r-grams: Relational Grams

Niels Landwehr and **Luc De Raedt**
Machine Learning Lab, Department of Computer Science
Albert-Ludwigs-University Freiburg
Georges-Köhler-Allee 79, 79110 Freiburg, Germany
{landwehr,deraedt}@informatik.uni-freiburg.de

Abstract

We introduce relational grams (r-grams). They upgrade n-grams for modeling relational sequences of atoms. As n-grams, r-grams are based on smoothed n-th order Markov chains. Smoothed distributions can be obtained by decreasing the order of the Markov chain as well as by relational generalization of the r-gram. To avoid sampling object identifiers in sequences, r-grams are generative models at the level of variablized sequences with local object identity constraints. These sequences define equivalence classes of ground sequences, in which elements are identical up to local identifier renaming. The proposed technique is evaluated in several domains, including mobile phone communication logs, Unix shell user modeling, and protein fold prediction based on secondary protein structure.

1 Introduction

Probabilistic sequence models occupy an important position within the field of machine learning. They are not only theoretically appealing but have also proven to provide effective learning algorithms across many application areas, ranging from natural language processing to bioinformatics and robotics. In traditional sequence models, sequences are strings $s = w_1 \dots w_k$ over a (finite) alphabet Σ. Typically, the goal is to estimate the joint distribution $\mathbf{P}(\mathbf{s})$ over all possible strings. Given $\mathbf{P}(\mathbf{s})$ several tasks can be solved, including sequence classification, sampling, or predicting the next event in a sequence given a history of preceding events.

Recent technological advances and progress in artificial intelligence has led to the generation of structured data sequences. One example is that of the smart phone, where communication events of many phone users have been logged during extended periods of time [Raento *et al.*, 2006]. Other ones are concerned with logging the activities and locations of persons [Liao *et al.*, 2005], or visits to websites [Anderson *et al.*, 2002]. Finally, in bioinformatics, sequences often contain also structural information [Durbin *et al.*, 1998]. These developments together with the interest in statistical relational learning (see [De Raedt and Kersting, 2003] for an overview) have motivated the development of probabilistic models for relational sequences. These are sequences of tuples or atoms such as that indicated in Example 1. The Relational Markov Model (RMM) approach by [Anderson *et al.*, 2002] extends traditional Markov models to this domain, and the Logical Hidden Markov Models (LOHMMs) by [Kersting *et al.*, 2006] upgrade traditional HMMs towards relational sequences and also allow for logical variables and unification. Motivated by these models and the success and simplicity of n-gram models, we introduce relational grams, which upgrade the traditional n-grams towards relational sequences. There are two distinguished features of the r-grams. First, as LOHMMs and RMMs they allow one to work with generalized, i.e. abstract atoms in the sequences. For instance, in the bioinformatics example the atom helix(right,alpha,short) can be generalized to helix(X,alpha,Y) to describe an alpha-helix of any orientation and length. Secondly, the use of variables and unification allows one to make abstraction of *object identifiers* and to share information between events. For example, the (abstract) sub-sequence outcall(X,fail), outtxt(X) describes that a user, after failing to reach a person, writes a text message to *the same person*, without stating the identity of the person. This is especially important when generalizing the patterns across phones or users, as the objects referred to will typically be different, and the precise identifiers do not matter but the relationships and events they occur in do.

The paper is structured as follows: in Section 2, we introduce relational sequences and define the notion of a generative model that takes into account the nature of identifiers and object identity; in Section 3, we then start from n-grams to derive r-grams; in Section 4, we report on some experiments, and finally, in Section 5, we conclude and touch upon related work.

2 Relational Sequences

An *atom* $p(t_1, \dots, t_n)$ is a relation p of arity n that is followed by n terms t_i. We will work with three kinds of terms: constants (in slanted font), identifiers (in *italic*) and variables (starting with an upper case character). Furthermore, the relations are *typed*, i.e. for each argument position i of the relation p, one can either have constants or identifiers. Variables may appear at any position. We shall also distinguish constant-variables from identifier-variables; the former can be instantiated to constants, the latter to identifiers. They will be written in $italic$ or slanted font respec-

Example 1. *The following sequences over atoms are examples for relational sequences from different domains:*

outcall(*116513*,fail), outcall(*116513*,fail), incall(*116513*,succ), outtxt(*213446*), intxt(*213446*), ...

mkdir(*rgrams*), ls(*rgrams*), emacs(*rgrams.tex*), latex(*rgrams.tex*), ...

strand(*sa*,plus,short), helix(right,alpha,medium), strand(*blb*,plus,short), helix(right,f3to10,short),...

