. PIN's are a mechanism that allows modelling knowledge and reasoning.

Figure 1 presents the famous example [10] about burglary, earthquake, and alarm. It shows the network expressing that a burglary implies an alarm, and so does an earthquake. Indeed, there is also an implication from earthquake to burglary (since burglaries tend to happen during and after earthquakes). This arc increases the complexity of the network, since the event earthquake implies alarm via two different paths.

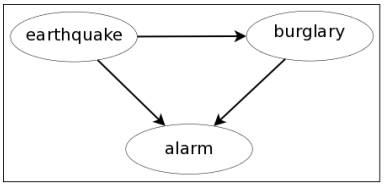

Fig. 1. PIN representing the hypothesis of an alarm being triggered in case of burglary/earthquake

The theory around PIN's has been influential in many domains, including artificial intelligence and information retrieval. However, the latter tend to use more complex representations. This is because in IR the application is large-scale in the sense that there is a PIN for each document to retrieve. [17] describe the dominant approach of how IR is modelled in a PIN.

Probabilistic inference also impacted the logical approach to IR [18, 19, 6], and the approach to utilise a probabilistic version of Datalog to model IR [5].

Intuitively, PDatalog should allow to model a PIN and we report in this paper that this intuition is true to a certain degree, but in "real-world" applications, the modelling is usually much more complex than initially expected.

Moreover, since the PIN theory is a natural candidate to model classification, we investigate the modelling of classifiers in 2nd generation PDatalog, reporting results on expressiveness, processing issues, and quality measures.

1.2 Structure and Contributions

The remainder of this paper is structured as follows: Section 2 reviews PIN's and their application in IR. Section 3 reviews PDatalog, reflecting on the 1st generation [5], and the 2nd generation (which incorporates the relational Bayes [13]). Section 4 binds the sections on PIN's and PDatalog: Modelling PIN's in PDatalog. The main contributions in this first part of the paper are the introduction and discussion of the 2nd generation PDatalog and the modelling of PIN's. Then, Section 5 presents a specific application of PIN's: Bayesian classifiers. Section 6 focuses on the modelling of Bayesian classifiers in PDatalog. The contribution of the sections on classification is to study how PDatalog copes with this concrete task. Finally, section 7 presents study of feasibility and experiments, showing that the implementation achieves quality levels that could be expected for other approaches.

2 Probabilistic Inference Networks

Probabilistic Inference Networks (PIN's), also referred as Bayesian Networks, are one of the most established technique for different IR and AI tasks. PIN's are used for representing conditional probabilities between different events. The definition of a PIN can be formulate as:

Definition 1. *A PIN is a directed acyclic graph (DAG). Let* (N, V) *denote a PIN where* N *is the set of nodes and* V *is a set of arcs, where an arc is a pair* (n_i, n_j)*, and* n_i *and* n_j *are nodes. For each node, there is a so-called conditional dependence probability (CDP) matrix. This matrix represents the probability* $P(n_i|parents_i)$*.*

This technique allows to represent and use for reasoning conditional probabilities between different events.

2.1 PIN-based Modelling of IR

[17, 16] utilised the PIN framework to investigate how the ranking provided by IR models (e.g. TF-IDF) can be explained via a PIN and its interpretation. The PIN model is a formal framework which infers the probability that each document in the collection satisfies the user's information need.

It uses document representations as sources of evidence about its content and multiple query representations as source of information need. In addition, it provides representation nodes that reflect different concepts in the model.

An inference network model (taken from [16]) is presented in figure 2. It contains four different type of nodes and the conections (with a weight assigned) between them. The links between nodes d and r are not shown in the diagram for clarity reasons:

- Document nodes (d): Representing documents in the corpus
- Representation nodes (r): Modelling the concepts considered (terms, phrases,...)
- Query nodes (q): They are related to parts of the information needed by the user
- Information need (I): It models the complete information needed by the user

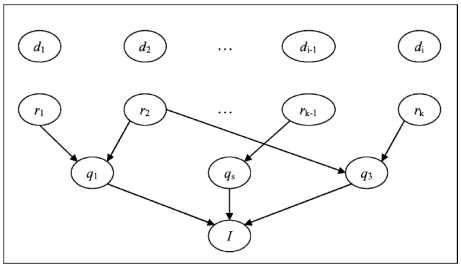

Fig. 2. Inference Network Model

This model represents the probabilities of an event class respect all the possible combinations in its parents values. It uses a link matrix known as conditional dependence probabilities (CDP), an example of an event with respect two parents is shown in equation 1.

