

Relational Transformation-based Tagging for Activity Recognition

Niels Landwehr

Bernd Gutmann

Ingo Thon

Luc De Raedt

Department of Computer Science

Katholieke Universiteit Leuven

Celestijnenlaan 200 A, B-3001 Heverlee, Belgium

firstname.lastname@cs.kuleuven.be

Matthai Philipose

Intel Research Seattle

1100 NE 45th Street

Seattle, WA 98105, USA

matthai.philipose@intel.com

Abstract. The ability to recognize human activities from sensory information is essential for developing the next generation of smart devices. Many human activity recognition tasks are — from a machine learning perspective — quite similar to tagging tasks in natural language processing. Motivated by this similarity, we develop a relational transformation-based tagging system based on inductive logic programming principles, which is able to cope with expressive relational representations as well as a background theory. The approach is experimentally evaluated on two activity recognition tasks and an information extraction task, and compared to Hidden Markov Models, one of the most popular and successful approaches for tagging.

Keywords: relational learning, sequence tagging, activity recognition, information extraction

1. Introduction

Smart systems that assist humans must be able to recognize the current context of the user and the activity she is performing in order to suggest or take actions in an intelligent manner. To recognize the context and activity, such systems can rely on streams of past activities, context, and sensory information (visual, object-interaction, ...). Recognizing the current activity or context then corresponds to inferring the activity or context from such sequential information. From a machine learning perspective, this task is akin to many tagging tasks pursued in statistical natural language processing [1]. For instance, in part-of-speech tagging, a form of "shallow parsing", the words in a sentence are to be labeled with the corresponding parts-of-speech (word categories). Many techniques have been developed and employed for this type of task, which has been coined *sequential supervised learning* by [2]. Two popular techniques for sequential supervised learning are Hidden Markov Models and transformation-based learning [3]. However, whereas Hidden Markov models have been applied in many different areas, ranging from speech-recognition to activity recognition and bioinformatics [4], to the best of the authors' knowledge, transformation-based learning has only seldomly been applied outside the field of natural language processing.

Because the structure of natural language is quite rigid as compared to that of typical activity recognition tasks, the existing transformation-based learners cannot directly be applied for activity recognition. Therefore, we develop a more flexible *relational* transformation-based tagger within the inductive logic programming paradigm. This does not only provide an *expressive* representation but also allows one to easily incorporate background theory during the learning process. Thus the key contribution of this paper is a relational extension of transformation-based tagging based upon inductive logic programming principles. It also extends earlier work on relational transformation-based learning by [5] in that it focuses on *tagging* rather than *classification*. More specifically, from inductive logic programming (and the work by [5]) our technique inherits its search and refinement techniques (including a branch-and-bound algorithm) and from transformation-based learning the error driven stacking of rules.

The proposed method is evaluated in two activity recognition domains: "Activities of Daily Living" (ADL) recognition from a stream of "object interaction" data [6], and mobile phone profile prediction based on data collected by [7]. As the method we develop originates from natural language processing, we also evaluate it in this context using an information extraction data set [8]. This should allow us to get insight into the promise of the method for natural language based applications. Experiments show that the obtained accuracies are competitive with those of HMM-based approaches, and that it is easy to incorporate human-supplied background knowledge into the learning process. Furthermore, and that is perhaps the key advantage of the relational transformation-based tagger, the method can easily be extended to deal with variants of the tagging problem, for instance the prediction of structured output tags (as in Logical Hidden Markov Models [9]), and to cope with rich background knowledge.

This paper is organized as follows: in Section 2, we introduce the problem of sequence tagging and briefly review transformation-based tagging and hidden Markov models; in Section 3, we upgrade the traditional transformation-based tagging approach to the use of relational sequences; in Section 4, we report on experiments in activity recognition and information extraction, and finally, in Section 5, we conclude and touch upon related work.

Algorithm 1 Basic transformation-based tagging algorithm.

tb-tagging(input: sequences S ; true sequence tags L)

```

1   $\hat{L} := \text{initial-tags}(S, L)$ 
2  initialize  $R = []$ 
3  repeat
4       $r := \text{find-best-rule}(S, \hat{L}, L)$ 
5      update  $\hat{L} := \text{apply-rule}(\hat{L}, r)$ 
6      update  $R := \text{append}(R, r)$ 
7  until (no improvement)
8  return  $R$ 

```

2. Sequence Tagging

Sequence tagging is the task of assigning to each element in a given sequence an appropriate label or *tag*. Let $W = \{w^1, \dots, w^k\}$ denote the vocabulary of sequence elements, and $T = \{t^1, \dots, t^m\}$ the vocabulary of tags. The most prominent instance of the tagging problem is part-of-speech-tagging in natural language processing, where the task is to assign lexical categories $t \in T$ to words $w \in W$ in a given natural language sentence [1]. Many other interesting sequence analysis problems can be cast in this framework, such as activity recognition in user modeling or gene finding and protein secondary structure prediction in bioinformatics [4, 2].

In natural language processing, the two most common tagging approaches are transformation-based taggers (rule-based) and probabilistic methods (hidden Markov models or related techniques) [1]. Both of these approaches yield competitive results, and have received a lot of attention. Before discussing our extension to transformation-based learning, we briefly review these two approaches in the next two sections.

2.1. Transformation-based Tagging

Transformation-based learning is a rule-based learning approach that iteratively stacks rules on top of each other to improve performance [3]. The basic transformation-based learning algorithm for the tagging problem is summarized in Algorithm 1. The algorithm takes as input a set S of sequences with known true tags L . During learning, it maintains a set of current tags \hat{L} for all $s \in S$. \hat{L} is initialized using some simple scheme, such as assigning to every element $w \in W$ its most common tag $t \in T$ in the training data (procedure *initial-tags*). The algorithm then tries to improve the current tagging \hat{L} with respect to the true tagging L by learning a list of *transformation rules* R . Transformation rules can re-tag sequence elements based on the context they appear in. A transformation rule has the form $t' \leftarrow t : \text{context}$ and simultaneously replaces all occurrences of tag t in all sequences with t' whenever the constraint *context* is satisfied.

Example 2.1. As an example from natural language processing, the word “move” could be initially tagged as “verb”, but would be re-tagged as “noun” if the preceding word was tagged as “article”. This

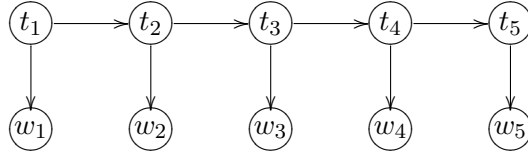


Figure 1. Example lattice generated by unrolling a tagging HMM to a sequence w_1, \dots, w_5 . Inference in this model is carried out with the Viterbi algorithm, which yields the most likely joint state of the hidden variables t_1, \dots, t_5 given the observations on w_1, \dots, w_5 .

can be encoded by the following transformation rule:

$$\textit{noun} \leftarrow \textit{verb} : \textit{word} = \textit{move}, \textit{preceding tag} = \textit{article}$$

The transformation rule languages employed in traditional transformation-based tagging are mostly simple instantiations of some template—for instance, querying in *context* the word and tag at the current position and the next or preceding position(s). We will replace this constraint by a first-order logical expression in Section 3.