The first domain describes incoming and outgoing calls and text messages for a mobile phone user. In the second domain, Unix shell commands executed by a user are described in terms of command name and arguments. In the third domain, helices and strands as protein secondary structure elements are defined in terms of their orientation, type, and length.

tively. For instance, the relation outcall has identifiers at the first position, and constants at the second one. Therefore, outcall(*111365*,fail) is type-conform. The types, constants, identifiers, variables and relations together specify the alphabet Σ, i.e. the set of type-conform atoms. $\bar{\Sigma} \subseteq \Sigma$ is the set of atoms that do not contain identifiers. A *relational sequence* is then a string $s = w_1, \ldots, w_m$ in Σ^*. An expression is *ground* if it does not contain any variable. Example sequences will typically be ground, cf. Example 1.

Notice that constants typically serve as attributes describing properties of the relations, and identifiers identify particular objects in the domain. Identifiers have no special meaning and they are only used for sharing object identity between events. Moreover, there is no fixed vocabulary for identifiers which is known a priori, rather, new identifiers will keep appearing when applying a model to unseen data. Therefore, it is desirable to distinguish ground sequences only up to identifier renaming, which motivates the following definition.

Definition 1 (Sequence Congruence)**.** *Two relational sequences $p_1 \ldots p_m$, $q_1 \ldots q_m$ are* **n-*congruent*** *if for all $i \in \{1, \ldots, m-n\}$ the subsequences $p_i \ldots p_{i+n-1}$, $r_i \ldots r_{i+n-1}$ are identical up to identifier renaming. Two sequences s_1, s_2 are identical up to identifier renaming if there exist a one-to-one mapping φ of the identifiers in s_1 to those in s_2 such that s_1 equals s_2 after replacing the identifiers according to φ.*

Example 2. *The sequences* r(x)p(a,y)r(y)p(b,x) *and* r(z)p(a,w)r(w)p(b,u) *are* 3*-congruent (but not* 4*-congruent).*

m-congruent sequences are identical up to identifier renaming. For $n < m$, the definition takes into account a limited history: two sequences are congruent if their object identity patterns are locally identical. Finally we note that n-congruence defines an equivalence relation, i.e. it is reflexive, symmetric and transitive.

Let us now define a generative model for relational sequences. It should not sample actual identifier values, but rather equivalence classes of congruent sequences. This yields the following definition:

Definition 2 (Generative Model)**.** *Let Σ be a relational alphabet. Let $S(\Sigma)$ be the set of all ground sequences of length m over Σ, and let $\mathbb{S}_n(\Sigma)$ be the set of equivalence classes induced on $S(\Sigma)$ by n-congruence. Then a* ***generative model*** *of order n over Σ defines a distribution over $\mathbb{S}_n(\Sigma)$.*

Such a generative model can be learned from a set of ground sequences and the alphabet Σ. In its simplest form, the learning problem can be stated as maximum likelihood estimation:

Given

- a relational alphabet Σ
- a set S of ground sequences over Σ
- $n \in \mathbb{N}$
- a family Λ of generative models of order n over Σ,

Find a model $\lambda \in \Lambda$ that maximizes

$$P(S \mid \lambda) := \prod_{s \in S} P([s] \mid \lambda)$$

where $[s]$ denotes the equivalence class of s with regard to n-congruence. A simple instance of such a family of generative models will be introduced in the next section.

3 r-gram Models for Relational Sequences

Markov Chains are amongst the simplest yet most successful approaches for sequence modeling. *n-grams* are based on higher-order Markov chains, and employ certain smoothing techniques to avoid overfitting the sample distribution. In this section, we will briefly review n-grams, and then propose a simple extension of n-grams for relational sequences.

3.1 n-grams: Smoothed Markov Chains

n-grams define a distribution over sequences $w_1 ... w_m$ of length m by an order $n-1$ Markov assumption:

$$P(w_1...w_m) = \prod_{i=1}^{m} P(w_i \mid w_{i-n+1}...w_{i-1})$$

(where w_{i-n+1} is a shorthand for $w_{max(1,i-n+1)}$). In the most basic case, the conditional probabilities are estimated from a set S of training sequences in terms of "gram" counts:

$$P_n(w_i \mid w_{i-n+1} \ldots w_{i-1}) = \frac{C(w_{i-n+1} \ldots w_i)}{C(w_{i-n+1} \ldots w_{i-1})} \qquad (1)$$

where $C(w_1 ... w_k)$ is the number of times $w_1 ... w_k$ appeared as a subsequence in any $s \in S$. Note that this is indeed the estimate maximizing the likelihood.