$$L = \left[\begin{array}{c|cccc} & x_3 & x_2 & x_1 & x_0 \\ & 11 & 10 & 01 & 00 \\ \hline P(q|x) & a_{11} & a_{12} & a_{13} & a_{14} \\ P(\bar{q}|x) & a_{21} & a_{22} & a_{23} & a_{24} \end{array}\right] \tag{1}$$

One of the problems of this model is that the number of combinations exponentially rises making its processing extremely expensive in terms of computational power. Therefore, this model can not be directly applied to problems with a large quantity of features such as *ad hoc* retrieval or text classification.

As a solution for the problems of the original PIN model, a modification was designed by Turtle and Croft [16]. They assumed independence between terms and defined a special setting of link matrix with a normalization over the total weight of the features. These modifications lead to the equation 2. Using this approach, the model only needs the probability of being and not being in a class respect the existence of each term.

$$P(q|d) = \sum_t \frac{P(q|r)}{\sum_{t'} P(q|r')} \cdot P(r|d) \tag{2}$$

The reason for the normalization in equation 2 is that the sum over $P(q|r)$ for the terms/representations in a query could be greater than one. Therefore, $P(q|d)$ could be it as well.

3 Probabilistic Datalog

PDatalog is a probabilistic logical retrieval framework that combines deterministic Datalog (a query language used in deductive databases) and probability theory ([4, 12]). It was extended in [13, 20] 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. We utilise PDatalog here for implementing an abstraction layer for modelling PIN's. Moreover, it is also used for the implementation, as a proof of concept, of Bayesian classifiers.

3.1 1st Generation PDatalog

The 1st generation PDatalog uses free Horn clauses with a probability attached to rules and facts. It was introduced for IR in [5]. The main idea is to allow for probabilities in facts and rules. Figure 3 describes the syntax utilized in traditional datalog and 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.

Traditional Datalog	
fact	::= NAME '(' constants ')'
rule	::= head ':-' body
head	::= goal
body	::= subgoals
goal	::= NAME '(' args ')'
subgoal	::= pos_subgoal | neg_subgoal
pos_subgoal	::= atom
neg_subgoal	::= '!' atom
atom	::= NAME '(' args ')'
arg	::= constant | variable
constant	::= NAME | STRING | NUMBER
variable	::= VAR_NAME
args	::= | arg ',' args
constants	::= | constant ',' constants
subgoals	::= | subgoal ',' subgoals

1st Generation Probabilistic Datalog	
prob_fact	::= prob fact
prob_rule	::= prob rule

2nd Generation Probabilistic Datalog	
goal	::= tradGoal | bayesGoal | aggGoal
subgoal	::= tradSubgoal | bayesSubgoal | aggGoal
tradGoal	::= see 1st Generation
tradSubgoal	::= see 1st Generation
bayesGoal	::= tradGoal '|' {estAssump} evidenceKey
bayesSubgoal	::= tradSubgoal '|' {estAssump} evidenceKey
evidenceKey	::= '(' variables ')'
aggGoal	::= NAME {aggAssump} '(' args ')'
aggSuboal[1]	::= NAME {aggAssump} '(' args ')'
tradAssump	::= 'DISJOINT' | 'INDEPENDENT' | 'SUBSUMED'
irAssump	::= 'DF' | 'TF' | 'MAX_IDF' | 'MAX_ITF' | ...
probAssump	::= tradAssump | irAssump
algAssump	::= 'SUM' | 'PROD'
aggAssump	::= probAssump probAssump | complexAssump

Fig. 3. PDatalog syntax

3.2 2nd Generation PDatalog

The 2nd generation PDatalog includes a more complex syntax allowing assumptions and probability estimation. These modifications include, among others, score agregation (SUM, PROD) for the facts following certain patterns. For example, given "grade(Mike, A, DCS225); grade(Mike, B, DCS115); grade(Mike, A, DCS111);", a rule can be defined for computing $P(grade|student)$:

"$p\_grade\_student SUM(Grade, Student) : -grade(Student, Grade, Module)|(Student)$;"

This line uses probability estimation and an aggregation assumption (SUM) which is needed for the score agregation. A simplified version (for improving readability) of the syntax specification for the 2nd generation PDatalog is outlined in Figure 3. 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 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 (optimisation), complex assumptions can be specified. The decomposition of complex assumptions is shown in [13].