In every iteration, the transformation rule yielding the greatest reduction in error between \hat{L} and L is greedily selected (*find-best-rule*), applied to the current tagging \hat{L} and appended to the rule list R . As conditions of rules in R match not only sequence elements but also currently predicted tags \hat{L} , rules can effectively bootstrap the current predictions. This makes transformation-based learning strictly more powerful than standard rule learning [3].

2.2. Hidden Markov Model Tagging

Tagging with hidden Markov models is typically performed with a model in which there is a hidden state q_t for every possible tag t , and state emission symbols correspond to symbols $w \in W$. That is, the observed sequence of symbols is seen as being generated by the hidden sequence of tags. Formally, the joint probability of an observation sequence $s = w_1 \dots w_n$ with hidden tag sequence $t_1 \dots t_n$ is given by

$$P(w_1 \dots w_n, t_1 \dots t_n) = P(t_1) \prod_{i=1}^{n-1} P(t_{i+1} | t_i) P(w_i | t_i)$$

where $P(t_1)$ is an initial probability for tag t_1 and $P(w_i | t_i)$, $P(t_i | t_{i-1})$ are conditional probabilities for the emitted word w_i and next tag t_{i+1} given the current tag t_i . When such a model is applied to a sequence $w_1 \dots w_n$, it is unrolled into a lattice as depicted in Figure 1, and the Viterbi algorithm [10] is employed to efficiently compute

$$\begin{aligned} \hat{t}_1 \dots \hat{t}_n &= \arg \max_{t_1 \dots t_n} P(t_1 \dots t_n, w_1 \dots w_n) \\ &= \arg \max_{t_1 \dots t_n} P(t_1 \dots t_n | w_1 \dots w_n), \end{aligned}$$

the most likely sequence of tags for the given sequence.

This technique has been used successfully for tagging problems in many domains. For instance, HMM-based approaches are a popular technique for inferring hidden user activities from a stream of



Figure 2. Relational representation of the ADL recognition problem. While a person is performing activities of daily living (such as preparing breakfast), a stream of object interaction data is generated from a wearable RFID reader (“sensor reading”). This can be represented in a relational form by collapsing identical sensor readings to one sequence element w_i , and encoding the starting point and duration of the observation in another predicate. Furthermore, additional background knowledge can be used to encode prior knowledge about the domain.

object-interaction data in the so-called ADL (“Activities of Daily Living”) problem [6, 11], which will be described in more detail below.

3. Relational Transformation-based Tagging

The general motivation for our work on relational transformation-based tagging is to apply the transformation-based tagging methodology to complex datastreams, which are generated, for instance, by sensors or sensor networks in ubiquitous computing environments. For such complex domains it is not always possible to represent all available information as flat (or *propositional*) symbols from a fixed alphabet. This problem can be overcome by using a more expressive *relational* representation for sequence elements. We will therefore extend the template-based rule language traditionally used in transformation-based learning to a more flexible *relational* rule language, which can take advantage of such richer representations for sequence elements. Furthermore, it is easy in this case to incorporate domain-specific background knowledge into the learning process. Analyzing such relational sequences has received considerable attention recently, for instance, with relational extensions of Hidden Markov Models [9] or n-gram models [12].

Example 3.1. As an example, consider the ADL (“Activities of Daily Living”) recognition problem, which is visualized in Figure 2. In ADL recognition, objects used in activities of daily living such as making breakfast are equipped with small RFID tags that can be picked up by a wearable reader while a person performs an activity [6]. The task is to recover the activity currently performed from the stream of sensor data, that is, to tag the sequence of object interactions with activities.

It is obvious that this kind of data is less rigidly structured than natural language data: there are no “grammatical rules” that determine the exact sequence of touching knife, toast, butter and jam when

adding flavor to a toast. Nevertheless, context information can help determine the right tag. For instance, using a spoon can indicate activities FlavorTea or EatCereals. This ambiguity can be resolved by looking at the context: the observation of a spoon closely followed by sugar indicates activity FlavorTea, while observation of a spoon after milk and cereals indicates activity EatCereals.

Furthermore, the stream of object data obtained from the sensor has some internal structure, as an object observation has a starting point and duration in time. A representation in first-order logic allows to capture this structure, and to express flexible rule conditions such as *object x has (not) been observed less than t seconds before/after the current time-step* or *the most frequent (currently estimated) tag around the current time-step is t* using manually defined background knowledge.

At the same time, activity recognition can be seen as a *data stream mining* task—the analysis of a continuous, potentially infinite stream of data. In this context, issues such as online learning (with only one pass through the data necessary) are of considerable interest. However, we will not address these issues in the paper, and instead assume that a limited amount of training data is given a priori. Extending the proposed methods to an online-learning scenario is an interesting direction for future work.

The next section introduces the formal learning setting for relational transformation-based tagging, before discussing learning algorithms and experimental results.

3.1. Learning Setting

The learning setting for relational transformation-based tagging can be formalized as follows:

Given

- a relational language \mathcal{W} for describing sequence elements, i.e., a set of typed first-order logical predicates
- a set of tags T ;
- a set of training sequences $S = \{s_1, \dots, s_m\}$ with sequence elements described in \mathcal{W} and corresponding true tags L over T ;
- a scheme for setting initial tags given by a function *init*;
- a language \mathcal{L} of transformation rules $t' \leftarrow t : q$ where $t, t' \in T$, $q = l_1, \dots, l_r$ and the l_i are atoms in \mathcal{W} .

Find an ordered lists of transformations $R = [R_1, \dots, R_l]$, $R_i \in \mathcal{L}$, such that applying the initial tagging scheme and the transformation rules R_1, \dots, R_l minimizes

$$error(\hat{L}) = \sum_{s \in S} \sum_{i=1}^{n_s} \delta(l_{is}, \hat{l}_{is})$$

where n_s is the length of sequence s and l_{is}, \hat{l}_{is} denotes the tag assigned to element i in sequence s according to L and \hat{L} .