The gram order n defines the tradeoff between reliability of probability estimates and discriminatory power of the model. Rather than selecting a fixed order n, performance can be increased by combining models of different order which are discriminative but can still be estimated reliably [Manning and Schütze, 1999]. The two most popular approaches are back-off and interpolation estimates. In this paper, we will focus on the latter approach, which defines conditional distributions as linear combinations of models of different order:

$$P(w_i \mid w_{i-n+1} \dots w_{i-1}) = \sum_{k=1}^{n} \alpha_k P_k(w_i \mid w_{i-k+1} \dots w_{i-1}) \tag{2}$$

where the $\alpha_1, ..., \alpha_n$ are suitable weights with $\sum_{k=1}^{n} \alpha_k = 1$, and the lower-order distributions $P_k(w_i \mid w_{i-k+1} \dots w_{i-1})$ are estimated according to maximum likelihood (Equation 1). Several more advanced smoothing techniques have been proposed (cf. [Manning and Schütze, 1999]), but are beyond the scope of this paper.

3.2 r-grams: Smoothed Relational Markov Chains

Consider ground relational sequences of the form $g_1 \dots g_m \in S(\Sigma)$. The key idea behind r-grams is a Markov assumption

$$P(g_1...g_m) = \prod_{i=1}^{m} P(g_i \mid g_{i-n+1}...g_{i-1}). \tag{3}$$

However, defining conditional probabilities at the level of ground grams does not make sense in the presence of object identifiers. Thus, ground grams will be replaced by generalized ones. Generality is defined by a notion of subsumption:

Definition 3 (Subsumption). *A relational sequence $l_1, \dots, l_k$ subsumes another sequence $k_1, \dots, k_n$ with substitution θ, notation $l_1, \dots, l_k \preceq_\theta k_1, \dots, k_n$, if and only if $k \leq n$ and $\forall i, 1 \leq i \leq k : l_i\theta = k_i$. A substitution is a set $\{V_1/t_1, \dots, V_l/t_l\}$ where the V_i are different variables and the t_i are terms such that no identifier or identifier variable occurs twice in $\{t_1, ..., t_l\}$.*

The restriction on the allowed substitutions implements the object identity subsumption of [Semeraro *et al.*, 1995]. The notion of sequence subsumption is due to [Lee and De Raedt, 2003]. It can be tested in linear time.

Example 3. *Let Y, Z, U, V be identifier variables.*

$$\mathsf{r}(X)\mathsf{p}(\mathsf{a}, Y)\mathsf{r}(Y) \preceq_{\theta_1} \mathsf{r}(w)\mathsf{p}(\mathsf{a}, u)\mathsf{r}(u)$$
$$\mathsf{r}(X)\mathsf{p}(\mathsf{a}, Y)\mathsf{r}(Z) \preceq_{\theta_2} \mathsf{r}(W)\mathsf{p}(\mathsf{a}, U)\mathsf{r}(V)$$

but

$$\mathsf{r}(X)\mathsf{p}(\mathsf{a}, Y)\mathsf{r}(Z) \not\preceq_{\theta} \mathsf{r}(W)\mathsf{p}(\mathsf{a}, U)\mathsf{r}(U).$$

with $\theta_1 = \{X/w, Y/u\}$ and $\theta_2 = \{X/W, Y/U, Z/V\}$.

We can now refine equation 3 to take into account generalized sequences. This will be realized by defining

$$P(g_1...g_m) = \prod_{i=1}^{m} P(l_i \mid l_{i-n+1}...l_{i-1})$$

where $l_{i-n+1}...l_i \preceq_\theta g_{i-n+1}...g_i$. This generalization abstracts from identifier values and at the same time yields smoothed probability estimates, with the degree and characteristic of the smoothing depending on the particular choice of $l_{i-n+1}...l_i$. This is formalized in the following definition.