4 Modelling PIN in PDatalog

In this section we explain the modelling of PIN's in PDatalog, showing one example for each of its generations (Figure 4 for the 1st generation and Figure 5 for the 2nd). The former illustrates how can we represent prior and conditional probabilities while the latter represents all the input information about earthquakes, burglaries and alarms being triggered. The main difference is that probabilistic rules are manually specified in the 1st generation whereas probability estimation is being used in 2nd gen. In the second case we model $P(region|burglary)$ and $P(alarm \wedge burglary)$ for each region using the relational bayes as en example of probability estimation.

According to the first example, the probability of an alarm being triggered if there is a burglary is 90% while it is 40% in case of an earthquake. In addition, the example represent the prior probability of a burglary (0.1%) and an earthquake (0.001%). Due to the limitations of 1st generation the representation of all possible combinations between any parents and their son is needed.

```
0.001 hypo(burglary);
0.00001 hypo(earquake);
0.1 hypo(burglary) :− hypo(earthquake);
0.9 evidence(alarm) :− hypo(burglary);
0.4 evidence(alarm) :− hypo(earthquake);
0.35 evidence(alarm) :− hypo2(burglary,earthquake);
0.55 evidence(alarm) :− hypo2(burglary,n_earthquake);
0.05 evidence(alarm) :− hypo2(n_burglary,earthquake);
0.05 evidence(alarm) :− hypo2(n_burglary,n_earthquake);
```

Fig. 4. Modelling PIN in 1st Generation PDatalog

```
event(burglary); #... 200 more facts similar this one representing 20 different crimes.
event(earthquake); #... 3 facts
event(alarm); #... 150 more facts representing 10 different alarms being triggered
event2(burglary, earthquake); #... 10 facts
event2(n_burglary, earthquake); #... 30 facts
event2(alarm, burglary); #... 35 facts
event2(n_alarm, burglary); #... 5 facts
event2(alarm, earthquake); #... 10 facts
event2(n_alarm, earthquake); #... 30 facts
...
p_event1_event2 SUM(Event1, Event2) :− event2(Event1, Event2)|(Event2);
...
```

Fig. 5. Modelling PIN in 2nd Generation PDatalog

5 Bayesian Classifiers

Bayesian classifiers are a set of different classifiers that uses the Bayes Theorem for inference knowledge (Equation 3). However, different models use a different event space for representation, different techniques for calculate certain probabilities or other assumptions. Applying the Bayes theorem we can calculate the probability of a class given a document, being d a document for classify and c one of the classes. This equation could be extended by referring $P(d|c)$ and P(d) to the terms inside document d (Equation 3). Finally, its numerator can be rewritten, applying equation 4.

$$P(c|d) = \frac{P(d|c) \cdot P(c)}{P(d)} = \frac{P(t_1, t_2, ..., t_n|c) \cdot P(c)}{P(t_1, t_2, ..., t_n)} \tag{3}$$

$$P(t_1, t_2, ..., t_n|c) = P(t_1|c) \cdot P(t_2|c, t_1) \cdot ...P(t_n|c, t_1, ...t_{n-1}) \tag{4}$$

5.1 Independence Assumption

The computational power required for Bayes inference exponentially grows with the number of features making this method difficult to apply in large scale environments.

One of the most common solutions for this problem, known as "Independence Assumption", is assuming independence between features given the context of a class. Applying this assumption, the join probability from the general equation for Bayesian classifiers (equation 4) is modified, leading to:

$$P(t_1, t_2, ..., t_n|c) = P(t_1|c) \cdot P(t_2|c)... \cdot P(t_n|c) \tag{5}$$

Assuming independence between features we can define the probability of a document being labelled in a class as follows, where n(t,d) is the number of times that word t appears in document d,

$$P(c|d) = \frac{P(c)}{P(d)} \cdot \prod_{t \in d} P(t|c)^{n(t,d)} \tag{6}$$

Classifiers that make this assumption are usually referred as Naive-Bayes, even if there are differences between them [8]. This is a common assumption that allows the application of this algorithms to larger collections. However, it could be not correct.