In contrast to standard (propositional) transformation-based tagging approaches, the languages \mathcal{W} (sequence elements) and \mathcal{L} (rules) employed are relational; that is, rule conditions q are first-order queries

of the form l_1, \dots, l_k where the l_i are first-order logical atoms. Applying a first-order transformation rule $t' \leftarrow t : q$ means simultaneously replacing all tags t in \hat{L} by t' wherever the first-order context constraint q matches the relational description of the corresponding sequence element.

Example 3.2. As an example for a relational transformation rule in the ADL recognition domain consider

$$FlavorTea \leftarrow EatCereals : sensor(X, spoon), close(X, sugar, 10), not(close(X, bowl, 5))$$

where the variable X is bound to the sequence element under consideration and the background predicate $near/3$ is defined by

$$close(X, O, T) \leftarrow time(X, S, E), sensor(X', O), time(X', S', E'), dist(S, E, S', E', T'), T' \leq T$$

and $dist(S, E, S', E', T)$ measures the distance between the intervals $[S, E]$ and $[S', E']$. This rule re-tags *spoon* objects from *EatCereals* to *FlavorTea* if implied by the context.

3.2. A Branch-and-Bound Learning Algorithm

For learning the list R of relational transformation rules, a large space of possible rules has to be searched. However, the structure on the search space can be exploited to make this search more efficient. More specifically, the algorithm we use combines ideas from transformation-based learning (branch-and-bound search based on upper bounds for the error reduction of a transformation rule) and inductive logic programming (refinement search in a generalization/specialization lattice). It is closely related to the algorithm presented in [5].

Recall that the goal of learning is to find a list R of transformation rules that minimizes the $error(\hat{L})$ on a set of training sequences S with known true labels L . As in propositional transformation-based learning [3], the rule list is learned greedily: starting with an empty list, the algorithm incrementally adds one rule after the other, at every step selecting the rule which yields the greatest reduction in $error(\hat{L})$ and updating the current tagging \hat{L} (cf. Algorithm 1).

When searching for an individual rule with maximum error reduction, a significant part of the search space can be pruned away by computing upper bounds for the error reduction a rule can achieve. One obvious bound for the reduction achievable by a transformation rule $t^i \leftarrow t^j : context$ is given by the number of sequence elements whose true tag (in L) is t^i and which are currently (in \hat{L}) assigned tag t^j . Let \mathcal{M} denotes the current confusion matrix, i.e., $\mathcal{M}[i, j]$ denote the number of sequence elements with true tag t^i currently tagged as t^j . This can be exploited by considering rules $t^i \leftarrow t^j : context$ in (decreasing) order of their potential $\mathcal{M}[i, j]$ for error reduction and keeping track of the best error reduction Δ_{best} found so far. Now, all rules of the form $t^i \leftarrow t^j : context$ for which $\mathcal{M}[i, j] \leq \Delta_{best}$ can be removed from consideration (cf. [3]).

This idea can be taken one step further if it is combined with a general-to-specific search for the first-order constraint $context$ [5]. As a complete search in the space of first-order constraints is infeasible in most cases, we perform a greedy general-to-specific search. To generate the specializations of the current condition q , a so-called refinement operator ρ under θ -subsumption is employed. A condition q_1 θ -subsumes a condition q_2 if and only if there is a substitution θ such that $q_1\theta \subseteq q_2$. 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, and the application

Algorithm 2 Branch-and-bound algorithm for relational transformation-based tagging

```

rtb-tagging(input: sequences  $S$ ; true sequence tags  $L$ ; language bias  $\mathcal{L}$ )
1   $\hat{L} := \text{initial-tags}(S, L)$ 
2  initialize  $R := []$ 
3  repeat
4      initialize  $\Delta_{best} := 0$ 
5      compute  $\mathcal{M} := \text{confusion-matrix}(\hat{L}, L)$ 
6      for all  $i, j \in \{1, \dots, k\}, i \neq j$ , sorted by  $\mathcal{M}[i, j]$  descending do
7          initialize  $\Gamma := \mathcal{M}[i, j]$ 
8          initialize  $q := \text{true}$ 
9          while ( $\Gamma > \Delta_{best}$ ) do
10             for all  $q' \in \rho(q, \mathcal{L})$  do
11                 compute  $\Delta_{q'} := \text{error-reduction}(t^j \leftarrow t^i : q')$ 
12                 compute  $\Gamma_{q'} := \text{max-reduction}(t^j \leftarrow t^i : q')$ 
13             end for
14             let  $q := \text{argmax}_{q'} \Delta_{q'}$ 
15             let  $\Delta_{best} := \max(\Delta_{best}, \Delta_q)$ 
16             let  $\Gamma := \Gamma_q$ 
17         end while
18     end for
19     let  $r := t^i \leftarrow t^j : q$  be a rule with error reduction  $\Delta_{best}$ 
20     update  $\hat{L} := \text{apply-rule}(\hat{L}, r)$ 
21     update  $R := \text{append}(R, r)$ 
22 until (no improvement)
23 return  $R$ 

```

of the substitution replaces the variables V_1, \dots, V_l by the corresponding terms t_1, \dots, t_l . $\rho(q)$ typically returns all minimal specializations of q within \mathcal{L} . For our purposes, the refinement operator specializes a condition $q = l_1, \dots, l_n$ simply by adding a new literal l to the clause yielding $h \leftarrow l_1, \dots, l_n, l$. This operator is monotone in the sense that for $q' \in \rho(q)$ the number of matches in the data can only decrease. Consequently, the maximum gain achievable from specializations of a transformation rule $t^i \leftarrow t^j : q$ can be bounded in terms of the current matches. More specifically, assume that a constraint q matches a number of sequence elements in the training data S , and that for p_q of these it has a positive effect (current tag is t^j , but true tag is t^i) and for n_q it has a negative effect (current and true tag are t^j). The error reduction of applying the transformation $t^i \leftarrow t^j : q$ is $\Delta_q = p_q - n_q$. It is now obvious that no specialization $t^i \leftarrow t^j : q'$ with $q' \in \rho^*(q)$ can achieve an error reduction greater than $\Gamma_q = p_q$.