Definition 4 (r-gram model). *An **r-gram model** R of order n over an alphabet Σ is a set of relational grams*

$$l_n^1 \vee ... \vee l_n^d \leftarrow l_1...l_{n-1}$$

where

1. *$\forall i : l_1...l_{n-1}l_n^i \in \bar{\Sigma}^*$;*
2. *$\forall i : l_n^i$ contains no constant-variables;*
3. *$\forall i : l_n^i$ is annotated with the probability values $P_r(l_n^i \mid l_1...l_{n-1})$ such that $\sum_{i=1}^{d} P_r(l_n^i \mid l_1...l_{n-1}) = 1$*
4. *$\forall i \neq j : l_1...l_{n-1}l_n^i \not\preceq_\theta l_1...l_{n-1}l_n^j$; i.e. the heads are mutually exclusive.*

Example 4. *The following is an example of an order 2 relational gram in the mobile phone domain (see Example 1).*

$$\left.\begin{array}{rl} 0.3 & \mathsf{outtxt}(X) \\ 0.05 & \mathsf{outtxt}(Y) \\ 0.2 & \mathsf{outcall}(X, \mathsf{fail}) \\ ... & \\ 0.05 & \mathsf{intxt}(Y) \end{array}\right\} \leftarrow \mathsf{outcall}(X, \mathsf{fail})$$

It states that after not reaching a person a user is more likely to write a text message to this person than to somebody else.

We still need to show that an r-gram model R defines a distribution over relational sequences. We first discuss a basic model by analogy to an unsmoothed n-gram, before extending it to a smoothed one in analogy to Equation 2.

A Basic Model

In the basic r-gram model, for any ground sequence $g_1...g_{n-1}$ there is exactly one gram $l_n^1 \vee ... \vee l_n^d \leftarrow l_1...l_{n-1}$ with $l_1...l_{n-1} \preceq_\theta g_1...g_{n-1}$. Its body $l_1...l_{n-1}$ is the most specific sequence in $\bar{\Sigma}^*$ subsuming $g_1...g_{n-1}$. According to Equation 3, we start by defining a probability $P_R(g \mid g_1...g_{n-1})$ for any ground atom g given a sequence $g_1...g_{n-1}$ of ground literals. Let g be a ground literal and consider the above gram r subsuming $g_1...g_{n-1}$. If there is an $i \in \{1, ..., d\}$ such that $l_1...l_{n-1}l_n^i \preceq_\theta g_1...g_{n-1}g$ it is unique and we define

$$P_R(g \mid g_1...g_{n-1}) := P_r(g \mid g_1...g_{n-1}) := P_r(l_n^i \mid l_1...l_{n-1})$$

Otherwise, $P_R(g \mid g_1...g_{n-1}) = 0$. From $P_R(g \mid g_1...g_{n-1})$, a probability value $P_R(g_1...g_m)$ can be derived according to Equation 3. Note that this is *not* a distribution over all ground sequences of length m, as the model does not distinguish between n-congruent sequences. Instead, the following holds:

Lemma 1. *Let R be an order n r-gram over Σ, and $s, s' \in S(\Sigma)$ be relational sequences with s n-congruent to s'. Then $P_R(s) = P_R(s')$.*

Let us therefore define $P_R([s]) := P_R(s)$ for any $[s] \in \mathbb{S}_n(Q)$. Furthermore, $\sum_{[s] \in \mathbb{S}_n(\Sigma)} P_R([s]) = 1$. Therefore,

Theorem 1. *An order n r-gram over Σ is a generative model over Σ.*

Example 5. *Consider the r-gram model R with grams*

$$\mathsf{p}(\mathsf{a}, X) \vee \mathsf{p}(\mathsf{b}, X) \leftarrow \mathsf{r}(X)$$
$$\mathsf{r}(X) \leftarrow \mathsf{p}(\mathsf{b}, X)$$
$$\mathsf{r}(X) \vee \mathsf{r}(Y) \leftarrow \mathsf{p}(\mathsf{a}, X)$$
$$\mathsf{r}(X) \leftarrow \epsilon$$

and uniform distributions over head literals. ϵ is an artificial start symbol that only matches at the beginning of the sequence. The ground sequence $g_1...g_5 =$ $\mathsf{r}(u)\mathsf{p}(\mathsf{a},u)\mathsf{r}(v)\mathsf{p}(\mathsf{b},v)\mathsf{r}(v)$ has probability $P_R(g_1...g_5) = 1 \cdot 0.5 \cdot 0.5 \cdot 0.5 \cdot 1 = 0.125$.

Smoothing r-grams

In the basic model, there was exactly one gram $r \in R$ subsuming a ground subsequence $g_1...g_{n-1}$, namely the most specific one. As for n-grams, the problem with this approach is that there is a large number of such grams and the amount of training data needed to reliably estimate all of their frequencies is prohibitive unless n is very small. For n-grams, grams are therefore generalized by shortening their bodies, i.e., smoothing with k-gram estimates for $k < n$ (Equation 2).