5.2 Uniform prior distribution assumption

We can erase $P(d)$ and $P(c)$ from equation 6 if we assume that they are uniformly distributed. In that case the prior probabilities of a document and a class are constants, not having any effect in the ranking. Therefore, they can be erased.

5.3 Multi-variate Bernoulli

In this model [8] we represent different features (i.e. terms) using a binary vector that indicates which features are present in which elements (i.e. documents). We can apply this model for classification using the general Bayes formula (equation 3) substituting the equations shown in this section. This model computes the class prior probability by the maximum likelihood estimate (Equation 7), assuming $P(d) = \frac{1}{|D|}$ for all documents and the document prior (Equation 8). In addition, $P(d|c)$ and $P(t|c)$ are specified in equations 9 and 10 respectively where B_t is the binary value indicating if term t appears in document d.

$$P(c) = \frac{\sum_{d \in D} P(c|d)}{|D|} \tag{7}$$

$$P(d) = \sum_{c \in C} P(c) \cdot P(d|c) \tag{8}$$

$$P(d|c) = \prod_{t \in V} (B_t \cdot P(t|c) + (1 - B_t) \cdot (1 - P(t|c))) \tag{9}$$

$$P(t|c) = \frac{\sum_{d \in D} B_{dt} \cdot P(c|d)}{\sum_{d \in D} P(c|d)} \tag{10}$$

This model applies the naive independence assumption explained in section 5.1 and it explicitly takes into account the non-occurrence probability of features that are not in the element.

5.4 Multinomial

This model, explained in [8], uses a non-binary vector for representing different features. It uses the frequency of each parameter (i.e. term) for each element (i.e. document).

We can apply this model for classification using the general Bayes formula (equation 3) substituting the equations shown in this section. It computes the probability of a document given a class using Equation 11, where $n(t, d)$ represents the number of times term t occurs in document d and V is the set of terms contained in the document. The definition of $P(t|c)$ in Multinomial-Bayes is illustrated in equation 12.

$$P(d|c) = P(|d|) \cdot |d|! \prod_{t \in V} \frac{P(t|c)^{n(t,d)}}{n(t,d)!} \tag{11}$$

$$P(t|c) = \frac{\sum_{d \in D} n(t, d) \cdot P(c|d)}{|V| + \sum_{t \in V} \sum_{d \in D} n(t, d) \cdot P(c|d)} \tag{12}$$

Class and document priors are computed as they were in the Bernoulli model, applying equations 7 and 8 respectively.

6 Modelling Bayesian Classifiers in PDatalog

The modelling of Bayesian classifiers in PDatalog is a proof of concept regarding the expressiveness of PDatalog. This section outlines the case for Naive-Bayes, and then underlines that PDatalog programs are the result of a translation process, namely the translation of a PIN/BN specification to PDatalog. This translation frees the developer from actually "writing" PDatalog.

6.1 Naive-Bayes Classifier

Figure 6 shows a PDatalog program for modelling Naive-Bayes. This program uses as input facts tuples representing terms in training documents (termDoc_sample), terms contained in documents to classify (termDoc_classify) and the class labelled for each training document (part_of) (i.e. *0.2 termDoc_sample(car, d1); 1.0 part_of(d1, earn);*)

First of all, we define termClass(Term, Class), the representation of classes as it is derived from the members of the class and the prior probability of classes prior(Class). Secondly, It specifies the rule for predicate p_t_c, to model the feature likelihood $P(t|c)$. The expressiveness of 2nd-generation PDatalog supports the description of this step, namely the estimation of the feature probability. For IR, termClass(Term, Class) will be term-based representation of the classes derived from the documents that are part of the class. Then, there is a rule describing conditional probabilities of a document given a class, $P(d|c)$. Finally, it shows the probability of a class given a document using the expression $P(c|d)$.

```
1 prior(Class) :− part_of(Doc, Class) | ();
2 termClass(Term, Class) :− termDoc_sample(Term, Doc) & part_of(Doc, Class);
3 p_t_c SUM(Term,Class) :− termClass(Term, Class) | (Class);
4 p_d_c PROD(Doc, Class) :− termDoc_classify(Term, Doc) & p_t_c(Term, Class);
5 p_c_d(Class, Doc) :− p_d_c(Doc, Class) & prior(Class);
```

Fig. 6. Naive-Bayes Classifier in PD

6.2 Turtle-Croft-PIN-based Classifier

Figure 7 shows a PDatalog program for using the PIN model as a classifier. It follows the same notation and input data explained for the Naive-Bayes classifier in section 6.1.