A greedy branch-and-bound algorithm exploiting these two bounds is outlined in Algorithm 2. It takes as input a set of training sequences S , true sequence tags L , and the language bias \mathcal{L} . The algorithm starts with an empty rule list R and initial tags assigned in \hat{L} . Transformation rules are then greedily added to R , and their effect applied to the current tagging \hat{L} (lines 3–21). Transformations are considered

Relation	Description
$sensor(Id, Object)$	The object observed at sequence element Id is $Object$
$duration(Id, T)$	The object observation at sequence element Id lasted T seconds
$close(Id, Obj, T)$	The object Obj has been observed within T seconds of sequence element Id
$time_bin(T, Bin)$	The time span T falls into the bin $Bin \in \{short, medium, long\}$
$closest_tag(Id, Act)$	The closest sequence position to Id for which an activity (i.e., a tag \neq “no activity”) is assigned in \hat{L} is tagged with Act
$close_used(Id, Act, T)$	Less than T seconds away from sequence element Id an object has been observed which is typically used in Act

Table 1. Example relations used to describe the activity data. Some relations are directly derived from the data (e.g. $sensor$, $duration$, $close$), others include human-supplied prior knowledge (e.g. $close_used$).

in order of decreasing $\mathcal{M}[i, j]$ (line 6). At every step of the search for a single transformation $t^i \leftarrow t^j : q$ (lines 6–18), the algorithm keeps track of the largest reduction Δ_{best} achieved by a rule so far. During refinements of the context constraint q (lines 9–17) a bound Γ_q for the maximum reduction that any specialization of a rule q can still achieve is computed (max-reduction), and only parts of the search space for which Γ is greater than Δ_{best} are explored.

4. Experiments

The proposed method was implemented in the RETRO (for Relational TransfORMation-based tagging) system. We first evaluate the system in two different real-world activity recognition domains: recognition of *activities of daily living* (ADL) from object interaction data, and mobile phone profile prediction from traces of provider cells and phone communication events. As our method originated in the area of natural language processing, we also apply it to a challenging NLP problem that is receiving increasing attention: *information extraction*, the automated extraction of relational facts from text sources. More specifically, we consider the automatic extraction of protein subcellular localization information from thousands of sentences from the biomedical literature [8].

4.1. Activity Recognition

Broadly speaking, the goal in activity recognition is to infer context information about a user, such as the activity currently performed, from a sequence of sensor observations. The first experimental domain we consider is a classical activity recognition domain, namely recognition of activities of daily living (ADL). In ADL recognition, object-interaction data for a user having breakfast at home has been gathered by a wearable RFID reader and RFID tags on objects such as milk, cereals, kettle, water tap, cutlery etc. (23 objects in total). The stream of tags picked up by the RFID reader indicates which object is close (approximately 10–15 centimeters) to the wrist of the user at a particular point in time. A single object observation is returned at every second—if several tags are within reach, one is returned randomly. Note that the data is relatively noisy: tags might sometimes be missed, or a tag not related to a particular activity can be reported by the reader because the corresponding object is accidentally close. The task is to predict the current activity performed, out of a set of 24 possible activities such as boiling

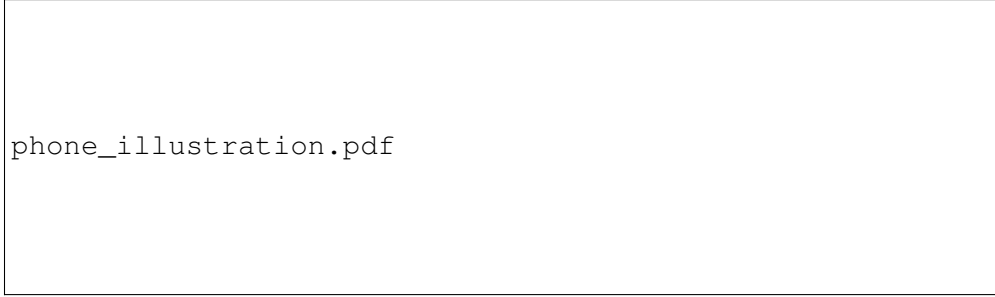


Figure 3. Illustration of the Phone data (predicates for cell location, duration, user activity, active applications, and communication events).

water, toasting bread, reading a newspaper or “no activity”. The sequence data obtained from the RFID reader is represented in relational form by collapsing identical observations into one observation with a starting point and duration in time (cf. Figure 2 for an illustration). Furthermore, additional background predicates have been defined, see Table 1 for examples. The dataset contains a total of 20 sequences with 4597 time points to tag.

The second activity recognition task we consider is mobile phone profile prediction (**Phone**). The profile of a cellular phone (silent, meeting, or normal) is context information that is related to the activity a user is currently performing, e.g. whether she is in a meeting and therefore not reachable or in the office and ready to take calls. If a device was able to anticipate such context information it could adapt to the current situation, e.g. by turning down the volume of a phone if it suspects the user to be in a meeting. The data we use was extracted from a dataset¹ published within the *Context Phone* project [7]. In this project, data about the communication behavior of 12 users has been gathered over a long period of time, using software running on Nokia Smartphones. The software automatically logs communication and context data, such as the current provider cell, incoming and outgoing calls and text messages, and other phone status information. We built a joint model for all users, restricted to logs of at least 100 events with at least 5 profile changes. The task is to predict the current profile of the phone (silent, meeting, or normal) at every point in time. See Figure 3 for an illustration of the data and the predicates used. The dataset consist of 10 sequences with 1735 time points to tag.

To initialize the tagging \hat{L} in the transformation-based tagger, RETRO simply assigns the most frequent tag given the propositional symbol $w \in W$ observed at the current sequence position:

$$\text{init}(w) = \underset{t \in T}{\text{argmax}} C(w, t)$$

where $C(w, t)$ is the number of times symbol w was tagged with t in the training data. The propositional alphabet W used are the sensor observations (objects) for the **ADL** domain, the mobile phone cells for the **Phone** domain, and the actual words in the information extraction domain discussed in Section 4.2. More elaborate initialization schemes (such as using the HMM tagging as an initialization for the transformation-based tagger) are an interesting direction for future work.

The basic search algorithm described in Section 3.2 performs a greedy search through the space of rules. However, it is well-known that greedy search suffers from premature convergence in local optima

¹<http://www.cs.helsinki.fi/group/context/#data>

of the search space. The most straightforward extension of greedy search is to use a beam search, where the best k candidate rules at every step in the search are kept and refined further. In all experiments, we use a beam search with beam size $k = 10$. Furthermore, we keep a set of k best scoring rules, and the refinement search terminates if this set does not change between two successive steps in the search. The main loop of the algorithm is terminated if no rule with a gain of at least $min_gain = 10$ is found. The number of literals in a rule was limited to $r_{max} = 3$ for **ADL** and $r_{max} = 2$ for **Phone**.

RETRO was compared to the following other tagging methods. First, we have conducted experiments with two propositional approaches: HMMs and *conditional random fields* (CRFs) [13]. CRFs model the conditional probability $P(Y|X)$, where X is an input sequence and Y is the corresponding label sequence, and are trained discriminately. We used the *CRF package* implementation² by S. Sarawagi, which implements the algorithm described in [14]. As it is not possible to encode all relevant information propositionally, the most relevant information was selected to be used as the propositional alphabet W for the HMM and CRF taggers. For **ADL**, this is the sequence of objects observed, and for **Phone**, it is the sequence of cells the phone was located in. The CRF tagger was run using standard settings, 50 iterations for the training algorithm, and with features corresponding to the same data representation as used with the HMM.