The basic idea behind smoothing in r-grams is to generalize grams logically, and mix the resulting distributions:

$$P_R(g \mid g_1...g_{n-1}) = \sum_{r \in \hat{R}} \frac{\alpha_r}{\alpha} P_r(g \mid g_1...g_{n-1})$$

where $P_r(g \mid g_1...g_{n-1})$ is the probability defined by r as explained above, $\hat{R}$ is the subset of grams in R subsuming $g_1...g_{n-1}$, and α is a normalization constant, i.e. $\alpha = \sum_{r \in \hat{R}} \alpha_r$. The more general r, the more smooth the probability estimate $P_r(g \mid g_1...g_{n-1})$ will be. The actual degree and characteristic of the smoothing is defined by the set of matching r-grams together with their relative weights α_r.

By analogy with n-grams, additional smoothing can be obtained by also considering relational grams $r \in R$ with shorter bodies $l_1...l_{k-1}$, $k < n$. However, there is a subtle problem with this approach: Relational grams of order $k < n$ define a probability distribution over $\mathbb{S}_k(\Sigma)$ rather than $\mathbb{S}_n(\Sigma)$, i.e. the sequences are partitioned into a smaller number of equivalence classes. However, this can be taken care of by a straightforward normalization, which distributes the probability mass assigned to an equivalence class modulo k equally among all subclasses modulo n.

Example 6. *Consider the r-gram model R with grams r,q given by*

$$r: \quad \mathsf{r}(X) \vee \mathsf{r}(Y) \leftarrow \mathsf{r}(X)$$
$$q: \quad \mathsf{r}(X) \leftarrow$$

uniform distribution over head literals and $\alpha_r = \alpha_q = 0.5$. We expect $P_R(r(u) \mid r(v)) + P_R(r(v) \mid r(v)) = 1$. However, when directly mixing distributions

$$P_R(\mathsf{r}(u) \mid \mathsf{r}(v)) = \alpha_r P_r(\mathsf{r}(Y) \mid \mathsf{r}(X)) + \alpha_q P_q(\mathsf{r}(X)) = 0.75$$
$$P_R(\mathsf{r}(v) \mid \mathsf{r}(v)) = \alpha_r P_r(\mathsf{r}(X) \mid \mathsf{r}(X)) + \alpha_q P_q(\mathsf{r}(X)) = 0.75,$$

as the r-gram q does not distinguish between the sequences $\mathsf{r}(x)\mathsf{r}(x)$ and $\mathsf{r}(x)\mathsf{r}(y)$. Instead, we mix by

$$P_R(\mathsf{r}(x) \mid \mathsf{r}(y)) = \alpha_r P_r(\mathsf{r}(X) \mid \mathsf{r}(X)) + \frac{1}{\gamma} \alpha_q P_q(\mathsf{r}(X))$$

where $\gamma = 2$ is the number of subclasses modulo 2-congruence of the class [r(X)].

The ultimate level of smoothing can be obtained by a relational gram r of the form $l_n^1 \vee ... \vee l_n^d \leftarrow$ where the l_n^i are fully variablized (also for non-identifier arguments). For this gram

$$P_r(g \mid g_1...g_{n-1}) = P_r(l_n^i) \prod_{j=1}^{a} P(X_j = x_j)$$

where $X_1, ..., X_a$ are the non-identifier arguments of l_n^i and $x_1, ..., x_a$ their instantiations in g. This case corresponds to an out-of-vocabulary event (observing an event that was not part of the training vocabulary) for n-grams.

3.3 Building r-grams From Data

To learn an r-gram from a given set S of training sequences, we need to 1) choose the set R of relational grams; 2) estimate their corresponding conditional probabilities; and 3) define the weights α_r for every $r \in R$. Before specifying our algorithm, we need to define counts in the relational setting:

$$C(l_1 \dots l_k) = |\{i | s_1 \dots s_m \in S \text{ and } l_1 \dots l_k \preceq_\theta s_i \dots s_{i+k}\}| \quad (4)$$

Our algorithm to learn r-grams is specified below:

```
r-grams(input: sequences $S$; alphabet: $\Sigma$; parameters: $\gamma, n$)
1  $B := \{l_1 \dots l_{k-1} \in \bar{\Sigma}^* | C(l_1 \dots l_{k-1}) > 0$ and $k \leq n\}$
2  for each $l_1 \dots l_{k-1} \in B$
3      do let $\{l_k^1 \dots l_k^d\}$ contain all maximally specific
             literals $l_k^i \in \bar{\Sigma}$ such that $C(l_1 \dots l_{k-1} l_k^i) > 0$
4         add $r = l_k^1 \vee \dots \vee l_k^d \leftarrow l_1 \dots l_{k-1}$ to $R$ with
             $P_r(l_k^i | l_1 \dots l_{k-1}) = \frac{C(l_1 \dots l_{k-1} l_k^i)}{C(l_1 \dots l_{k-1})}$
5         $L(r) := \prod_{g_1 \dots g_{n-1} g \in S(r)} P_r(g \mid g_1 \dots g_{n-1})$.
6         $t := |S(r)|$
7         $\alpha_r := L(r)^{\frac{\gamma}{t}}$
8  return $R$
```