6.3 Generation of PDatalog Classifier Programs

A PDatalog program representing a classifier can be viewed as the result of translating a PIN specification into a PDatalog program. This is currently a manual process.

```
1 termClass(Term, Class) :− termDoc_sample(Term, Doc) & part_of(Doc, Class);
2 p_t_c SUM(Term, Class) :− termClass(Term, Class) | (Class);
3 p_d_t SUM(Doc, Term) :− termDoc_classify(Term, Doc) | (Term);
4 p_d_c SUM(Class, Doc) :− p_d_t(Doc, Term) & p_t_c(Term, Class);
```

Fig. 7. Turtle-Croft-PIN-based Classifier in PD

The future idea is to automatically generate PDatalog programs for specific problems. We can use generic definitions of Bayesian classifiers that could be adapted to a single problem using a mapping layer. This layer would be a connection between the abstract implementation of classifiers and some specific facts representing the problem to be solved. In addition, this strategy would be extended to other classification algorithms, creating an abstraction layer for classification in PDatalog.

7 Feasibility and Experimental Study

The main focus at this stage of research is the feasibility of modelling PINs using PD. However, we also present, as a proof of concept, the quality evaluation of our approach for the Enron collection.

7.1 Collection

The Enron collection [1] contains emails from many of the senior management of Enron Corporation. This data was made public by SRI after clean-up and attachments removal. This corpus has a large number of emails, although there are many users with folders almost empty. The final corpus used in this paper is a subset of this collection. It contains the emails of seven employees with a large number of folders and mails. In addition, non-topical folders such as "all_documents", "calendar" and "contacts" were removed. After this, folder hierarchies were flatten and folders with less than three messages were deleted. Finally, the "X-folder" field in mail headers were removed as it contains the class label. This collection was obtained from http://www.cs.umass.edu/∼ronb/enron_dataset.html.

7.2 Training splits

The most common strategy to divide a collection between train an test documents for classification is using random splits. However, this method could create unnatural dependencies of earlier documents in latter documents for email foldering because this collections could have strong chronological dependences. Some authors have recommended splits based on the chronological order of the emails, using incremental splits [1] or using only one big split using half of the collections as a training set [7]. The former strategy can report non-realistic high rates for quality measures whereas the latter has the problem that some classes could have documents only in one of the splits.

We propose a modification of the second strategy applying a chronological split for each of the classes. By doing this we represent chronological dependencies and we guarantee that all the classes have documents in both splits. The focus of this paper is not a comparison between different split algorithms. However, we have decide to do experiments using four different methods:

Global chronological split The train set is formed by the first $n/2$ emails, where n is the size of the collection.

Class chronological split The train set is formed by the first $n_i/2$ emails of each class, where n_i is the size of the class i.

Global random split The documents in the train set are randomly chosen from the collection until its size is $n/2$, where n is the size of the collection.

Class random split The documents in the train set are randomly chosen from each class until the number of emails selected is $n_i/2$, where n_i is the size of class i.

7.3 Document representation

For our experiments we have used a "bag of words" representation. In addition, we use the terms that appear in more than 75% of the classes as stopwords.

7.4 Measures

The measures used for evaluation are the macro and micro-averaged F_1. They measure the classifier's quality based on its precision and recall values for each class. Precision is defined as the ratio of correctly classified documents respect to the number of documents classified by the system, while recall calculates the ratio between correctly classified documents and total number of documents truly belonging to the class [15].

Macro-averaged is usually computed in two different ways. The "correct" method, according to [21], is calculate the F_1 values for each category and then compute the average. On the other hand, F_1 could be computed using the macro-averaged recall and precision. Both methods give different results and the "correct" one is often significantly lower than the "incorrect" one [21]. We are using in the experiments the first method described.