Furthermore, we performed experiments with TILDECRF [15], a relational extension of CRFs. TildeCRF was given the relational encoding used for RETRO, the same background knowledge, and the same language bias. It was trained using Gradient Tree Boosting as described in [15] with a maximum tree depth of 4 for the **ADL** dataset and 10 for the **Phone** dataset. To prevent overfitting, we restricted the model to one tree per class, which made 19 trees in total for **ADL** and 3 for **Phone**. The predictions from TildeCRF were obtained with the forward-backward algorithm, as this gave slightly better results than the default Viterbi. Finally, we also give results for MAJORITY TAG (most frequent tag observed in the training data) and MAJORITY TAG PER OBSERVATION (the tag most frequently observed in the training with the current propositional observation) as two simple baselines.

The systems are evaluated in a leave-one-sequence-out cross-validation. On the respective test data, precision and recall are calculated. Precision is the number of tags correctly predicted divided by the total number of tags predicted (i.e., the sequence positions for which a tag $t \neq$ "no activity" was assigned by the system). Recall is the number of tags correctly predicted divided by the number of sequence positions for which a true activity $t \neq$ "no activity" was observed. Note that in the phone domain no "no activity" tag exists, so precision and recall coincide with tagging accuracy. From precision and recall, we computed the F-measure

$$F = \frac{2 \cdot precision \cdot recall}{precision + recall},$$

and averaged over the different folds of the cross-validation.

Table 2 shows the results for RETRO, TILDECRF, CRF and HMM in the **ADL** and **Phone** domains. In the **ADL** domain, RETRO and HMM perform comparably well, TILDECRF slightly weaker, and the propositional CRF again weaker. However, the weak result of CRF is due to one relatively short sequence which is tagged very badly (F-measure of 0 as the tagger fails to pick up any activity). Note that all tested approaches outperform the simple MAJORITY TAG and MAJORITY TAG PER OBSERVATION baselines. The second domain, **Phone** profile prediction, is significantly more challenging. Both the HMM and CRF/TILDECRF tagger fail to improve upon the MAJORITY TAG prediction, while RETRO

²<http://crf.sourceforge.net/>

Algorithm	ADL	Phone
MAJORITY TAG	19.5 ± 22.3	56.7 ± 13.1
MAJORITY TAG PER OBSERVATION	54.5 ± 10.8	61.4 ± 11.4
HMM	74.9 ± 12.5	56.7 ± 13.1
CRF	61.3 ± 23.2	57.2 ± 21.3
TILDECRF	72.4 ± 12.3	56.6 ± 15.1
RETRO	75.4 ± 7.8	67.7 ± 10.3

Table 2. Average F-measure on the **ADL** Recognition and **Phone** problems based on a leave-one-sequence-out cross-validation.

Learned Rules		
<i>ObtainNewspaper</i>	←	<i>ReadNewspaper</i> : $close(Id, Obj, T), Obj = door, time_bin(T, medium)$
<i>FlavorTea</i>	←	<i>EatCereals</i> : $closest_tag(A, FlavorTea)$
<i>SteepTeaBag</i>	←	<i>DrinkTea</i> : $close(Id, Obj, T), Obj = stove$
<i>PourCereal</i>	←	<i>ObtainNewspaper</i> : $close_used(Id, PourCereal, T),$ $not(close_used(Id, ObtainNewspaper, T')), time_bin(T, short)$
<i>SteepTeaBag</i>	←	<i>noActivity</i> : $duration(Id, T), time_bin(T, long), closest_tag(ID, SteepTeaBag)$

Table 3. Examples for rules learned by RETRO on the **ADL** dataset.

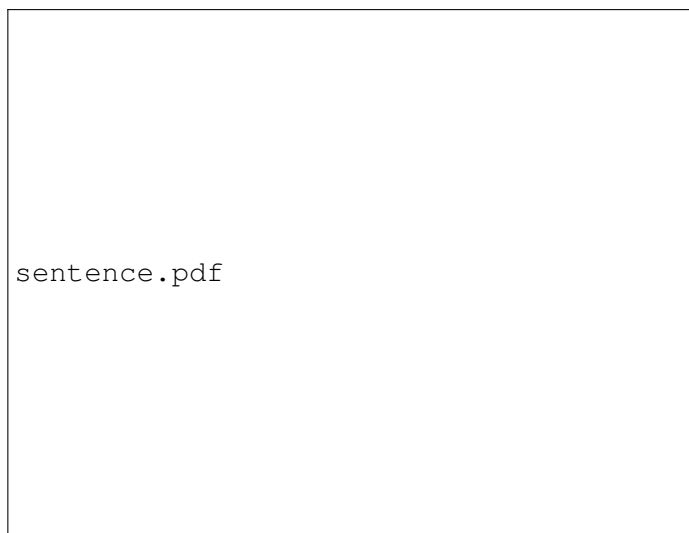
yields a (borderline) significant increase in F-measure compared to MAJORITY TAG (paired two-sided t-test, $p = 0.051$).

Experiments in the **Phone** domain are still preliminary to a certain degree. Predicting the profile of a user’s phone based on the data currently available appears to be a very hard problem; further experimentation and better data will be needed to obtain models that are accurate enough to be of practical relevance. Nevertheless, context-recognition techniques such as the discussed phone profile prediction hold much promise in creating smarter devices if prediction accuracy can be improved. Note furthermore that although HMM tagging is a standard approach in activity recognition, more advanced probabilistic methods based on dynamic Bayesian networks have recently been developed, which would possibly yield slightly higher accuracy in this domain [16].

Examples for rules learned by RETRO on **ADL** are shown in Table 3. For instance, consider the last rule: it states that if a sequence element corresponding to a long object observation is tagged with *noActivity* and the closest currently predicted activity is *SteepTeaBag*, this sequence element should also be tagged with *SteepTeaBag*. This rule is useful for “filling in gaps” as *SteepTeaBag* only causes characteristic object observations at the beginning and end of the activity.

4.2. Information Extraction

Another interesting instance of the tagging problem is *information extraction*. Assume we are given a set of natural language sentences, for instance from a medical paper, and possibly additional information such as the grammatical structure of the sentences or a-priori domain knowledge. Information extraction



sentence.pdf

Table 4. Representation of a sentence in the **Subcellular-Localization** dataset, with flattened sentence structure information and target tags *PROTEIN* and *LOCATION*. The sentence contains two *subcellular-localization* tuples.

now aims at recovering instances of a particular semantic relation, such as the localization of a protein p in a certain cell compartment l , from such syntactic sentence data. Such a mapping from unstructured (text) information to structured (relational) information has the potential to increase the utility of text data, by allowing for easier information retrieval or the automatic population of semantic databases from text sources.