In Line 1 of the algorithm, one computes all r-grams that occur in the data. Notice that no identifiers occur in these r-grams (cf. $\bar{\Sigma}^*$). This can be realized using either a frequent relational sequence miner, such as MineSeqLog [Lee and De Raedt, 2003], or using an on-the-fly approach. In the latter case, a relational gram with body $l_1...l_k$ is only built and added to R when and if it is needed to evaluate $P_R(g \mid g_1...g_{n-1})$ with $l_1...l_k \preceq g_1...g_k$ on unseen data. In Line 3, all possible literals l_k^i are sought that occur also in the data. They are maximally specific, which means that they do not contain constant-variables (cf. condition 2 of Definition 4). Line 4 then computes the maximum likelihood estimates and Lines 5–7 the weight α_r. Here S(r) denotes the set of all ground subsequences $g_1...g_{n-1}g$ appearing in the data which are subsumed by r. The likelihood $L(r)$ of r defined in Line 5 is a measure for how well the distribution defined by r matches the sample distribution. The α_r as in Line 7 is then defined in terms of $|S(r)|$ and the parameter γ, which controls the tradeoff between smoothness and discrimination. Highly discriminative (specific) rules have higher likelihood than more general ones as they are able to fit the sample distribution better, and thus receive more weight if $\gamma > 0$.

4 Experiments

This section reports on an empirical evaluation of the proposed method in several real-world domains. More specifically, we seek to answer the following questions:

(Q1) Are r-grams competitive with other state-of-the-art approaches for relational sequence classification?

(Q2) Is relational abstraction, especially of identifiers, useful?

Experiments were carried out on real-world sequence classification problems from three domains. In the **Unix** Shell domain [Greenberg, 1988; Jacobs and Blockeel, 2003],

Domain	r-grams	LOHMM	LOHMM + FK
Protein	83.3	74.0	82.7

Domain	r-grams	kNN	C4.5
Unix-50	93.8 ± 2.7	91.0	88.8
Unix-1000	97.2 ± 0.4	95.3	94.7

Table 1: Comparison of classification accuracy of r-grams to Logical Hidden Markov models and Fisher kernels in the Protein Fold domain, and to k-nearest neighbor and C4.5 in the Unix Shell domain. For protein fold prediction, a single split into training and test set is used. In the Unix Shell domain, 10 subsets of 50/1000 examples each are randomly sampled, accuracy determined by 10-fold cross-validation and averaged.

Domain	r-grams	n-grams	n-grams w/o IDs
Protein	83.3	79.7	83.3
Unix-50	93.8 ± 2.7	74.4	95.6 ± 2.7
Unix-1000	97.2 ± 0.4	76.3	97.1 ± 0.4
Phone I	95.0 ± 2.0	30.0	86.8 ± 3.3
Phone II	93.3 ± 14.9	33.3	86.7 ± 18.3

Table 2: Accuracy comparison of r-grams to n-grams, and to n-grams w/o IDs. For Protein/Unix domains settings are as before. For the Phone I domain, 5 subsets of size 100 have been sampled from the data, a 5-fold cross-validation is performed on each set and results are averaged. For Phone II, results are based on one 5-fold cross-validation. Results for **n-grams** are based on one sample only.

the task is to classify users as *novice programmers* or *non-programmers* based on logs of 3773 shell sessions containing 94537 commands (constants) and their arguments (identifiers). To reproduce the setting used in [Jacobs and Blockeel, 2003], we sampled 10 subsets of 50/1000 instances each from the data, measured classification accuracy on these using 10-fold cross-validation, and averaged the results. In the **Protein** fold classification domain, the task is to classify proteins as belonging to one of five folds of the SCOP hierarchy [Hubbard *et al.*, 1997]. Strand names are treated as identifiers, all other ground terms as constants. This problem has been used as a benchmark before [Kersting *et al.*, 2006; Kersting and Gärtner, 2004], and we reproduce the experimental setting used in this earlier work: the same 200 examples per fold are used for training, and the remaining examples as the test set. In the Context **Phone** domain, data about user communication behavior has been gathered using a software running on Nokia Smartphones that automatically logs communication and context data. In our study, we only use information about incoming and outgoing calls and text messages. Phone numbers are identifiers, other ground terms constants. The task in **Phone I** is to discriminate between real sequences of events and "corrupted" ones, which contain the same sequence elements but in random order. For $k \in \{20, 40, 60, 80, 100\}$, 5 subsets of size k were sampled randomly, 5-fold cross-validation performed and averaged for each k. In **Phone II**, the task is to classify communication logs as belonging to one of three users, based only on their communication patterns but without referring to actual phone numbers in the event sequence.