7.5 Results and Discussion

Figure 8 shows the macro-average F_1 values for Naive-Bayes (implementation is shown in figure 6) and PIN implementation in PDatalog. The NB code follows the definition in equation 6 assuming uniform document frequency. $P(t|d)$ is represented by the number of times t appear in d divided by the total number of terms in it. Figure 9 presents the micro-averageF_1 (best value for each user in bold). They illustrate this quality measures using four different split algorithms for the collection. The values shown for random splits represent the average of three different executions for each configuration.

	Global Random		Class Random		Global Chronological		Class Chronological	
Enron user	NB	PIN	NB	PIN	NB	PIN	NB	PIN
beck-s	0.4642	0.3546	**0.5099**	0.3546	0.2035	0.1346	0.4108	0.3170
farmer-d	0.6166	0.3949	**0.6169**	0.3615	0.4541	0.2167	0.3937	0.2630
kaminski-v	0.5893	0.3295	**0.6009**	0.3357	0.3619	0.2223	0.4086	0.2540
kitchen-l	0.4655	0.2823	**0.4799**	0.2522	0.2116	0.0534	0.3204	0.2462
lokay-m	0.7072	0.5188	**0.7276**	0.5262	0.5481	0.4994	0.6115	0.4132
sanders-r	0.6466	0.6010	**0.6580**	0.6206	0.4900	0.4508	0.4904	0.6362
williams-w3	0.6465	0.4055	0.6363	0.3631	0.8989	**0.8998**	0.5935	0.2612
Average	0.5908	0.4124	**0.6042**	0.4020	0.4526	0.3539	0.4613	0.3415

Fig. 8. Macro-Average F1 for different split strategies

	Global Random		Class Random		Global Chronological		Class Chronological	
Enron user	NB	PIN	NB	PIN	NB	PIN	NB	PIN
beck-s	0.5019	0.3656	**0.5377**	0.3672	0.2315	0.1076	0.4086	0.2978
farmer-d	0.7932	0.4700	**0.7983**	0.4261	0.5708	0.1531	0.6284	0.2612
kaminski-v	**0.6534**	0.3044	0.6514	0.3050	0.4176	0.2242	0.4955	0.2327
kitchen-l	0.5119	0.2164	**0.5333**	0.2016	0.1574	0.0279	0.3786	0.2051
lokay-m	0.8095	0.5677	**0.8163**	0.6546	0.6996	0.4683	0.7454	0.2981
sanders-r	0.6953	0.5387	**0.7156**	0.5830	0.5286	0.3384	0.5751	0.5392
williams-w3	0.9275	0.5798	0.9360	0.5926	0.9949	**0.9993**	0.9160	0.5757
Average	0.6990	0.4347	**0.7127**	0.4472	0.5143	0.3313	0.5925	0.3443

Fig. 9. Micro-Average F1 for different split strategies

As we can see, the choice of splitting algorithm significantly changes the quality measure assigned to it. The best performance for the Naive-Bayes implementation has been obtained, both for macro and micro-averaged, using the random split per class. The explanation for these results is that a random split per class uses different periods of time as an information source, representing the "meaning" of the folder in all the time considered. In addition, this split algorithm guarantees that all the classes have at least one representative in the test model. We consider that it could be beneficial doing more experiments for an exhaustive comparison between different splitting methods. However, such research is beyond the scope if this paper.

The F_1 value for the user "williams-w3" with a global chronological split is much higher than the ones obtained applying other strategies because using that configuration the test collection only has documents from two different classes.

Given the assumption that "macro-averaged is higher for classifiers that behave well for few positive training documents while micro-averaged is better in the opposite case" [15] and the figures shown in this paper, we can say that the classifiers considered perform slightly better for common classes in this specific case.

Results obtained by the PDatalog-based Naive-Bayes classifier are similar to the ones expected for other implementations, showing its feasibility. On the other hand, Turtle-Croft-PIN based classifier has obtained lower results than Naive-Bayes in almost all the executions.

8 Summary and Conclusions

This paper has presented the modelling of PIN's in 1st and 2nd generation PDatalog. In 1st generation PDatalog, probabilistic rules have to be used to model the conditional probabilities in PIN's. In addition, it has no means to express the probability estimation that is required to derive/learn a PIN from sample data, i.e. the "learning probabilities" process is 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.