The information extraction task considered here is to automatically infer the localization of proteins in the cell from abstracts of medical papers extracted from the MEDLINE database [17]. The dataset, which we will refer to as **Subcellular-Localization**, is described in more detail in [8], whose authors made it publicly available³. It comprises 7129 sentences annotated with instances of the *subcellular-localization* relation. In addition to word tokens, information about the grammatical structure of the sentences is available. More specifically, sentences have been parsed with the Sundance parser [18], and the resulting parse trees have been flattened into a two-level description: the first level represents the sentence as a sequence of phrase segments, and the second level individual tokens and their part-of-speech tag. Figure 4, taken from [8], visualizes the representation of sentences. In this figure, rows correspond to sequence positions 1, ..., 15 in the given sentence. The first column contains phrase segment information, the second column part-of-speech tags, and the third column the word tokens. The 7129 sentences contain a total of 149303 words to tag.

For use in the RETRO system, the data has been represented in a relational form as outlined in Table 5. The given relational representation allows one to build rich relational rules for tagging sequence elements (an example of a learned rule set is given below). Note that background predicates such as $alphanumeric(W)$ encode some prior knowledge about the domain, namely that proteins typically carry alphanumeric names (though not all proteins carry such names, and vice versa, not all alphanumeric words in the dataset are tagged as proteins).

³<http://www.biostat.wisc.edu/~craven/ie/>

Relation	Description
$pos_tag_at(Id, POS)$	The part-of-speech tag at sequence element Id is POS .
$phrase_seg_at(Id, PHR)$	The phrase segment at sequence element Id is PHR .
$word(Id, W)$	The word token at sequence element Id is W .
$prefix(W, W1, W2)$	The word W consists of prefix $W1$ and postfix $W2$.
$alphanumeric(W)$	The word W is alphanumeric (characters followed by numbers).
$is_char(W, CHR)$	The word W consists of the single character CHR .
$location_word(W, F)$	W is tagged with LOCATION in at least fraction F of the cases in the train data.
$location_detected(S)$	Some position in sentence S has been tagged by the system as LOCATION.

Table 5. Relations used for representing data in the **Subcellular-Localization** domain. Some relations are directly derived from the data (e.g. $pos_tag_at, word$), others include human-supplied prior knowledge (e.g. $alphanumeric$).

The experimental setup is now as follows. The available data is split into the same five folds as in [8], and a cross-validation is performed. On the respective training set, known true instances of the *subcellular-localization* relation are tagged with PROTEIN and LOCATION. Note that the dataset is very unbalanced: only about 10% of all sentences contain any relation instances (“positive sentences”). We used all of the positive sentences but randomly sampled only 10% of the negative sentences for the training set. On this training set, RETRO is used to infer a set of tagging rules. Parameters are set as for the activity recognition tasks, except that $min_gain = 5$ and the maximum number of literals in a rule is $r_{max} = 7$. The learned rules are then applied to tag sequences in the test data with PROTEIN and LOCATION accordingly. However, note that this does not yet fully solve the information extraction problem: we are interested in extracting relation instances from the test sentence, and unless there is exactly one predicted PROTEIN and one predicted LOCATION tag in the sentence it is not clear how to do this.

The most elegant solution to this problem would be to use a relational representation for *tags* as well as for the sequence data. In this case, relation instances could be directly represented at the level of tags. However, we leave this approach for future work and instead heuristically derive instances of the *subcellular-localization* relation from the predicted tags on the test data as follows. Let S denote a test sentence, and assume that n elements in S have been tagged as PROTEIN and m elements in S have been tagged as LOCATION. If $n = 0$ or $m = 0$, no relation instances are predicted. If $n = 1$ or $m = 1$, all pairs $(protein, location)$ with $protein$ tagged as PROTEIN and $location$ tagged as LOCATION are returned as relation instances. Finally, if $n = m$, n pairs $(protein_i, location_i)$ are returned, where $protein_i$ is the i -th sequence element tagged as PROTEIN and $location_i$ is the i -th sequence element tagged as LOCATION. There are very few test sentences for which none of the above conditions hold; in those cases it is unclear how to extract relation instances and thus none are predicted.

As outlined in [8], the extracted set of relation instances on the test data can be compared against the true set of relation instances by computing the precision and recall. Precision is the fraction of predicted relation instances that are true relation instances. Recall is the fraction of true relation instances that have been predicted by the system. The set of tagging rules returned by RETRO is deterministic, thus a rule set only produces a single point in precision-recall space. It is therefore not possible to trade off recall vs. precision as, e.g. in a probabilistic model by varying a decision threshold. However, the

pr_plot.pdf

results_craven.pdf

Figure 4. Left Figure: precision vs. recall for relational transformation-based tagging on the *subcellular-localization* data set. Individual points in PR space are obtained by varying the reweighting factor α , with $\alpha \in \{0.25, 0.5, 0.75, \dots, 4.5\}$. For $\alpha > 4.5$ no rules with reasonable coverage on the training set are found. Right Figure (taken from [8]): precision vs. recall measured on the same splits into training/test set for different versions of hierarchical and standard HMMs.

trade-off between precision and recall can be influenced, before training takes place; by giving a different weight to type one errors (or false positives) vs. type two errors (or false negatives). Here, false positives and negatives refer to the special tag NONE (negative) as opposed to the other two tags PROTEIN and LOCATION (positives). More specifically, assume that a particular rule R corrects t_p false negatives and t_n false positives, but at the same time introduces f_n false negatives and f_p false positives. Define

$$gain(R) = t_p - f_n + \alpha(t_n - f_p) \quad (1)$$

as the error reduction achieved by such a rule, where α is a reweighting factor that trades off type one vs. type two errors. For $\alpha > 1$, type one errors are weighted higher, which favors rule sets with high precision but low recall. Accordingly, $\alpha < 1$ gives a higher weight to type two errors and thus favors rule sets with high recall but low precision. Note that $\alpha = 1$ corresponds to the standard error measure. It is straightforward to adapt the branch-and-bound search methodology discussed in Section 3.2 to incorporate the reweighting factor α , by weighting the potential error reduction of a rule accordingly.

Figure 4, left plot, shows the precision and recall obtained on **Subcellular-Localization** for different α values (as defined by Equation 1). Precision and recall are separately averaged over the five folds of the cross-validation. For high *alpha* values, the system fails to learn rules with a reasonable coverage on the training set. The precision-recall curve has therefore been conservatively extended to the left to facilitate the comparison.