In all domains sequence classification is performed by building an r-gram model R_C for each class C and labeling unseen sequences s with the class that maximizes $P_C(s)P(C)$. We used bigram models in the Phone II domain and trigram models for all other domains, and the smoothing parameter γ was set to 1 in all experiments. Learning the r-gram model was done on-the-fly as explained in Section 3.3.

Table 1 compares the classification accuracy of r-grams with accuracy results from the literature in the Protein Fold and Unix Shell domains. In the Protein Fold domain, a handcrafted Logical Hidden Markov Model achieves 74% accuracy [Kersting *et al.*, 2006]. This has been improved to 82.4% by a fisher kernel approach, in which the gradient of the likelihood function of the Logical Hidden Markov Model is used

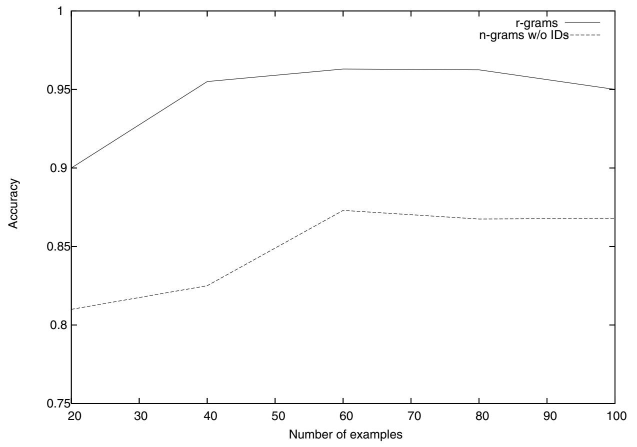

Figure 1: Accuracy for different training set sizes ranging from 20 to 100 examples in the Phone I domain. For each size, 5 sets are sampled, a 5-fold cross-validation is performed and the result averaged.

as input in a support vector machine [Kersting and Gärtner, 2004]. The Unix Shell log classification problem was originally tackled using a k-nearest neighbor method based on customized sequence similarity [Jacobs and Blockeel, 2003]. In the same paper, the authors present results for a C4.5 decision tree learner using a bag-of-words representation. In both cases, r-grams yield competitive classification accuracy, which is a positive answer to question **(Q1)**. Furthermore, we note that even using a naïve implementation r-grams are computationally efficient. Times for building an r-gram model ranged from 3 to 240 seconds in the presented experiments[1].

In a second set of experiments, the effect of using relational abstraction was examined in more detail. More precisely, r-grams were compared to n-grams which implement non-relational smoothing as outlined in Section 3.1, treating the atoms in Σ as flat symbols. For these experiments, we tried keeping identifiers in the events (**n-grams**) or removing them from the data (**n-grams w/o IDs**). Accuracy results for the Protein Fold, Unix Shell and Context Phone domains are

[1]All experiments were run on standard PC hardware with 3.2GHz processor and 2GB of main memory.

given in Table 2. If identifiers are treated as normal constants, accuracy is reduced in all cases, especially for the identifier-rich Unix and Context Phone domains. This is not surprising, as most identifiers appearing in the test data have never been observed in the training data and thus the corresponding event has probability zero. When identifiers are removed from the data, performance is similar as for r-grams in the Protein Fold and Unix Shell domain, but worse in the Context Phone domains. On Phone I, r-grams significantly outperform n-grams w/o IDs (unpaired sampled t-test, p = 0.01). Figure 1 shows more detailed results on Phone I for different numbers of training examples. It can be observed that relational abstraction is particularly helpful for small numbers of training examples. To summarize, there are domains where it is possible to ignore identifiers in the data, but in other domains relational abstraction is essential for good performance. Therefore, question **(Q2)** can be answered affirmatively as well.