The main contribution of this paper is to present and discuss the issues when modelling PIN's in PDatalog. Having used PDatalog for a decade, it was surprising to realise that the modelling of PIN's turned out to be much more challenging than could be expected, since, intuitively, a probabilistic logical framework could be expected to naturally provide what the modelling of PIN's requires. In addition to the conceptual and theoretical aspects of modelling PIN's, this paper contributes the modelling of classifiers in PDatalog, and an experimental study to confirm feasibility and investigate

the quality of a PDatalog-based implementation. We have demonstrated that it is possible to model PIN's in PDatalog. In addition, we adapted these models to the concrete task of text classification using a mapping layer, that would be automatically generated in the future. Furthermore, we have shown that this implementation achieves quality measures than could be expected by other implementations of Bayesian classifiers.

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 PIN's. It has been used as an intermediate processing layer for semantic/terminological logics in different IR tasks such as *ad-hoc* retrieval [9], annotated document retrieval [3] and summarization [2].

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 [11, 14]. 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. Therefore, it can be expected to have an impact beyond the flexible modelling of classification.

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.

The next objective in the creation of a logical framework for Information Retrieval is developing an abstraction layer for PIN's and classification in PDatalog. This module should include not only Naive-Bayes but also non-probabilistic classifiers such as Support Vector Machines (SVM) or k-NN. In addition, it should allow single and multilabel strategies applying flat or hierarchical classification. Furthermore, the module would be able to deal with large-scale classification tasks (in terms of documents and classes).

Moreover, we have started to work on design techniques (design patterns, UML-like) to assist the design, composition and test of PDatalog programs. This framework supports teams of knowledge engineers to efficiently ”plug-and-play“ components they need for solving complex scenarios as they occur in information management tasks.

References

1. R. Bekkerman et al Automatic categorization of email into folders: Benchmark experiments on enron and sri corpora. *Tech.rep Center for Intelligent Information Retrieval* 2004.
2. J. F. Forst, A. Tombros, and T. Roelleke. Polis: A probabilistic logic for document summarisation. In *Studies in Theory of Information Retrieval*, pages 201–212, 2007.
3. I. Frommholz. Annotation-based document retrieval with probabilistic logics. In *ECDL*, pages 321–332, 2007
4. N. Fuhr. Probabilistic datalog - a logic for powerful retrieval methods. In *ACM SIGIR*, pages 282–290, 1995
5. N. Fuhr. Optimum database selection in networked ir. In *NIR'96 (SIGIR)*, 1996.
6. A. Kheirbeck and Y. Chiaramella. Integrating hypermedia and information retrieval with conceptual graphs formalism. In *Hypertext - Information Retrieval - Multimedia, Synergieeffekte elektronischer Informationssysteme*, pages 47–60, 1995
7. B. Klimt and Y. Yang. The Enron corpus: A new dataset for email classification research. In *ECML*, pages 217–226. 2004
8. A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI/ICML-98 Workshop on Learning for Text Categorization*, page 41, 1998.
9. 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
10. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, San Mateo, California, 1988.
11. A. Polleres. From SPARQL to rules (and back). In *16th international conference on World Wide Web (WWW)*, pages 787–796. ACM, 2007.
12. T. Roelleke and N. Fuhr. Information retrieval with probabilistic datalog. In *Uncertainty and Logics - Advanced models for the representation and retrieval of information*. 1998.
13. 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*, 2009.
14. S. Schenk. A SPARQL semantics based on Datalog. In *KI '07: Proceedings of the 30th annual German conference on Advances in Artificial Intelligence*, pages 160–174, 2007.
15. F. Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, 2002.
16. H. Turtle and W. Croft. Efficient probabilistic inference for text retrieval. In *Proceedings RIAO 91*, pages 644–661, 1991.
17. H. Turtle and W. B. Croft. Inference networks for document retrieval. In *ACM SIGIR*, pages 1–24, New York, 1990.
18. C. J. van Rijsbergen. Towards an information logic. In *ACM SIGIR*, pages 77–86, 1989.
19. S. Wong and Y. Yao. On modeling information retrieval with probabilistic inference. *ACM Transactions on Information Systems*, 13(1):38–68, 1995.
20. H. Wu, G. Kazai, and T. Roelleke. Modelling anchor text retrieval in book search based on back-of-book index. In *SIGIR Workshop on Focused Retrieval*, pages 51–58, 2008.
21. Y. Yang. A study on thresholding strategies for text categorization. In *ACM SIGIR*, pages 137–145. Press, 2001.