Figure 4, right plot, shows results obtained by different versions of HMMs in the same setting, as reported in [8]. The most successful version of HMMs on this task are Context HMMs, a version of hierarchical HMMs specifically tailored to this information extraction problem. The precision-recall curve

Learned Rules		
<i>Location</i>	←	<i>None</i> : word(A,D),location_word(D,0.67)
<i>Protein</i>	←	<i>None</i> : word(A,D),prefix(D,E,F),alphanumeric(E),location_detected(A)
<i>Location</i>	←	<i>None</i> : word(A,D),location_word(D,0.30),prefix(D,E,F),is_char(E,'v'),phrase_seg_at(A,npSeg,-1)
<i>Protein</i>	←	<i>None</i> : word(A,D),prefix(D,E,F),alphanumeric(E),location_detected(A)
<i>None</i>	←	<i>Protein</i> : pos_tag_at(A,unk,-1),word(A,D),prefix(D,E,F),alphanumeric(F)
<i>None</i>	←	<i>Protein</i> : word(A,D),prefix(D,E,F),is_char(E,'u'),phrase_seg_at(A,npSeg,-1)
<i>None</i>	←	<i>Protein</i> : word(A,D),prefix(D,E,F),is_char(E,'r'),alphanumeric(F)
<i>None</i>	←	<i>Protein</i> : phrase_seg_at(A,npConjSeg,-1),pos_tag_at(A,n,-2)
<i>None</i>	←	<i>Location</i> : pos_tag_at(A,conj,-1),phrase_seg_at(A,npSeg,-1)
<i>Location</i>	←	<i>None</i> : word(A,D),location_word(D,0.20),phrase_seg_at(A,vpSeg,-2),prefix(D,E,F),is_char(E,'c')

Table 6. An example rule set learned by RETRO on the **Subcellular-Localization** dataset ($\alpha = 3.0$).

of RETRO shows a similar behavior as for the Context HMM, but achieves slightly higher recall values at the same precision on average. This is quite remarkable as Context HMMs have been specifically designed for this problem. On the other hand, we have supplied the RETRO system with hand-crafted domain-specific background knowledge, which has a similar effect of adapting the system for a particular task. It is the easy incorporation of prior domain knowledge that makes relational (or logic-based) approaches so flexible and easy to adapt to different learning domains.

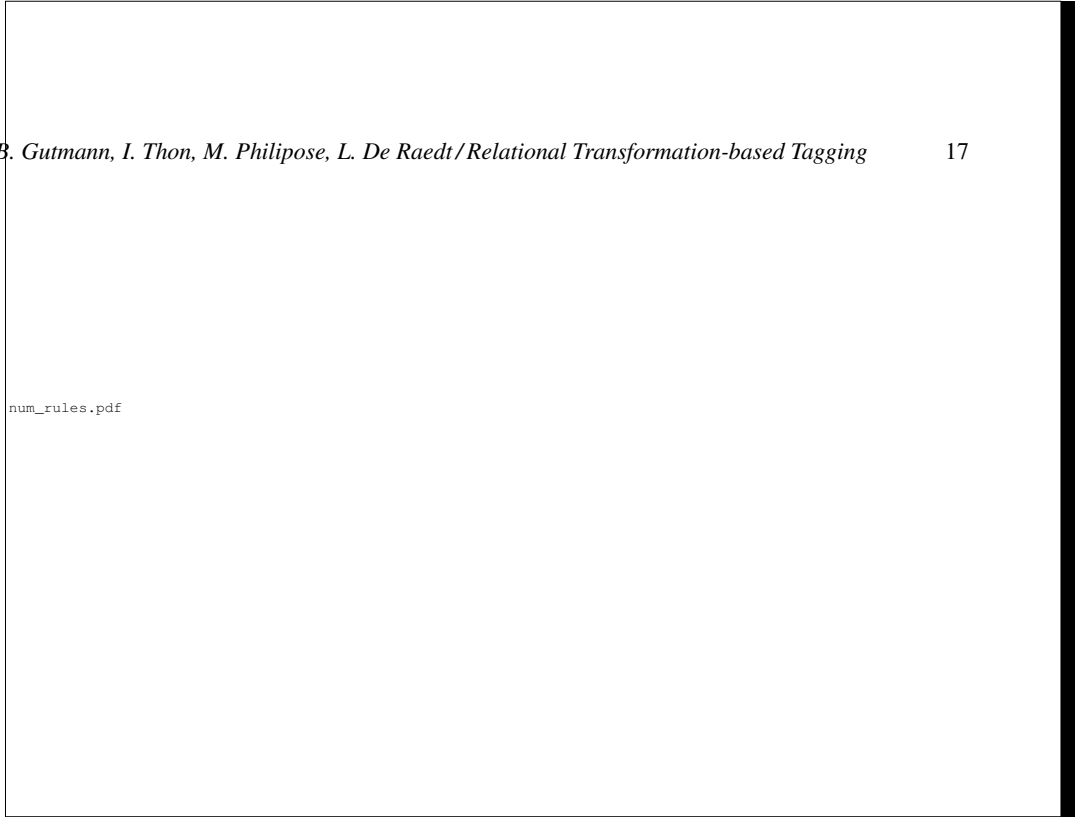
An example for a learned rule set is given in Table 6. The system basically starts by tagging words as LOCATION if they have been labeled LOCATION in two-thirds of all cases in the training set (rule 1). It then bootstraps its own prediction on LOCATION, and labels words that start with an alphanumeric expression as PROTEIN if a LOCATION has already been detected in that sentence (rule 2). Rule 3 will tag more words with LOCATION if the word has been tagged as LOCATION in some cases on the training set, and a certain grammatical restriction is satisfied. Rule 4 simply re-applies rule 2 to bootstrap these new LOCATION predictions to generate additional PROTEIN tags. The remaining rules mostly remove false-positives by looking at the grammatical structure of the sentence.

As a further illustration of how the reweighting factor affects the rule sets found by the system, Figure 5 shows the number of rules learned by the system depending on different α values. It is relatively easy to achieve high recall but low precision (low α , small rule set), but a more elaborate rule set is needed to achieve a high precision (high α).

Finally, note that an obvious disadvantage of a deterministic system such as RETRO compared to probabilistic approaches such as HMMs is that the precision-recall trade-off has to be determined *before* learning takes place, that is, the system has to be re-run to generate every individual point on the precision-recall curve shown in Figure 4. The systematic experiments needed to generate this curve were run on the VIC/HPC cluster of the Katholieke Universiteit Leuven, and took about 700 hours (or one month) of CPU time in total (AMD Opteron 250 cores, 2.2GHz). Averaged over all values of α and all folds, inducing a single rule set with RETRO took about 7.5 hours.