5 Conclusions and Related Work

We have presented the first approach to upgrading n-grams to relational sequences involving identifiers. The formalism employs variables, unification and distinguishes constants from identifiers. It also implements smoothing by relationally generalizing the r-grams. The approach was experimentally evaluated and shown to be promising for applications involving the analysis of various types of logs.

The work that we have presented is related to other work on analyzing relational and logical sequences. This includes most notably, the RMMs by [Anderson *et al.*, 2002] who presented a relational Markov model approach, and the LOHMMs by [Kersting *et al.*, 2006] who introduced Logical Hidden Markov Models. RMMs define a probability distribution over ground relational sequences *without identifiers*. Furthermore, they do not employ unification (or variable propagation) but realize shrinkage through the use of taxonomies over the constant values appearing in the atoms and decision trees to encode the probability values. RMMs only consider first order Markov models, i.e. the next state only depends on the previous one, so RMMs do not smooth over sequences of variable length. RMMs have been applied to challenging applications in web-log analysis. Whereas RMMs upgrade Markov models, LOHMMs upgrade HMMs to work with logical sequences. As r-grams, they allow for unification and offer also the ability to work with identifiers. In addition, they allow one to work with structured terms (and functors). However, as RMMs, they only consider first order Markov models. Furthermore, they do not smooth distributions using models of different specificity. Finally, MineSeqLog [Lee and De Raedt, 2003] is a frequent relational sequences miner that employs subsumption to test whether a pattern matches a relational sequence. One might consider employing it to tackle the first step in the r-gram Algorithm.

There are several directions for further work. These include: extending the framework to work with structured terms, considering back-off smoothing instead of interpolation, and, perhaps most importantly, applying the work to challenging artificial intelligence applications.

Acknowledgments

The authors would like to thank the anonymous reviewers, Kristian Kersting and Hannu Toivonen for valuable comments, and Mika Raento for making the Context Phone data available. The research was supported by the European Union IST programme, contract no. FP6-508861, *Application of Probabilistic Inductive Logic Programming II*.

References

[Anderson *et al.*, 2002] C. R. Anderson, P. Domingos, and D. S. Weld. Relational Markov Models and their Application to Adaptive Web Navigation. In *Proceedings of the 8th KDD*, pages 143–152. ACM, 2002.

[De Raedt and Kersting, 2003] L. De Raedt and K. Kersting. Probabilistic Logic Learning. *SIGKDD Explorations*, 5(1):31–48, 2003.

[Durbin *et al.*, 1998] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[Greenberg, 1988] S. Greenberg. Using Unix: Collected Traces of 168 Users. Technical report, Department of Computer Science, University of Calgary, Alberta, 1988.

[Hubbard *et al.*, 1997] T. Hubbard, A. Murzin, S. Brenner, and C. Chotia. *SCOP*: a Structural Classification of Proteins Database. *Nucleic Acids Res.*, 27(1):236–239, 1997.

[Jacobs and Blockeel, 2003] N. Jacobs and H. Blockeel. User Modeling with Sequential Data. In *Proceedings of the 10th HCI*, pages 557–561, 2003.

[Kersting and Gärtner, 2004] K. Kersting and T. Gärtner. Fisher Kernels for Logical Sequences. In *Proceedings of the 15th ECML*, volume 3201 of *Lecture Notes in Computer Science*, pages 205–216. Springer, 2004.

[Kersting *et al.*, 2006] K. Kersting, L. De Raedt, and T. Raiko. Logical Hidden Markov Models. *Journal of AI Research*, 25:425–456, 2006.

[Lee and De Raedt, 2003] S. D. Lee and L. De Raedt. Mining Logical Sequences Using SeqLog. In *Database Technology for Data Mining*, volume 2682 of *Lecture Notes in Computer Science*. Springer, 2003.

[Liao *et al.*, 2005] L. Liao, D. Fox, and H. A. Kautz. Location-based Activity Recognition using Relational Markov Networks. In *Proceedings of the 19th IJCAI*, pages 773–778. Professional Book Center, 2005.

[Manning and Schütze, 1999] C. H. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[Raento *et al.*, 2006] M. Raento, A. Oulasvirta, R. Petit, and H. Toivonen. ContextPhone - a Prototyping Platform for Context-aware Mobile Applications. *IEEE Pervasive Computing*, 4(2):51–59, 2006.

[Semeraro *et al.*, 1995] G. Semeraro, F. Esposito, and D. Malerba. Ideal Refinement of Datalog Programs. In *Proceedings of the 5th LOPSTR*, volume 1048 of *Lecture Notes in Computer Science*. Springer, 1995.