4.3. Effectiveness of Pruning

Figure 6 visualizes the effectiveness of the pruning schemes based on the two upper bounds discussed in Section 3.2 for the three tagging problems considered. More specifically, Figure 6, left plot, shows



num_rules.pdf

Figure 5. Number of rules learned by RETRO on the **Subcellular-Localization** task as a function of the reweighting factor α . The number of rules is averaged over the models learned on the different folds of the cross-validation.

the fraction of pairs (t^i, t^j) that have to be considered when searching for rules $t^i \leftarrow t^j$ in lines 6–18 of Algorithm 2 as a function of the algorithm iteration. This pruning scheme is very effective if the number of different tags is large, as in the **ADL** problem. There, it reduces the search space by 93%–99%. On the other two tasks it achieves a reduction of about 60%. It is also typically more effective in earlier iterations of the algorithm, when it is easier to find a rule that yields a large reduction in error.

Figure 6, right plot, shows which fraction of refinements is removed from the beam when rules are refined in lines 10–13 of Algorithm 2 because no further specialization can reach the performance of the best rule found so far. Note that this form of pruning basically allows for a more thorough search through the space of possible rules given a limited beam size) by effectively reducing the branching factor of the search. Indirectly, it can also reduce overall computational complexity if a good error reduction is achieved more quickly due to the more focused search. This pruning mechanism is particularly effective in the information extraction task **Subcellular-Localization**, but also prunes a significant part of the search space for the two activity recognition tasks **ADL** and **Phone**.

5. Conclusions and Related Work

Motivated by the needs of activity recognition problems, we have introduced a relational transformation-based tagging system. It tightly integrates principles of inductive logic programming (especially search, representations, operators, background knowledge) with transformation-based tagging (error-driven search, branch-and-bound idea). The approach has been evaluated on two activity recognition data sets as well as an information extraction task. The results are competitive with those of hidden Markov models (in activity recognition) and also extensions of HMMs such as hierarchical HMMs (in

pruning1.pdf

pruning2.pdf

Figure 6. Effectiveness of the two pruning schemes Bound I (maximum gain attainable from changing a certain tag into a certain other tag) and Bound II (maximum gain attainable from specializing a given rule). Results are averaged over a leave-one-sequence-out cross-validation (**ADL** and **Phone**) or a five-fold cross-validation (**Subcellular-Localization**).

information extraction). Perhaps more important than the experimental results obtained so far is the ease with which one can extend the transformation-based tagging approach beyond the propositional setting typically used with HMMs. There are several important directions in this regard. One direction we have already partially explored is the use of rich sources of background knowledge (that take not only into account the inputs but also the already available produced tags). Another direction is the prediction of structured output sequences (predicting sequences of logical atoms, cf. [19], such as a tag *call(anna,10)* denoting the prediction that *anna* will be called in 10 minutes. Finally, relational representations also allow to relax the purely sequential nature of the output (which can be important e.g. in ADL prediction, where different activities may overlap in time and therefore ordering them is not always possible).

Acknowledgments We would like to thank Mika Raento and Hannu Toivonen for making the Context Phone data available. The authors would like to acknowledge support for this work from the Research Foundation-Flanders (FWO-Vlaanderen), and GOA/08/008 project “Probabilistic Logic Learning”.

References

- [1] Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. The MIT Press (1997)
- [2] Dietterich, T.G.: Machine learning for sequential data: A review. In Caelli, T., Amin, A., Duin, R.P.W., Kamel, M.S., de Ridder, D., eds.: Proceedings of the Joint Conference on Structural, Syntactic, and Statistical Pattern Recognition. Volume 2396 of Lecture Notes in Computer Science. (2002) 15–30
- [3] Brill, E.: Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics* **21**(4) (1995) 543–565

- [4] Durbin, R., Eddy, S., Krogh, A., Mitchison, G.: *Biological Sequence Analysis*. Cambridge University Press (1998)
- [5] Dehaspe, L., Forrier, M.: Transformation-based learning meets frequent pattern discovery. In Cussens, J., ed.: *Proceedings of the 1st Workshop on Learning Language in Logic*, Bled, Slovenia (1999) 40–51
- [6] Patterson, D., Fox, D., Kautz, H., Philipose, M.: Fine-grained activity recognition by aggregating abstract object usage. In: *Proceedings of ISWC 2005*, Osaka (2005)
- [7] Raento, M., Oulasvirta, A., Petit, R., Toivonen, H.: ContextPhone - a Prototyping Platform for Context-aware Mobile Applications. *IEEE Pervasive Computing* **4**(2) (2006) 51–59
- [8] Skounakis, M., Craven, M., Ray, S.: Hierarchical hidden Markov models for information extraction. In: *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, Acapulco, Mexico (2003)
- [9] Kersting, K., De Raedt, L., Raiko, T.: Logical hidden Markov models. *Journal of Artificial Intelligence Research* **25** (2006) 425–456
- [10] Rabiner, L.: A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE* **77**(2) (1989) 257–286
- [11] Wilson, D., Philipose, M.: Maximum a posteriori path estimation with input trace perturbation: Algorithms and application to credible rating of human routines. In: *Proceedings of IJCAI 2005*, Edinburgh, Scotland (August 2005)
- [12] Landwehr, N., De Raedt, L.: r-grams: Relational grams. In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, Hyderabad, India (2007) 907–912
- [13] Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: *Proc. 18th International Conf. on Machine Learning*, Morgan Kaufmann, San Francisco, CA (2001) 282–289
- [14] Sha, F., Pereira, F.: Shallow parsing with conditional random fields. In: *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, Morristown, NJ, USA, Association for Computational Linguistics (2003) 134–141
- [15] Gutmann, B., Kersting, K.: TildeCRF: Conditional random fields for logical sequences. In Fürnkranz, J., Scheffer, T., Spiliopoulou, M., eds.: *Proceedings of the 15th European Conference on Machine Learning (ECML-2006)*. Volume 4212 of *LNAI (Lecture Notes in Artificial Intelligence)*, Berlin, Germany, Springer (September 2006) 174–185
- [16] Wang, S., Pentney, W., Popescu, A.M., Choudhury, T., Philipose, M.: Common sense based joint training of human activity recognizers. In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence*. (2007) 2237–2242
- [17] National Library of Medicine: The MEDLINE database (2003) <http://www.ncbi.nlm.nih.gov/PubMed/>.
- [18] E. Riloff: The sundance sentence analyzer (1998) <http://www.cs.utah.edu/projects/nlp/>.
- [19] Kersting, K., De Raedt, L., Gutmann, B., Karwath, A., Landwehr, N.: Relational sequence learning. In De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S., eds.: *Probabilistic Inductive Logic Programming*. Volume 4911/2008 of *Lecture Notes in Computer Science*. Springer (February 2008) 28–55